Oracle® Financial Analyzer

Application Programming Interface: Structure and Supporting Meta Data Guide

Release 11*i*

August 2000

Part No. A86119-01

Oracle Financial Analyzer Application Programming Interface: Structure and Supporting Meta Data Guide, Release 11*i*

Part No. A86119-01

Contents

Send Us Your Comments

Application Programming Interface: Structure and Supporting Meta Data Guide, Release 11*i*

Part No. A86119-01

Oracle Corporation welcomes your comments and suggestions on the quality and usefulness of this publication. Your input is an important part of the information used for revision.

- Did you find any errors?
- Is the information clearly presented?
- Do you need more information? If so, where?
- Are the examples correct? Do you need more examples?
- What features did you like most about this manual?

If you find any errors or have any other suggestions for improvement, please indicate the chapter, section, and page number (if available). You can send comments to us in the following ways:

- electronic mail - olapdoc@us.oracle.com
- FAX - 781-684-5880 Attn: Oracle Financial Analyzer
- postal service:
  OLAP Products Documentation Manager
  Oracle Corporation
  200 Fifth Avenue
  Waltham, MA 02451-8720
  USA

If you would like a reply, please give your name, address, and telephone number below.

Preface

Introduction

What this manual is about

The *Application Programming Interface: Structure and Supporting Meta Data Guide* describes how to use the Oracle® Financial Analyzer (hereinafter referred to as "Financial Analyzer") application programming interface (API) to maintain structure and supporting meta data.

Intended audience

This manual is intended for advanced users who are responsible for implementing the system and who want to use programs to maintain Financial Analyzer structure and supporting metadata.

Structure of this document

The Application Programming Interface: Structure and Supporting Meta Data Guide is structured as follows:

- Chapter 1 provides an overview of using the API to maintain structure and supporting meta data.
- Chapter 2 describes the programs that you can use to maintain structure and supporting meta data.

Related Documentation

Books

This manual is part of a set of documentation, which also includes the following books:

- *Oracle Financial Analyzer User's Guide* — Describes the Financial Analyzer user environment and provides information about database maintenance and document customization tasks.
- *Oracle Financial Analyzer Installation and Upgrade Guide* — Describes how to install various types of Financial Analyzer workstations on a computer network.

Help files

You may want to refer to the following Help systems:

- Financial Analyzer Help — Provides online procedural and reference information for both system administrators and users.
- Express Language Help — Describes the syntax and usage of Express commands, functions, and options.

Conventions

Text conventions

You will find the following text conventions in this document.

Convention	Usage
Boldface text	Indicates menu items, command buttons, options, field names, and hyperlinks. Bold text is also used for notes and other secondary information in tables (for example, **Result**).
`Fixed-width` text	Indicates folder names, file names, operating system commands, and URLs. Also indicates examples and anything that you must type exactly as it appears. **For example:** If you are asked to type `show eversion`, you would type all the characters exactly as shown in the fixed-width font.
Italic text	Indicates variables, including variable text. Variable text is used when dialog boxes or their components are unlabeled or have labels that change dynamically based on their current context. The wording of variable text does not exactly match what you see on your screen. Italic type is also used for emphasis, for new terms, and for titles of documents.
UPPERCASE text	Indicates Express commands and objects and acronyms.

Mouse usage

Always use the left mouse button unless you are specifically instructed to use the right mouse button.

The term "left mouse button" refers to the dominant button. If you have reconfigured your mouse to reverse the functions of the left and right buttons, then you will need to use the reverse button when you follow the procedures in this manual.

Formats for key combinations and sequences

Key combinations and key sequences appear in the following formats.

IF you see the format . . .	THEN . . .
Key1+Key2,	press and hold down the first key while you press the second key. **For example:** "Press Alt+Tab" means to press and hold down the Alt key while you press the Tab key.
Key1, Key2,	press and release the keys one after the other. **For example:** "Press Alt, F, O" means to press and release the Alt key, press and release the F key, then press and release the O key.

1

Structure and Meta Data Maintenance Using the Application Programming Interface

In This Chapter

Chapter summary

This chapter provides an overview of maintaining Oracle® Financial Analyzer (hereinafter referred to as “Financial Analyzer”) structure and metadata through the application programming interface (API).

List of topics

This chapter includes the following topics:

- Overview
- Environments
- Available programs

Overview

Using programs to maintain structure and meta data

You can maintain Financial Analyzer structure and the associated meta data through the normal Financial Analyzer user interface. However, you can also accomplish these maintenance tasks through the API.

This guide describes a set of programs that allow you to easily create and modify Financial Analyzer objects (dimensions, dimension values, financial data items, hierarchies, attributes, and models) through the API. Using these programs, you can maintain a Financial Analyzer database through code that you write, rather than through the use of normal user interface.

There are specific programs related to each type of object. The list of arguments for a call to a given program generally corresponds to the sequence of fields in the dialog box for creating or modifying a particular type of object, normally accessed through the Financial Analyzer user interface.

Objects and supporting meta data

All Financial Analyzer objects have a supporting range of meta data. This meta data is in the form of additional objects and special variables (catalogs) that serve as a grouping mechanism and hold data that is common to objects of a particular type. The call to the application programming interface (API) creates the object and, when necessary, the supporting meta data. The call also registers the name of the object, together with the relevant attributes, in the corresponding catalog.

Usage note: TIME and USER dimensions

The TIME dimension and the USER dimension are system dimensions. You cannot maintain these dimensions through the API.

Environments

Introduction

The calls to the API can create Financial Analyzer objects in two different environments, the Financial Analyzer Shared Database (OFAS.DB) and the personal database of the administrator.

Financial Analyzer shared database

The procedure for creating objects directly in the shared database involves the use of a custom program and a custom task. You must first create a custom program that uses the Financial Analyzer API, and then create a custom task that runs the custom program. See Appendix A for further information about custom tasks.

For objects created in the shared database, you must call the OFA_API_INIT program to trigger the automatic distribution of the new objects to the administrator. Each call to OFA_API_INIT defines everything that follows as a single set of objects that are ready for distribution and sets a property called SOURCE for each object. OFA_API_INIT must be the first call in the custom program because it keeps the pending distribution open until the end of the current task.

The first time that the administrator runs the Administrator workstation after the API calls have been run in the shared database, these objects are refreshed in the administrator's personal database. At that point, the administrator can distribute the objects to the appropriate users.

Administrator's personal database

If the calls to the API are made from the administrator's personal database, every new object is automatically ready for distribution. In this case, there is no need to call OFA_API_INIT.

You can call the API from an open Financial Analyzer session or you can make the calls externally, attaching the personal database with a SNAPI connection within the Express Connection Utility or within a custom application. If the personal database is attached within a custom application, the application code must attach certain databases before the call is made. These databases must be attached in the order in which they appear in the following list (the list also indicates the attachment mode for each database):

```
DBA personal database (R/W)
OFASERVE (R/O)
OFATOOLS (R/O)
Financial Analyzer language database (R/O)
XPDDDATA (R/O)
XPDDCODE (R/O)
XPADMIN (R/O)
```

Note: The name of the Financial Analyzer language database depends on the language being used. The format for the name is `OFA`*`xxx`*, where xxx is a three-character symbol that represents a specific language.

Available programs

Introduction

The programs fall into several categories, based on the type of object to which they are related.

List of programs

The following table lists the various categories of programs and the specific programs within each category.

Category	Programs
Initialization	OFA_API_INIT — Initialize Distribution
Dimension maintenance	OFA_CREATE_DIM — Create Dimension OFA_DEL_DIM — Delete Dimension OFA_MODIFY_DIM — Modify Dimension
Dimension value maintenance	OFA_CREATE_DMV — Create Dimension Value OFA_DEL_DMV — Delete Dimension Value OFA_MODIFY_DMV — Modify Dimension Value
Financial data item maintenance	OFA_CREATE_FDI — Create Financial Data Item OFA_DEL_FDI — Delete Financial Data Item OFA_MODIFY_FDI — Modify Financial Data Item
Hierarchy maintenance	OFA_CREATE_HIER — Create Hierarchy OFA_SET_HIER — Set Values for Hierarchy Population OFA_DEL_HIER — Delete Hierarchy OFA_MODIFY_HIER — Modify Hierarchy Description
Attribute maintenance	OFA_CREATE_ATTR — Create Attribute OFA_SET_ATTRS — Set Relations for Attribute OFA_CLEAR_ATTRS — Clear Relations for Attribute OFA_DEL_ATTR — Delete Attribute OFA_MODIFY_ATTR — Modify Attribute Description
Model maintenance	OFA_CREATE_MODEL — Create Model OFA_SET_EQUATION — Set Equation for Model OFA_DEL_MODEL — Delete Model OFA_MODIFY_MODEL — Modify Model Description

Program descriptions

Chapter 2 describes the programs listed above, grouping them by category. The description for each program includes an explanation of the program, the syntax for calling the program, and a table that lists the arguments for the call. At the end of each topic, there is a set of examples that provides sample calls for programs in the related category, as appropriate.

2 Programs for Maintaining Structure and Meta Data

In This Chapter

Chapter summary

This chapter describes the programs in each maintenance category that you can use to maintain Financial Analyzer structure and meta data through the application programming interface (API).

List of topics

This chapter includes the following topics:

- Initialization
- Dimension Maintenance
- Dimension Value Maintenance
- Financial Data Item Maintenance
- Hierarchy Maintenance
- Attribute Maintenance
- Model Maintenance

Initialization

Introduction

Every user object defined in Financial Analyzer has a property called SOURCE. The system uses this property to initialize distribution mechanisms for objects.

OFA_API_INIT — Initialize Distribution

Use the OFA_API_INIT program to define a new source before you create any objects in the shared database. After you call OFA_API_INIT, all API-defined objects are automatically included in the distribution list for the administrator, using the new source as the owner of the objects.

The syntax for the OFA_API_INIT program is as follows:

```
ofa_api_init(source)
```

The following table describes the argument for the OFA_API_INIT program.

Argument	Description
source	The name of the external source of the items to be created.

Dimension Maintenance

OFA_CREATE_DIM — Create Dimension

Use the OFA_CREATE_DIM program to create a dimension and its associated meta data.

The syntax for the OFA_CREATE_DIM program is as follows:

```
ofa_create_dim(dimension, description, width, prefix, is_time, agg_allowed,
variance_allowed, scaleable, dba_sort, hierarchy_creator, relation_creator,
model_creator)
```

The following table describes the arguments for the OFA_CREATE_DIM program.

Argument	Description
dimension	The name of the dimension that you want to create
description	A text description for the dimension (used as the Selector description)
width	The maximum width of a dimension value in the dimension.
prefix	The prefix to be used in defining supporting object names. The prefix must not exceed six characters.
is_time	Indicates whether the dimension is a time dimension. Specify YES or NO.
agg_allowed	Indicates whether aggregation types are allowed for the dimension. Specify YES or NO.
variance_allowed	Indicates whether a B/W variance indicator can be set for the dimension. Specify YES or NO.
scaleable	Indicates whether scaling can be controlled at the dimension value level for the dimension (by default, all dimension values are scaleable). Specify YES or NO.
dba_sort	Indicates whether the values for the dimension are to be refreshed in each user's database according to the order specified in the administrator's database.
hierarchy_creator	Indicates who can define hierarchies for the dimension. Specify EVERYONE or NOONE.
relation_creator	Indicates who can define attributes for the dimension. Specify EVERYONE or NOONE.
model_creator	Indicates who can define models based on the dimension. Specify EVERYONE or NOONE.

Upon successful completion of the OFA_CREATE_DIM program, the system returns the name of the created dimension. If there is an error, the system returns the value NA; you can find out what the actual error is through the use of the ERRORNAME option. For information about using the ERRORNAME option, refer to the Express Language Help system.

OFA_DEL_DIM — Delete Dimension

Use the OFA_DEL_DIM program to delete a dimension and its associated meta data.

The OFA_DEL_DIM program deletes the specified dimension and prepares the system for the subsequent distribution of the deletion by updating the list of items ready for distribution. The DBA can then specify the **Delete from System** action through the normal Financial Analyzer user interface.

Note: The OFA_DEL_DIM program is valid only when issued in the personal database of an administrator workstation.

The syntax for the OFA_DEL_DIM program is as follows:

```
ofa_del_dim(dimension)
```

The following table describes the argument for the OFA_DEL_DIM program.

Argument	Description
dimension	The name of the dimension that you want to delete.

Upon successful completion of the OFA_DEL_DIM program, the system returns the name of the deleted dimension. If there is an error, the system returns the value `NA`; you can find out what the actual error is through the use of the ERRORNAME option. For information about using the ERRORNAME option, refer to the Express Language Help system.

OFA_MODIFY_DIM — Modify Dimension

Use the OFA_MODIFY_DIM program to modify an existing dimension and its associated meta data.

The syntax for the OFA_MODIFY_DIM program is as follows:

```
ofa_modify_dim(dimension, keyword_1, keyword_1_value, keyword_2,
keyword_2_value, ... keyword_n, keyword_n_value)
```

The following table describes the arguments for the OFA_MODIFY_DIM program.

Argument	Description
dimension	The name of the dimension that you want to modify.
keyword_1, *keyword_1_value* through *keyword_n*, *keyword_n_value*	The keywords and associated values that you specify. Specify at least one keyword. You can specify as many of the keywords as you choose. See the following keyword descriptions for further information.

There are several keywords that you can specify for the OFA_MODIFY_DIM program. These keywords pertain to the dimension attributes that the API can modify.

You can specify any number of keywords in any order. For each keyword that you specify, you must also specify a keyword value following the keyword. For any keywords that you do not specify, the corresponding attributes for those keywords remain unchanged. The following table lists the keywords and the valid keyword values for the OFA_MODIFY_DIM program.

Keyword	Description	Values
TIMEAGG.USED	Indicates whether the dimension supports the aggregation of financial data over time.	`YES` `NO`
VARTYPE.USED	Indicates the dimension type.	`TEXT` `TIME`
SORT.OK	Indicates whether the dimension values are to be refreshed in each user's database according to the order specified in the administrator's database.	`YES` `NO`
SCALING.USED	Indicates whether numeric scaling can be controlled for each value of the dimension.	`YES` `NO`
HIERARCHY.OK	Indicates who will be allowed to create new hierarchies.	`EVERYONE` `NOONE`
MODEL.OK	Indicates who will be allowed to create new models.	`EVERYONE` `NOONE`
RELATION.OK	Indicates who will be allowed to create new attributes.	`EVERYONE` `NOONE`
DESCRIPTION	A text description	Any text value

Upon successful completion of the OFA_MODIFY_DIM program, the system returns the name of the modified dimension. If there is an error, the system returns the value NA; you can find out what the actual error is through the use of the ERRORNAME option. For information about using the ERRORNAME option, refer to the Express Language Help system.

Dimension maintenance examples

The following example shows a sample OFA_CREATE_DIM call that creates a dimension named TESTDIM1.

```
retval = ofa_create_dim('TESTDIM1', 'Test dim #1', 10, 'TD1', NO, NO, NO, NO,
YES, 'EVERYONE', 'EVERYONE', 'EVERYONE')
```

The following example shows a sample OFA_DEL_DIM call that deletes the dimension TESTDIM1.

```
retval = ofa_del_dim('TESTDIM1')
```

The following example shows a sample OFA_MODIFY_DIM call that modifies some of the attributes of the dimension TESTDIM1.

```
retval = ofa_modify_dim('TESTDIM1', 'SORT.OK', 'NO', 'HIERARCHY.OK', 'NOONE',
'MODEL.OK', 'NOONE', 'RELATION.OK', 'NOONE', 'DESCRIPTION', 'Modified
Dimension')
```

Dimension Value Maintenance

OFA_CREATE_DMV — Create Dimension Value

Use the OFA_CREATE_DMV program to create a dimension value and its associated meta data.

The syntax for the OFA_CREATE_DMV program is as follows:

```
ofa_create_dmv(dimension, dimension value, description, column_label, row_label,
aggregate, variance, scale, period)
```

The following table describes the arguments for the OFA_CREATE_DMV program.

Argument	Description
dimension	The name of the dimension in which you want to create the dimension value
dimension_value	The name of the dimension value that you want to create

Argument	Description
description	A text description for the dimension value (used as the Selector description)
column_label	The text description used when the dimension value is displayed in a column
row_label	The text description used when the dimension value is displayed in a row

aggregate	Indicates the aggregation type for the dimension value. Specify one of the following values:
	
	`AVG`
	`AVGC`
	`BEG`
	`CALC`
	`END`
	`SUM`
	
	Note: This argument is valid only for dimensions that support aggregation types. If the dimension does not support aggregation types, specify the position of the argument by coding an empty set of single quotation marks ('').

variance	Indicates the type of variance indicator for the dimension value. Specify `BUD-ACT` or `ACT-BUD`. The default is `ACT-BUD`. **Note:** This argument is valid only for dimensions that support variance indicators. If the dimension does not support variance indicators, specify the position of the argument by coding an empty set of single quotation marks ('').

Argument	Description
scale	Indicates whether the dimension value is scaleable. Specify `YES` or `NO`. **Note:** This argument is valid only for dimensions that support scaling by value. If the dimension does not support scaling by value, specify the position of the argument by coding an empty set of single quotation marks (`''`).

period	Indicates what time period the dimension value represents. Specify one of the following values:
	
	`MONTH`
	`QUARTER`
	`YEAR`
	`YTD`
	
	Note: This argument is valid only for dimensions that are custom time dimensions. If the dimension is not a custom time dimension, specify the position of the argument by coding an empty set of single quotation marks (`''`).

Upon successful completion of the OFA_CREATE_DMV program, the system returns the name of the created dimension value. If there is an error, the system returns the value `NA`; you can find out what the actual error is through the use of the ERRORNAME option. For information about using the ERRORNAME option, refer to the Express Language Help system.

OFA_DEL_DMV — Delete Dimension Value

Use the OFA_DEL_DMV program to delete a dimension value and its associated meta data.

The OFA_DEL_DMV program deletes the specified dimension value and prepares the system for the subsequent distribution of the deletion by updating the list of items ready for distribution. The DBA can then specify the **Delete from System** action through the normal Financial Analyzer user interface.

Note: The OFA_DEL_DMV program is valid only when issued in the personal database of an administrator workstation.

The syntax for the OFA_DEL_DMV program is as follows:

```
ofa_del_dmv(dimension, dimension_value)
```

The following table describes the arguments for the OFA_DEL_DMV program.

Argument	Description
dimension	The dimension from which the dimension value is to be deleted
dimension_value	The name of the dimension value that you want to delete

Upon successful completion of the OFA_DEL_DMV program, the system returns the name of the deleted dimension value. If there is an error, the system returns the value `NA`; you can find out what the actual error is through the use of the ERRORNAME option. For information about using the ERRORNAME option, refer to the Express Language Help system.

OFA_MODIFY_DMV — Modify Dimension Value

Use the OFA_MODIFY_DMV program to modify an existing dimension value and its associated meta data.

The syntax for the OFA_MODIFY_DMV program is as follows:

```
ofa_modify_dmv(dimension, dimension_value, keyword_1, keyword_1_value,
keyword_2, keyword_2_value, ... keyword_n, keyword_n_value)
```

The following table describes the arguments for the OFA_MODIFY_DMV program.

Argument	Description
dimension	The name of the dimension that contains the dimension value that you want to modify
dimension_value	The name of the dimension value that you want to modify
keyword_1, *keyword_1_value* through *keyword_n*, *keyword_n_value*	The keywords and associated values that you specify. Specify at least one keyword. You can specify as many of the keywords as you choose. See the following keyword descriptions for further information.

There are several keywords that you can specify for the OFA_MODIFY_DMV program. These keywords pertain to the dimension value attributes that the API can modify.

You can specify any number of keywords in any order. For each keyword that you specify, you must also specify a keyword value following the keyword. For any keywords that you do not specify, the corresponding attributes for those keywords remain unchanged. The following table lists the keywords and the valid keyword values for the OFA_MODIFY_DMV program.

Keyword	Description	Values

TIME.AGG	Indicates the aggregation type for the dimension value.
	
	Note: This argument is valid only for dimensions that support aggregation types. If the dimension does not support aggregation types, specify the position of the argument by coding an empty set of single quotation marks ('').
		`AVG`
		`AVGC`
		`BEG`
		`CALC`
		`END`
		`SUM`

VARTYPE	Indicates the type of variance.	`BUD-ACT` `ACT-BUD`
SCALE	Indicates whether numeric scaling can be controlled for each value of the dimension.	`YES` `NO`
ROW.LABEL	The text description used when the dimension value is displayed in a row	Any text value
COL.LABEL	The text description used when the dimension value is displayed in a column	Any text value

PERIOD	Applies to time dimensions; indicates what period the dimension value represents.	`MONTH`
		`QUARTER`
		`YEAR`
		`YTD`

DESCRIPTION	A text description	Any text value

Upon successful completion of the OFA_MODIFY_DMV program, the system returns the name of the modified dimension value. If there is an error, the system returns the value `NA`; you can find out what the actual error is through the use of the ERRORNAME option. For information about using the ERRORNAME option, refer to the Express Language Help system.

Dimension value maintenance examples

The following example shows a set of sample OFA_CREATE_DMV calls that create dimension values in the TESTDIM1 dimension.

```
retval = ofa_create_dmv('TESTDIM1', 'TD1val1', 'TD1 Value #1', 'Val1Col',
'Val1Row', 'SUM', 'ACT-BUD', 'YES', '')

retval = ofa_create_dmv('TESTDIM1', 'TD1val2', 'TD1 Value #2', 'Val2Col',
'Val2Row', 'BEG', 'ACT-BUD', 'YES', '')

retval = ofa_create_dmv('TESTDIM1', 'TD1val3', 'TD1 Value #3', 'Val3Col',
'Val3Row', 'END', 'ACT-BUD', 'YES', '')

retval = ofa_create_dmv('TESTDIM1', 'TD1val4', 'TD1 Value #4', 'Val4Col',
'Val4Row', 'AVG', 'BUD-ACT', 'NO', '')

retval = ofa_create_dmv('TESTDIM1', 'TD1val5', 'TD1 Value #5', 'Val5Col',
'Val5Row', 'AVGC', 'BUD-ACT', 'NO', '')

retval = ofa_create_dmv('TESTDIM1', 'TD1val6', 'TD1 Value #6', 'Val6Col',
'Val6Row', 'CALC', 'BUD-ACT', 'NO', '')
```

The following example shows a sample OFA_DEL_DMV call that deletes a dimension value from the TESTDIM1 dimension.

```
retval = ofa_del_dmv('TESTDIM1', 'TD1VAL4')
```

The following example shows a sample OFA_MODIFY_DMV call that modifies a dimension value in the TESTDIM1 dimension.

```
retval = ofa_modify_dmv('TESTDIM1', 'TD1VAL4', 'ROW.LABEL', 'New Row Label',
'COL.LABEL', 'New Col Label', 'VARTYPE', 'ACT-BUD', 'TIME.AGG', 'SUM', 'SCALE',
'YES', 'DESCRIPTION', 'TD1VAL4 - Modified Name')
```

Financial Data Item Maintenance

OFA_CREATE_FDI — Create Financial Data Item

Use the OFA_CREATE_FDI program to create an Financial Analyzer financial data item. If the financial data item is a formula financial data item, the program evaluates the validity of the Express syntax for the formula.

The syntax for the OFA_CREATE_FDI program is as follows:

```
ofa_create_fdi(fdi_name, description, stored, column_label, row_label, scale,
data_dist, dimensions, data_type, sparse, sparse_dims, formula)
```

The following table describes the arguments for the OFA_CREATE_FDI program.

Argument	Description
fdi_name	The name for the financial data item
description	A text description for the financial data item (used as the Selector description)
stored	Indicates whether this object is a stored financial data item, an automatic formula financial data item, or a manual formula financial data item. Specify one of the following values: ▪ `STORED` ▪ `AUTOMATIC` ▪ `MANUAL`
column_label	The text description used when the financial data item is displayed in a column
row_label	The text description used when the financial data item is displayed in a row
scale	Indicates whether the financial data item is scaleable. Specify `YES` or `NO.`
data_dist	Indicates whether data distribution option is relevant for the financial data item. Specify `YES` or `NO.` **Note:** This argument applies only to stored financial data items.
dimensions	A list of names of existing dimensions that are used to define this variable. The following example, which lists the dimensions DIM1, DIM2, and DIM3, shows the syntax for this list. `DIM1\nDIM2\nDIM3`

Argument	Description
data_type	Indicates the data type for the financial data item. Specify one of the following: ■ DECIMAL ■ INTEGER ■ SHORTDECIMAL ■ SHORTINTEGER ■ TEXT
sparse	Indicates whether the financial data item uses a composite dimension. Specify YES or NO. **Note:** This argument applies only to stored financial data items.
sparse_dims	A list of all of the dimensions for this financial data item, specifying the sparse dimensions by enclosing them in brackets (< >). For example, the following sample value for sparse_dims lists the dimensions for a sample financial data item and indicates that DIM2 and DIM3 are the sparse dimensions. `DIM1\n<DIM2 DIM3>\nDIM4` **Note:** This argument applies only to stored financial data items.
formula	A text string representing the formula expression

Upon successful completion of the OFA_CREATE_FDI program, the system returns the name of the created financial data item. If there is an error, the system returns the value NA; you can find out what the actual error is through the use of the ERRORNAME option. For information about using the ERRORNAME option, refer to the Express Language Help system.

OFA_DEL_FDI — Delete Financial Data Item

Use the OFA_DEL_FDI program to delete a financial data item and its associated meta data.

The OFA_DEL_FDI program deletes the specified financial data item and prepares the system for the subsequent distribution of the deletion by updating the list of items ready for distribution. The DBA can then specify the **Delete from System** action through the normal Financial Analyzer user interface.

Note: The OFA_DEL_FDI program is valid only when issued in the personal database of an administrator workstation.

The syntax for the OFA_DEL_FDI program is as follows:

```
ofa_del_fdi(fdi_name)
```

The following table describes the argument for the OFA_DEL_FDI program.

Argument	Description
fdi_name	The name of the financial data item that you want to delete

Upon successful completion of the OFA_DEL_FDI program, the system returns the name of the deleted financial data item. If there is an error, the system returns the value NA; you can find out what the actual error is through the use of the ERRORNAME option. For information about using the ERRORNAME option, refer to the Express Language Help system.

OFA_MODIFY_FDI — Modify Financial Data Item

Use the OFA_MODIFY_FDI program to modify an existing financial data item and its associated meta data.

The syntax for the OFA_MODIFY_FDI program is as follows:

```
ofa_modify_fdi(fdi_name, keyword_1, keyword_1_value, keyword_2, keyword_2_value,
... keyword_n, keyword_n_value)
```

The following table describes the arguments for the OFA_MODIFY_FDI program.

Argument	Description
fdi_name	The name of the financial data item that you want to modify
keyword_1, *keyword_1_value* through *keyword_n*, *keyword_n_value*	The keywords and associated values that you specify. Specify at least one keyword. You can specify as many of the keywords as you choose. See the following keyword descriptions for further information.

There are several keywords that you can specify for the OFA_MODIFY_FDI program. These keywords pertain to the financial data item attributes that the API can modify.

You can specify any number of keywords in any order. For each keyword that you specify, you must also specify a keyword value following the keyword. For any keywords that you do not specify, the corresponding attributes for those keywords remain unchanged. The following table lists the keywords and the valid keyword values for the OFA_MODIFY_FDI program.

Keyword	Description	Values
DESCRIPTION	A text description	Any text value
SCALE	Indicates whether numeric scaling can be controlled for each value of the dimension.	`YES` `NO`
ROW.LABEL	The text description used when the dimension value is reported in a row	Any text value
COL.LABEL	The text description used when the dimension value is reported in a column	Any text value
DATA.DIST.OK	Indicates whether distribution of the data is allowed	`YES` `NO`
SPARSE	Indicates whether the financial data item uses a composite dimension	`YES` `NO`
SP_DIMS	A list of all of the dimensions for this financial data item, specifying the sparse dimensions by enclosing them in brackets (< >). For example, the following sample value for sparse_dims lists the dimensions for a sample financial data item and indicates that DIM2 and DIM3 are the sparse dimensions. `DIM1\n<DIM2 DIM3>\nDIM4`	A dimension list (see Description column for explanation)

Upon successful completion of the OFA_MODIFY_FDI program, the system returns the name of the modified financial data item. If there is an error, the system returns the value `NA`; you can find out what the actual error is through the use of the ERRORNAME option. For information about using the ERRORNAME option, refer to the Express Language Help system.

Financial data item maintenance examples

The following example shows a sample OFA_CREATE_FDI call that creates a financial data item named EXTSTO.

```
retval = ofa_create_fdi('EXTSTO', 'Ext Stored', 'STORED', 'StoCol', 'StoRow',
'YES', 'YES', 'TESTDIM1\nTESTDUM2\nTESTDIM3\nTESTDIM4', 'DECIMAL', 'YES',
'TESTDIM1\n<TESTDIM2 TESTDIM3 TESTDIM4>', '')
```

The following example shows a sample OFA_DEL_FDI call that deletes a financial data item named EXTSTO.

```
retval = ofa_del_fdi('EXSTO')
```

Hierarchy Maintenance

OFA_CREATE_HIER — Create Hierarchy

Use the OFA_CREATE_HIER program to create a hierarchy and its associated meta data.

The syntax for the OFA_CREATE_HIER program is as follows:

```
ofa_create_hier(description, dimension)
```

The following table describes the arguments for the OFA_CREATE_HIER program.

Argument	Description
description	A text description for the hierarchy (used as the Selector description)
dimension	The name of the dimension in which you want to create the hierarchy

Upon successful completion of the OFA_CREATE_HIER program, the system returns the value of HI.ENTRY that corresponds to the created hierarchy. If there is an error, the system returns the value NA; you can find out what the actual error is through the use of the ERRORNAME option. For information about using the ERRORNAME option, refer to the Express Language Help system.

OFA_SET_HIER — Set Values for Hierarchy Population

After you have created a hierarchy, you need to populate it by specifying the parent-child relationships. Use the OFA_SET_HIER program to populate a hierarchy.

The syntax for the OFA_SET_HIER program is as follows:

```
ofa_set_hier(hierarchy, child, parent)
```

The following table describes the arguments for the OFA_SET_HIER program.

Argument	Description
hierarchy	The HI.ENTRY value that corresponds to the hierarchy
child	The dimension value that is the child of the dimension value specified in the parent argument
parent	The dimension value that is the parent of the dimension value specified in the child argument

Upon successful completion of the OFA_SET_HIER program, the system returns the HI.ENTRY value that corresponds to the hierarchy. If there is an error, the system returns the value NA; you can find out what the actual error is through the use of the ERRORNAME option. For information about using the ERRORNAME option, refer to the Express Language Help system.

The following example shows a set of sample OFA_SET_HIER calls that add three parent-child relationships to a hierarchy.

```
retval = ofa_set_hier('HI.AAS22926', 'TD1VAL2', 'TD1VAL1')

retval = ofa_set_hier('HI.AAS22926', 'TD1VAL3', 'TD1VAL1')

retval = ofa_set_hier('HI.AAS22926', 'TD1VAL4', 'TD1VAL1')
```

You must call OFA_SET_HIER for every individual parent-child relationship that you define.

After the last call to OFA_SET_HIER, and before the actual usage of the hierarchy in a working environment, you must call the CM.UPDATE.HIER program to recalculate the supporting structures for the hierarchy. When you call CM.UPDATE.HIER, you supply the internal name of the hierarchy as the argument.

The following example shows a sample CM.UPDATE.HIER call where HI.AA34325 is the internal name of the hierarchy.

```
call cm.update.hier('HI.AA34325')
```

For further information about using CM.UPDATE.HIER, see Appendix B.

OFA_DEL_HIER — Delete Hierarchy

Use the OFA_DEL_HIER program to delete a hierarchy and its associated meta data.

The OFA_DEL_HIER program deletes the specified hierarchy and prepares the system for the subsequent distribution of the deletion by updating the list of items ready for distribution. The administrator can then specify the **Delete from System** action through the normal Financial Analyzer user interface.

Note: The OFA_DEL_HIER program is valid only when issued in the personal database of an administrator workstation.

The syntax for the OFA_DEL_HIER program is as follows:

```
ofa_del_hier(hierarchy)
```

The following table describes the argument for the OFA_DEL_HIER program.

Argument	Description
hierarchy	The HI.ENTRY value that corresponds to the hierarchy

Upon successful completion of the OFA_DEL_HIER program, the system returns the HI.ENTRY value that corresponds to the deleted hierarchy. If there is an error, the system returns the value NA; you can find out what the actual error is through the use of the ERRORNAME option. For information about using the ERRORNAME option, refer to the Express Language Help system.

OFA_MODIFY_HIER — Modify Hierarchy Description

Use the OFA_MODIFY_HIER program to modify the description for an existing hierarchy.

The syntax for the OFA_MODIFY_HIER program is as follows:

```
ofa_modify_hier(hierarchy, DESCRIPTION, desc_value)
```

The following table describes the arguments for the OFA_MODIFY_HIER program.

Argument	Description
hierarchy	The HI.ENTRY value that corresponds to the hierarchy
DESCRIPTION	The keyword `DESCRIPTION`.
desc_value	A text description for the hierarchy

Upon successful completion of the OFA_MODIFY_HIER program, the system returns the HI.ENTRY value that corresponds to the modified hierarchy. If there is an error, the system returns the value NA; you can find out what the actual error is through the use of the ERRORNAME option. For information about using the ERRORNAME option, refer to the Express Language Help system.

Hierarchy maintenance examples

The following example shows a sample OFA_CREATE_HIER call that creates a hierarchy in the TESTDIM1 dimension.

```
retval = ofa_create_hier('TESTDIM1 Hierarchy', 'TESTDIM1')
```

The following example shows a sample OFA_DEL_HIER call that deletes a hierarchy.

```
retval = ofa_del_hier('HI.AAS22926')
```

Attribute Maintenance

OFA_CREATE_ATTR — Create Attribute

Use the OFA_CREATE_ATTR program to create an attribute and its associated meta data.

The syntax for the OFA_CREATE_ATTR program is as follows:

```
ofa_create_attr(description, base_dimension, attribute_type, agg_dimension)
```

The following table describes the arguments for the OFA_CREATE_ATTR program.

Argument	Description
description	A text description for the attribute (used as the Selector description)
base_dimension	An existing dimension name used to define this attribute. This specifies the dimension to which one or more 'dimensions' are related.
attribute_type	The attribute type. Specify `ONE` (one to many) or `MANY` (many to many).
agg_dimension	The name of the aggregate dimension of the attribute

Upon successful completion of the OFA_CREATE_ATTR program, the system returns the value of RL.ENTRY that corresponds to the created attribute. If there is an error, the system returns the value `NA`; you can find out what the actual error is through the use of the ERRORNAME option. For information about using the ERRORNAME option, refer to the Express Language Help system.

OFA_SET_ATTRS — Set Relations for Attribute

Use the OFA_SET_ATTRS program to define the Express relations for an attribute.

The syntax for the OFA_SET_ATTRS program is as follows:

```
ofa_set_attrs(relation_name, relation_pairs)
```

The following table describes the arguments for the OFA_SET_ATTRS program.

Argument	Description
relation_name	The RL.ENTRY value that corresponds to the attribute
relation_pairs	A multi-line text string that specifies the Express relation pairs between the base dimension and the aggregate dimension. Each pair of lines describes a relation between a value of the base dimension (the first line of the pair) and a value of the aggregate dimension (the second line of the pair). For example, the following sample relation_pairs value specifies TD2VAL1 as an attribute of base dimension TD1VAL1 and specifies TD2VAL4 as an attribute of base dimension TD1VAL3. TD1VAL1\nTD2VAL1\nTD1VAL3\nTD2VAL4

Upon successful completion of the OFA_SET_ATTRS program, the system returns the value of RL.ENRTY that corresponds to the populated attribute. If there is an error, the system returns the value NA; you can find out what the actual error is through the use of the ERRORNAME option. For information about using the ERRORNAME option, refer to the Express Language Help system.

OFA_CLEAR_ATTRS — Clear Relations for Attribute

Use the OFA_CLEAR_ATTRS program to clear the Express relations that have been previously set for an attribute.

The syntax for the OFA_CLEAR_ATTRS program is as follows:

```
ofa_clear_attrs(relation_name, relation_pairs)
```

The following table describes the arguments for the OFA_CLEAR_ATTRS program.

Argument	Description
relation_name	The RL.ENTRY value that corresponds to the attribute
relation_pairs	A multi-line text string that specifies the Express relation pairs between the base dimension and the aggregate dimension for the attribute that is to be cleared. Each pair of lines describes a relation between a value of the base dimension (the first line of the pair) and a value of the aggregate dimension (the second line of the pair). For example, the following sample relation_pairs value specifies TD2VAL1 as an attribute of base dimension TD1VAL1 and specifies TD2VAL4 as an attribute of base dimension TD1VAL3. `TD1VAL1\nTD2VAL1\nTD1VAL3\nTD2VAL4`

Upon successful completion of the OFA_CLEAR_ATTRS program, the system returns the value of RL.ENTRY that corresponds to the cleared attribute. If there is an error, the system returns the value NA; you can find out what the actual error is through the use of the ERRORNAME option. For information about using the ERRORNAME option, refer to the Express Language Help system.

OFA_DEL_ATTR — Delete Attribute

Use the OFA_DEL_ATTR program to delete an attribute and its associated meta data.

The OFA_DEL_ATTR program deletes the specified attribute and prepares the system for the subsequent distribution of the deletion by updating the list of items ready for distribution. The DBA can then specify the **Delete from System** action through the normal Financial Analyzer user interface.

Note: The OFA_DEL_ATTR program is valid only when issued in the personal database of an administrator workstation.

The syntax for the OFA_DEL_ATTR program is as follows:

```
ofa_del_attr(attribute)
```

The following table describes the argument for the OFA_DEL_ATTR program.

Argument	Description
attribute	The RL.ENTRY value that corresponds to the attribute

Upon successful completion of the OFA_DEL_ATTR program, the system returns the value of RL.ENTRY that corresponds to the deleted attribute. If there is an error, the system returns the value NA; you can find out what the actual error is through the use of the ERRORNAME option. For information about using the ERRORNAME option, refer to the Express Language Help system.

OFA_MODIFY_ATTR — Modify Attribute Description

Use the OFA_MODIFY_ATTR program to modify the description for an existing attribute.

The syntax for the OFA_MODIFY_ATTR program is as follows:

```
ofa_modify_attr(attribute, DESCRIPTION, desc_name)
```

The following table describes the argument for the OFA_MODIFY_ATTR program.

Argument	Description
attribute	The RL.ENTRY value that corresponds to the attribute
DESCRIPTION	The keyword DESCRIPTION
desc_name	A text description for the attribute

Upon successful completion of the OFA_MODIFY_ATTR program, the system returns the value of RL.ENTRY that corresponds to the modified attribute. If there is an error, the system returns the value NA; you can find out what the actual error is through the use of the ERRORNAME option. For information about using the ERRORNAME option, refer to the Express Language Help system.

Attribute maintenance examples

The following example shows a sample OFA_CREATE_ATTR call that creates a an attribute named “One to Many Attribute” between the dimensions TESTDIM1 and TESTDIM2.

```
retval = ofa_create_attr('One to Many Attribute', 'TESTDIM1', 'ONE', TESTDIM2'))
```

The following example shows a sample OFA_SET_ATTRS call that populates a relation. In this example, RL.AAS22926 is an attribute between the TESTDIM1 and TESTDIM2 dimensions. The call sets TD2VAL1 as an attribute of TD1VAL1 and sets TD2VAL4 as an attribute of TD1VAL3.

```
retval = ofa_set_attrs('RL.AAS22926', 'TD1VAL1\nTD2VAL1\nTD1VAL3\nTD2VAL4')
```

The following example shows a sample OFA_DEL_ATTR call that deletes the attribute RL.AAS22926.

```
retval = ofa_del_attr('RL.AAS22926')
```

Model Maintenance

OFA_CREATE_MODEL — Create Model

Use the OFA_CREATE_MODEL program to create a model and its associated meta data.

The syntax for the OFA_CREATE_MODEL program is as follows:

```
ofa_create_model(description, base_dimension, time_dimension, variables)
```

The following table describes the arguments for the OFA_CREATE_MODEL program.

Argument	Description
description	A text description for the model (used as the Selector description)
base_dimension	An existing dimension whose values are to be used in the model calculation.

Argument	Description
time_dimension	The dimension that represents the time dimension over which you want to calculate data. You do not have to specify a value for this argument. If you do not specify a value, you must still specify the position of the argument by coding an empty set of single quotation marks (`''`). Note that if you do not specify a value for this argument, you will not be able to build time series calculations such as LAG and LEAD.
variables	The names of any financial data items that you want to assign to the model. Use multiple lines of text for multiple financial data items.

Upon successful completion of the OFA_CREATE_MODEL program, the system returns the value of MD.ENTRY that corresponds to the created model. If there is an error, the system returns the value `NA`; you can find out what the actual error is through the use of the ERRORNAME option. For information about using the ERRORNAME option, refer to the Express Language Help system.

OFA_SET_EQUATION — Set Equation for Model

After creating the model meta data in the model catalog, you need to populate the equation variable with the actual equations. Use the OFA_SET_EQUATION program to load the equations into the equation variable and compile the model. You must call OFA_SET_EQUATION for each equation.

The syntax for the OFA_SET_EQUATION program is as follows:

```
ofa_set_equation(model, equation)
```

The following table describes the arguments for the OFA_SET_EQUATION program.

Argument	Description
model	The MD.ENTRY value that corresponds to the model
equations	The equations to be set into the equation variable. Specify a text string (this can be a multi-line string) that represents the expression for the equation (for example, TD2VAL1=TD2VAL3 * TD2VAL4.

Upon successful completion of the OFA_SET_EQUATION program, the system returns the value of MD.ENTRY that corresponds to the model. If there is an error, the system returns the value `NA`; you can find out what the actual error is through the use of the ERRORNAME option. For information about using the ERRORNAME option, refer to the Express Language Help system.

OFA_DEL_MODEL — Delete Model

Use the OFA_DEL_MODEL program to delete a model and its associated meta data.

The OFA_DEL_MODEL program deletes the specified model and prepares the system for the subsequent distribution of the deletion by updating the list of items ready for distribution. The DBA can then specify the **Delete from System** action through the normal Financial Analyzer user interface.

Note: The OFA_DEL_MODEL program is valid only when issued in the personal database of an administrator workstation.

The syntax for the OFA_DEL_MODEL program is as follows:

```
ofa_del_model(model)
```

The following table describes the argument for the OFA_DEL_MODEL program.

Argument	Description
model	The MD.ENTRY value that corresponds to the model

Upon successful completion of the OFA_DEL_MODEL program, the system returns the value of MD.ENTRY that corresponds to the deleted model. If there is an error, the system returns the value NA; you can find out what the actual error is through the use of the ERRORNAME option. For information about using the ERRORNAME option, refer to the Express Language Help system.

OFA_MODIFY_MODEL — Modify Model Description

Use the OFA_MODIFY_MODEL program to modify the description for an existing model.

The syntax for the OFA_MODIFY_MODEL program is as follows:

```
ofa_modify_model(model, description)
```

The following table describes the argument for the OFA_MODIFY_MODEL program.

Argument	Description
model	The MD.ENTRY value that corresponds to the model
DESCRIPTION	The keyword `DESCRIPTION`
description	A text description for the model
VARIABLES	The keyword `VARIABLES`
var_list	The names of any financial data items that you want to assign to the model. Use multiple lines of text for multiple financial data items.

Upon successful completion of the OFA_MODIFY_MODEL program, the system returns the value of MD.ENTRY that corresponds to the modified model. If there is an error, the system returns the value `NA`; you can find out what the actual error is through the use of the ERRORNAME option. For information about using the ERRORNAME option, refer to the Express Language Help system.

Model maintenance examples

The following example shows a sample OFA_CREATE_MODEL call that creates a an model along the dimension TESTDIM2 to solve the financial data item named EXTSTO.

```
retval = ofa_create_model('TESTDIM2 Model', 'TESTDIM2', '', 'EXTSTO')
```

The following example shows a sample OFA_SET_EQUATION call that sets an equation for the TESTDIM2 Model, where MD.AAS22926 is the MD.ENTRY value associated with the model.

```
retval = ofa_set_equation('MD.AAS22926', 'TD2VAL1=TD2VAL3 * TD2VAL4')
```

The following example shows a sample OFA_DEL_MODEL call that deletes the model MD.AAS22926.

```
retval = ofa_del_model('MD.AAS22926')
```

A

Creating Custom Tasks for the Task Processor

Overview

Working with custom tasks

Some extensions may need to write to the shared database. This appendix explains how to use the Task Processor to process custom tasks. There are three basic steps to setting up tasks to run from the Task Processor.

Running tasks from the Task Processor

The following table describes the steps for setting up tasks to run from the Task Processor.

Step	Action
1	Add the custom task information into the OFASTASK database.
2	Create a program to submit the task.
3	Create a program to process the task in the Task Processor.

The following topics describe each of these steps in detail.

Adding the custom task information to the OFASTASK database

Procedure: Adding the custom task information to the OFASTASK database

The following procedure describes how to add custom task information to the OFASTASK Database.

Step	Action
1	Choose a name for the new task. The new name will become an entry in a catalog and must be in all capital letters. For example: ACTUALS READER
2	Attach the OFASTASK database.
3	Update TKS.CATALOG.

Example: Adding the custom task information to the OFASTASK database

In this example, the task “ACTUALS READER” is added to the database by entering the commands that are described in the following table.

Step	Action	Description
1	Enter the following command: `->mnt tks.entry add ’ACTUALS READER’`	This is the name of the task entry.
2	Enter the following command: `->limit tks.entry to ’ACTUALS READER’`	Limit tks.entry to the name of the task.
3	Enter the following command: `->tks.catalog(tks.prop ’PRG.TYPE’) = ’CUSTOM’`	This indicates the task is your custom task.
4	Enter the following command: `->tks.catalog(tks.prop ’PRG.MANAGER’) = ’ACTREADER’`	This is the name of the program that is processed when the Task Processor is run.

Creating a program to submit the task

Procedure: Creating a program to submit the task

To submit a custom task to the task processor, you must define a program that will submit the custom task. The following procedure describes how to define a task.

Step	Action
1	Use the function tk.allocate to allocate a task. For example: `_prefix = tk.allocate('ACTUALS READER', NA)` where: ■ The first argument (text) is the name of the task (tks.entry) ■ The second argument (text) is used only when a lower level DBA allocates a task in his super DBA task queue (should be 'SUPER' if this the case) ■ The return value (_prefix) is the unique prefix that is used to build file names for the task
2	Place all data and control information needed to submit the task to the task processor into a continuous series of files. Assign the files the following names: `joinchars(_prefix '.1'), joinchars(_prefix '.2'), ..`
3	Call the program tk.submit to submit a task. For example: `call tk.submit(2, 'YES',NA)` where: ■ The first argument (integer) is the number of files in the task (could be 0) ■ The second argument (text) determines if the task can be restarted if it is interrupted (if, for example, the server goes down during the execution of the task). YES specifies the task can be restarted, NO specifies it cannot be restarted. ■ The third argument (text) is used only when a lower level administrator submits a task to the super administrator task queue (should be 'SUPER' if this is the case)

Example: Creating a program to submit the task

The program in this example, LC.SUBMIT, is called from LC.CATALOG. LC.SUBMIT uses TK.ALLOCATE and TK.SUBMIT to register a custom task. The name of a file is passed to the task through a text variable called FILENAME.

```
DEFINE LC.SUBMIT PROGRAM
PROGRAM
argument _fname text
variable _prefix text trap on haderror
filename=_fname "******************************************************
"Submitting your task to the task processor
"In tk.allocate: argument 1 is the name of your task
"(the Task Entry) as define in tk.entry
"argument 2 is either NA or 'SUPER'. SUPER is used if
"this is DBA submitting a task to a Super DBA.
"
"In tk.submit: argument 1 is the number of files you
"want the Task Processor to cleanup after processing.
"It may be 0 in which case you will can dictate the
"names, location and cleanup of file(s).
" argument 2 YES = can restart the task
" argument 3 NA or SUPER used if you are
" DBA submitting to a Super DBA
"*****************************************************
__prefix = tk.allocate('ACTUALS READER', NA)
exp filename to eif file joinchars(_prefix '.1')
CALL TK.SUBMIT(1, 'YES', NA)
"*****************************************************
"Setting messages for the Visual Basic front end.
"SUCCESS = YES means the process has completed normally
"***************************************************** lc.message(lc.msg
'SUCCESS') = 'YES'
lc.message(lc.msg 'MESSAGE') = 'The Task has been submitted.' return
HADERROR:
lc.message(lc.msg 'SUCCESS') = 'NO'
lc.message(lc.msg 'MESSAGE') = 'The Task has NOT been submitted.' signal
errorname errortext
END
```

Creating a program to process the task in the Task Processor

Procedure: Creating a program to process the task in the Task Processor

The following procedure describes how to create a program that processes custom tasks in the Task Processor.

Step	Action
1	Create a program and assign it the same name that you entered into the PRG.MANAGER property of TKS.CATALOG. After you submit the custom task and the Task Processor is run, it calls your custom program several times. The Task Processor monitors the progress of the custom task by looking at a variable called TK.C.STATUS in the FMSSTASK database. The first time the custom task is called, TK.C.STATUS is set to STARTED. After the first time the custom task is called, TK.C.STATUS is set to IN PROGRESS. The custom task will continue to be called until your custom program sets the value of TK.C.STATUS to one of the following: ▪ COMPLETED ▪ FAILED ▪ INTERRUPTED

Step	Action

2	If you want to place task specific information into the task processor log, your custom program should assign the necessary information into TK.C.MESSAGE (text variable).
	The custom program is responsible for attaching, detaching and updating all databases except OFASTASK, OFASERVE, OFATOOLS, and OFALC.
	If the custom task was submitted with associated EIF files, when a custom task driver is executed, the following variables are set:
	■ TK.C.EIFS — number of files associated with a task (could be 0)
	■ TK.C.PATH — full path to the task file's directory
	■ TK.C.PREFIX — current task file's prefix
	Financial Analyzer automatically assigns the name
	`joinchars(tk.c.path tk.c.prefix '.1')`
	to the first EIF file for the task.
	The name of the second EIF file for the task is
	`joinchars(tk.c.path tk.c.prefix '.2')`

3	When a custom task finishes or fails, all files that are associated with it are deleted by the task processor.

Requirements for custom task drivers

Requirements for a custom task program (driver) are as follows:

- The custom task program (driver) should check tk.c.status and if the value is STARTED, your program should set tk.c.message to a text value indicating that your process is started. It should then return from the program, which will allow for the contents of tk.c.message to be displayed. For example, if `tk.c.status eq STARTED` then `tk.c.message = 'The task has started.'`
- The shared database OFAS should be attached and detached using the DB.ATTACH and DB.DETACH functions.
- A custom task driver should never signal an error. If an error occurs, an error message should be assigned to the variable tk.c.message and the value of tk.c.status should be set to FAILED.

- If the task is interrupted (errorname ATTN), then the value of tk.c.status should be set to INTERRUPTED.
- When a custom task driver is finished, it should set the value of tk.c.status to COMPLETED.

Example: Creating a Program to Process the Task in the Task Processor

The ACTREAD program in this example processes the task that is submitted by the program LC.SUBMIT, which is described in the previous example.

```
DEFINE ACTREADER PROGRAM
PROGRAM
variable _file.unit integer
trap on haderror "*******************************************************
"You must let the Task Processor know that you are
"initiating the task. "*****************************************************
If tk.c.status eq 'STARTED'
 then do
TK.C.MESSAGE = 'Started to read Actuals for this Month.'
 return
 doend
"******************************************************
"You must attach the shared database FMSS read/write.
"******************************************************
call db.attach('FMSS' 'rw' 'first')
"******************************************************
"You must import the information contained in any
"associated EIF files. "*****************************************************
If tk.c.eifs gt 0
 then import all from eif file
  -joinchars( tk.c.path,tk.c.prefix,'.1')
"************************************************************ "Your code to
perform the function you want to accomplish.
"Your reader, processor etc.
"************************************************************
"************************************************************ "Suggested tools to
update and detach "************************************************************
call db.update('FMSS')
call db.detach('FMSS')
"************************************************************ "Tell the Task
Processor you completed normally.
"Set a message that is meaningful to you.
"************************************************************ TK.C.STATUS =
'COMPLETED'
```

```
TK.C.MESSAGE = 'Actuals for this Month have been read.'
"**********************************************************
fileclose _file.unit
return "**********************************************************
"use the pattern below for error handling.
"********************************************************** HADERROR:
trap on ERROR noprint
call db.detach('FMSS')
fileclose _file.unit
ERROR:
trap off
if errorname eq 'ATTN'
 then do
tk.c.status = 'INTERRUPTED'
tk.c.message = 'Distribution is Interrupted.'
doend
 else do
tk.c.status = 'FAILED'
tk.c.message = errortext
doend
END
```

B

Generating Depth and Sequence Information for Hierarchies

Overview

Purpose of depth and sequence information

To support easy selection, drill-down, and roll-up capabilities over user defined hierarchies, Financial Analyzer requires information regarding the level and listing sequence for any dimension values inserted in those hierarchies. Calling the CM.UPDATE.HIER program, you can derive this information from the basic relation describing the hierarchy.

When to use CM.UPDATE.HIER

In most cases, you need to call CM.UPDATE.HIER after hierarchy maintenance. However, you do not need to call CM.UPDATE.HIER if either of the following is true:

- You maintained the hierarchy through the user interface,
- Your data reader is called from within Financial Analyzer using the Run Data Loaders dialog box.

Using the CM.UPDATE.HIER Program

Calling CM.UPDATE.HIER

Use the CM.UPDATE.HIER program to derive dimension value level and listing sequence information from the basic relation describing the hierarchy. The

CM.UPDATE.HIER program accepts as argument the internal name for the hierarchy to be analyzed.

Note: The supporting relation should be initialized before calling CM.UPDATE.HIER either through a data loader or manually.

Example: Calling CM.UPDATE.HIER

The following example shows a sample CM.UPDATE.HIER call, either through a data loader or manually.

```
call cm.update.hier ('HI.AA34325')
```

In this example, HI.AA34325 is the internal name of the hierarchy.

To ensure the hierarchy is built correctly after the load, you can add the above line to your data loader program.

Tip: If you know the description for the hierarchy, you can identify the internal name using the following command:

```
shw limit ( hi.entry to hi.desc eq 'My Description')
```

Index