C

Loaders

Generic Loader

The Generic Loader (FNDLOAD) is a concurrent program that can move Oracle Applications data between database and text file representations. The loader reads a configuration file to determine what data to access. For information on specific configuration files consult the *Open Interfaces Guide* for your product group. The following sections describe the operation of the Generic Loader.

> **Warning:** Use only the loader files provided by Oracle Applications. If you use files not provided by Oracle Applications or modify the provided scripts you risk corrupting your database. Oracle does not support the use of custom loader files or modified Oracle Applications loader files.

Overview

The Generic Loader can download data from an application entity into a portable, editable text file. This file can then be uploaded into any other database to copy the data. Conversion between database store and file format is specified by a configuration file that is read by the loader.

The Generic Loader downloads data from a database according to a configuration (.lct) file, and converts the data into a data file (.ldt file). The Generic Loader can then upload this data to another database using a configuration file.

The loader operates in one of two modes: download or upload. In the download mode, data is downloaded from the database to a text file; in the upload mode, data is uploaded from a text file to the database.

Data structures supported by the loader include master-detail relationships and foreign key reference relationships.

In both downloading and uploading, the structure of the data involved is described by a configuration file. The configuration file describes the structure of the data and also the access methods to use to copy the data into or out of the database. The same configuration file may be used for both uploading and downloading.

When downloading, the Generic Loader creates a second file, called the data file, that contains the structured data selected for downloading. The data file has a standard syntax for representing the data that has been downloaded. When uploading, the Generic Loader reads a data file to get the data that it is to upload. In most cases, the data file was produced by a previous download, but may have come from another source. The data file cannot be interpreted without the corresponding configuration file available.

Download database information to a text file

The text file is human-readable and portable, and can be examined and modified with any editor. Generally, a "developer key" is used to identify records written out to text files. For example, the PROFILE_OPTION_NAME, not the PROFILE_OPTION_ID, is used to identify records in the Profiles configuration file.

Upload (merge) the information in a text file to the database

In uploading, if a row exists, but has different attributes, the row is updated. If a row does not exist, a new row is inserted.

Depending on the configuration file, a row that exists in the database but not in the text file may or may not be deleted when the text file is uploaded. Refer to the configuration file to determine how such rows are handled.

These download and upload capabilities allow profile value information that is defined in one database to be easily propagated to other databases. This is useful for delivering Oracle Applications seed data to customers, as well as for copying customer profile definitions from a primary site to other sites.

The text file version of profile value data is also useful for bulk editing operations, which can be accomplished more efficiently with a text editor than with a form.

Preservation of data

FNDLOAD uses the OWNER and LAST_UPDATE_DATE attributes to determine whether to overwrite pre-existing data. The rules applied are:

1. If the entity being uploaded is not present in the database, a new entity is always inserted.
2. Entities uploaded from a file with OWNER=SEED never overwrite entities with OWNER=CUSTOM in the database.
3. Entities with OWNER=CUSTOM uploaded from a file always update entities with OWNER=SEED in the database.
4. If the owner of the entity is the same in the file and database, the entity is updated only if the LAST_UPDATE_DATE in the file is later than the LAST_UPDATE_DATE in the database.

FNDLOAD Executable

The Generic Loader is a concurrent program named FNDLOAD. The concurrent executable takes the following parameters:

```
FNDLOAD apps/pwd 0 Y mode configfile datafile entity [ param ...
]
```

where

<apps/pwd>
The APPS schema and password in the form username/password[@connect_string]. If connect_string is omitted, it is taken in a platform-specific manner from the environment using the name TWO_TASK.

< 0 Y >
Concurrent program flags

mode
UPLOAD or DOWNLOAD. UPLOAD causes the datafile to be uploaded to the database. DOWNLOAD causes the loader to fetch rows and write them to the datafile.

<configfile>
The configuration file to use (usually with a suffix of .lct, but not enforced or supplied by the loader).

<datafile>
The data file to write (usually with a suffix of .ldt, but not enforced or supplied by the loader). If the data file already exists, it will be overwritten.

<entity>
The entity(ies) to upload or download. When uploading, you should always upload all entities, so specify a "-" to upload all entities.

< [param] >
Zero or more additional parameters are used to provide bind values in the access SQL (both UPLOAD and DOWNLOAD). Each parameter is in the form NAME=VALUE. NAME should not conflict with an attribute name for the entities being loaded.

File Specifications

The configuration file and data file parameters are specified in one of two ways:

```
@<application_short_name>:[<dir>/.../]file.ext
```

For example,

```
@fnd/11i/loader/fndapp.lct
@po:install/data/poreq.ldt
```

Alternatively, the parameters can be specified as such:

```
<native path>
```

For example,

```
mydata.ldt
c:\loader\config\cfg102.lct
```

Examples

An example of downloading is:

```
FNDLOAD apps/apps@devdb 0 Y
     DOWNLOAD testcfg.lct out.ldt FND_APPLICATION_TL APPSNAME=FND
```

This command does the following:

- connects to apps/apps@devd
- downloads data using the configuration file testcfg.lct
- writes data to data file out.ldt
- downloads the FND_APPLICATION_TL entity with APPSNAME parameter defined as value 'FND'

An example of uploading is:

```
FNDLOAD apps/apps@custdb 0 Y
     UPLOAD fndapp.lct fnd1234.ldt -
```

This command does the following:

- connects to apps/apps@custdb
- uploads data using the configuration file in fndapp.lct from data file in fnd1234.ldt
- The contents of the entire data file is uploaded.

Configuration File

Operation of the Generic Loader is controlled by the specified configuration file. The configuration file contains the following:

- DEFINE block
- DOWNLOAD block
- UPLOAD block

The contents of the configuration file specify the structure of the data and the access methods to use to move the data between the data file and a database.

DEFINE Block

The DEFINE block specifies the structure of the datafile records. The define block format is identical to that already generated by existing Oracle Application Object Library loaders. The structure of this section is

```
DEFINE <entity> KEY <key_attribute_name> <datatype> ...
 (BASE|TRANS|CTX) <attribute_name> <datatype> ...
 [DEFINE <child_entity> ...]
END <entity>
```

Example

```
DEFINE FND_LOOKUP_TYPE
 KEY  VIEW_APPSNAME VARCHAR2(50)
 KEY  LOOKUP_TYPE  VARCHAR2(30)
 BASE  OWNER    VARCHAR2(6)
 TRANS  MEANING   VARCHAR2(80)
 TRANS  DESCRIPTION  VARCHAR2(240)
 DEFINE FND_LOOKUP_VALUE
  KEY LOOKUP_CODE   VARCHAR2(30)
  BASE END_DATE_ACTIVE VARCHAR2(10)
  BASE OWNER     VARCHAR2(6)
  TRANS MEANING    VARCHAR2(80)
  TRANS DESCRIPTION   VARCHAR2(240)
  CTX TAG     VARCHAR2(30)
  END FND_LOOKUP_VALUE
 END FND_LOOKUP_TYPE

VIEW_APPSNAME
```

One or more KEY attributes defines the primary key of each entity. BASE and CTX attributes are those that do not require translation. TRANS attributes do. Note that BASE and CTX attributes are treated identically. That is, CTX is just a synonym for BASE. The CTX attribute type is provided merely to allow users to optionally differentiate between

BASE attributes. For example, translators may wish to simplify their .ldt files by stripping out the BASE attributes. However, they may also want to keep some BASE attributes for context. By denoting some attributes as BASE and some as CTX, they can control which attributes to remove.

Data types can be standard Oracle scalar types, except that only VARCHAR2 is currently supported. An attribute can also be defined as a foreign key reference to another entity in your configuration file

The foreign key entity must be a "top-level" entity and its download statement must include filter parameters in its WHERE clause for each of its key attributes. Also, the parameter names must match the key attribute names exactly.

Note that entity definitions can be nested to indicate master-detail relationships. Nested entity definitions inherit the key attributes of their parent entities and should not redefine them.

DOWNLOAD Statement

The DOWNLOAD statement is a SQL statement that selects rows to download. The statement can join to other tables to resolve sequence generated ID numbers into developer keys where possible. The DOWNLOAD statement may also contain bind values of the form ':NAME' which are substituted with parameter values from the command line. DOWNLOAD statements have the form

```
DOWNLOAD <entity> "select <attribute expressions> from ..."
```

Example

```
DOWNLOAD FND_LOOKUP_TYPE
  "select VA.APPLICATION_SHORT_NAME VIEW_APPSNAME,
   LT.LOOKUP_TYPE,
   OA.APPLICATION_SHORT_NAME,
   LT.CUSTOMIZATION_LEVEL,
   decode(LT.LAST_UPDATED_BY, 1, 'SEED', 'CUSTOM') OWNER,
   LT.MEANING,
   LT.DESCRIPTION
   from   FND_LOOKUP_TYPES_VL LT,
   FND_APPLICATION VA,
   FND_APPLICATION OA,
   FND_SECURITY_GROUPS SG
   where  VA.APPLICATION_ID = LT.VIEW_APPLICATION_ID
   and    OA.APPLICATION_ID = LT.APPLICATION_ID
   and    (:VIEW_APPSNAME is null or
     (:VIEW_APPSNAME is not null
   and VA.APPLICATION_SHORT_NAME like :VIEW_APPSNAME))
   and    (:LOOKUP_TYPE is null or
      (LOOKUP_TYPE is not null and LT.LOOKUP_TYPE like :LOOKUP_TYP
E))
   and    SG.SECURITY_GROUP_ID = LT.SECURITY_GROUP_ID
   and    ((:SECURITY_GROUP is null and SG.SECURITY_GROUP_KEY = 'S
TANDARD') or
     (:SECURITY_GROUP is not null
   and SG.SECURITY_GROUP_KEY = :SECURITY_GROUP))
   order by 1, 2 "
```

Download statements for child entities may reference any key attribute of the parent entity, or any command line parameter.

UPLOAD Statement

The UPLOAD statement is a SQL statement or PL/SQL anonymous block which accepts file data and applies it to the database. The statement is executed once for each record read from the data file. Bind values in the statement are satisfied by attributes from the file data or command line parameters.

Example

```
UPLOAD FND_LOOKUP_TYPE
 BEGIN
 " begin
   if (:UPLOAD_MODE = 'NLS') then
     fnd_lookup_types_pkg.TRANSLATE_ROW(
  x_lookup_type  => :LOOKUP_TYPE,
  x_security_group      => :SECURITY_GROUP,
  x_view_application    => :VIEW_APPSNAME,
  x_owner        => :OWNER,
  x_meaning      => :MEANING,
  x_description  => :DESCRIPTION);
   else
  fnd_lookup_types_pkg.LOAD_ROW(
  x_lookup_type  => :LOOKUP_TYPE,
  x_security_group      => :SECURITY_GROUP,
  x_view_application    => :VIEW_APPSNAME,
  x_owner        => :OWNER,
  x_meaning      => :MEANING,
  x_description  => :DESCRIPTION);
   end if;
   end; "
```

As in the DOWNLOAD, the UPLOAD statement for child entities may reference any attributes from the parent record.

Example

```
UPLOAD FND_LOOKUP_VALUE
  " begin
 if (:UPLOAD_MODE = 'NLS') then
   fnd_lookup_values_pkg.TRANSLATE_ROW(
     x_lookup_type  => :LOOKUP_TYPE,
     x_lookup_code  => :LOOKUP_CODE,
     x_security_group      => :SECURITY_GROUP,
     x_view_application    => :VIEW_APPSNAME,
     x_owner        => :OWNER,
     x_meaning      => :MEANING,
     x_description  => :DESCRIPTION);
 else
   fnd_lookup_values_pkg.LOAD_ROW(
     x_lookup_type  => :LOOKUP_TYPE,
     x_lookup_code  => :LOOKUP_CODE,
     x_security_group      => :SECURITY_GROUP,
     x_view_application    => :VIEW_APPSNAME,
     x_owner        => :OWNER,
     x_meaning      => :MEANING,
     x_description  => :DESCRIPTION,
     x_tag   => :TAG);
 end if;
  end;"
```

Data File

A data file is a portable text file. The data file created from a download using the above configuration file would include:

```
# -- Begin Entity Definitions --

DEFINE FND_LOOKUP_TYPE
   KEY   VIEW_APPSNAME VARCHAR2(50)
   KEY   LOOKUP_TYPE VARCHAR2(30)
   BASE  OWNER VARCHAR2(6)
   TRANS MEANING VARCHAR2(80)
   TRANS DESCRIPTION VARCHAR2(240)
   DEFINE FND_LOOKUP_VALUE
     KEY   LOOKUP_CODE VARCHAR2(30)
     BASE  END_DATE_ACTIVE VARCHAR2(10)
     BASE  OWNER VARCHAR2(6)
     TRANS MEANING VARCHAR2(80)
     TRANS DESCRIPTION VARCHAR2(240)
     BASE  TAG VARCHAR2(30)
   END FND_LOOKUP_VALUE
 END FND_LOOKUP_TYPE

# -- End Entity Definitions --

BEGIN FND_LOOKUP_TYPE "FND" "YES_NO"
   OWNER = "SEED"
   MEANING = "Yes or No"

   BEGIN FND_LOOKUP_VALUE Y
     OWNER = "SEED"
     MEANING = "Yes"
   END FND_LOOKUP_VALUE

  BEGIN FND_LOOKUP_VALUE N
     OWNER = "SEED"
     MEANING = "No"
   END FND_LOOKUP_VALUE
END FND_LOOKUP_TYPE
```

Oracle Application Object Library Configuration Files

Oracle Application Object Library provides several configuration files for the Generic Loader that you can use with your setup data.

These configuration files operate on the following data:

- Concurrent program definitions
- Request groups
- Lookup types and lookup values
- Profile options and profile option values
- Flexfields setup data
- Attachments definitions
- Messages
- Security information

Concurrent Program Configuration File

The concurrent program configuration file afcpprog.lct downloads and uploads concurrent program definitions. It takes as parameters program name and application name.

The following table lists the entities, sub-entities (if any), and download parameters for this configuration file.

Entity	Sub-entities, if any	Download Parameters
PROGRAM	INCOMPATIBILITY	CONCURRENT_PROGRAM_NAME APPLICATION_SHORT_NAME
EXECUTABLE		EXECUTABLE_NAME

The entity definition is:

```
DEFINE PROGRAM
  KEY   CONCURRENT_PROGRAM_NAME   VARCHAR2(30)
  KEY   APPLICATION       VARCHAR2(50)
  CTX   OWNER         VARCHAR2(7)
  TRANS USER_CONCURRENT_PROGRAM_NAME  VARCHAR2(240)
  BASE  EXEC_APPLICATION     VARCHAR2(50)
  BASE  EXECUTABLE_NAME      VARCHAR2(30)
  BASE  EXECUTION_METHOD_CODE   VARCHAR2(1)
  BASE  ARGUMENT_METHOD_CODE    VARCHAR2(1)
  BASE  QUEUE_CONTROL_FLAG            VARCHAR2(1)
  BASE  QUEUE_METHOD_CODE             VARCHAR2(1)
  BASE  REQUEST_SET_FLAG              VARCHAR2(1)
  BASE  ENABLED_FLAG                  VARCHAR2(1)
  BASE  PRINT_FLAG                    VARCHAR2(1)
  BASE  RUN_ALONE_FLAG                VARCHAR2(1)
  BASE  SRS_FLAG                      VARCHAR2(1)
  TRANS DESCRIPTION                   VARCHAR2(240)
  BASE  CLASS_APPLICATION         VARCHAR2(50)
  BASE  CONCURRENT_CLASS_NAME         VARCHAR2(30)
  BASE  EXECUTION_OPTIONS             VARCHAR2(250)
  BASE  SAVE_OUTPUT_FLAG              VARCHAR2(1)
  BASE  REQUIRED_STYLE                VARCHAR2(1)
  BASE  OUTPUT_PRINT_STYLE            VARCHAR2(30)
  BASE  PRINTER_NAME                  VARCHAR2(30)
  BASE  MINIMUM_WIDTH                 VARCHAR2(50)
  BASE  MINIMUM_LENGTH                VARCHAR2(50)
  BASE  REQUEST_PRIORITY              VARCHAR2(50)
  BASE  ATTRIBUTE_CATEGORY      VARCHAR2(30)
  BASE  ATTRIBUTE1                VARCHAR2(150)
  BASE  ATTRIBUTE2                VARCHAR2(150)
  BASE  ATTRIBUTE3                VARCHAR2(150)
  BASE  ATTRIBUTE4                VARCHAR2(150)
  BASE  ATTRIBUTE5                VARCHAR2(150)
  BASE  ATTRIBUTE6                VARCHAR2(150)
  BASE  ATTRIBUTE7                VARCHAR2(150)
  BASE  ATTRIBUTE8               VARCHAR2(150)
  BASE  ATTRIBUTE9            VARCHAR2(150)
  BASE  ATTRIBUTE10             VARCHAR2(150)
  BASE  ATTRIBUTE11               VARCHAR2(150)
  BASE  ATTRIBUTE12               VARCHAR2(150)
  BASE  ATTRIBUTE13              VARCHAR2(150)
  BASE  ATTRIBUTE14               VARCHAR2(150)
  BASE  ATTRIBUTE15               VARCHAR2(150)
  BASE  OUTPUT_FILE_TYPE          VARCHAR2(4)
  BASE  RESTART                   VARCHAR2(1)
  BASE  NLS_COMPLIANT             VARCHAR2(1)
  BASE  CD_PARAMETER           VARCHAR2(240)
  BASE  INCREMENT_PROC            VARCHAR2(61)
  BASE  MLS_EXECUTABLE_APPLICATION  VARCHAR2(50)
  BASE  MLS_EXECUTABLE_NAME          VARCHAR2(50)
  BASE  ENABLE_TIME_STATISTICS        VARCHAR2(1)
  BASE  SECURITY_GROUP_NAME           NUMBER
  BASE  RESOURCE_CONSUMER_GROUP       VARCHAR2(30)
  BASE  ROLLBACK_SEGMENT              VARCHAR2(30)
  BASE  OPTIMIZER_MODE              VARCHAR2(30)

END PROGRAM
```

Request Groups Configuration File

Use the file afcpreqg.lct for loading request group data.

The following table lists the entities, sub-entities (if any), and download parameters for this configuration file.

Entity	Sub-entities, if any	Download Parameters
REQUEST_GROUP	REQUEST_GROUP_ UNIT	REQUEST_GROUP_NAME APPLICATION_SHORT_ NAME

The entity definition is:

```
DEFINE REQUEST_GROUP
  KEY   REQUEST_GROUP_NAME    VARCHAR2(30)
  KEY   APPLICATION_SHORT_NAME  VARCHAR2(50)
  CTX   OWNER               VARCHAR2(7)
  TRANS DESCRIPTION          VARCHAR2(800)
  BASE  REQUEST_GROUP_CODE    VARCHAR2(30)
END REQUEST_GROUP
```

Lookups Configuration File

Use the file aflvmlu.lct for loading Lookup types and Lookups values.

The following table lists the entities, sub-entities (if any), and download parameters for this configuration file.

Entity	Sub-entities, if any	Download Parameters
FND_LOOKUP_TYPE	FND_LOOKUP_VALUE	VIEW_APPSNAME LOOKU P_TYPE SECURITY_GROUP

The entity definition is:

```
DEFINE FND_LOOKUP_TYPE
  KEY   VIEW_APPSNAME     VARCHAR2(50)
  KEY   LOOKUP_TYPE      VARCHAR2(30)
  CTX   APPLICATION_SHORT_NAME  VARCHAR2(50)
  BASE  CUSTOMIZATION_LEVEL   VARCHAR2(1)
  CTX   OWNER        VARCHAR2(6)
  TRANS MEANING       VARCHAR2(80)
  TRANS DESCRIPTION      VARCHAR2(240)

  DEFINE FND_LOOKUP_VALUE
    KEY   LOOKUP_CODE    VARCHAR2(30)
    BASE  ENABLED_FLAG    VARCHAR2(1)
    BASE  START_DATE_ACTIVE  VARCHAR2(10)
    BASE  END_DATE_ACTIVE   VARCHAR2(10)
    BASE  TERRITORY_CODE   VARCHAR2(2)
    BASE  TAG                   VARCHAR2(30)
    BASE  ATTRIBUTE_CATEGORY  VARCHAR2(30)
    BASE  ATTRIBUTE1     VARCHAR2(150)
    BASE  ATTRIBUTE2        VARCHAR2(150)
    BASE  ATTRIBUTE3       VARCHAR2(150)
    BASE  ATTRIBUTE4       VARCHAR2(150)
    BASE  ATTRIBUTE5       VARCHAR2(150)
    BASE  ATTRIBUTE6       VARCHAR2(150)
    BASE  ATTRIBUTE7       VARCHAR2(150)
    BASE  ATTRIBUTE8        VARCHAR2(150)
    BASE  ATTRIBUTE9        VARCHAR2(150)
    BASE  ATTRIBUTE10       VARCHAR2(150)
    BASE  ATTRIBUTE11       VARCHAR2(150)
    BASE  ATTRIBUTE12        VARCHAR2(150)
    BASE  ATTRIBUTE13      VARCHAR2(150)
    BASE  ATTRIBUTE14       VARCHAR2(150)
    BASE  ATTRIBUTE15       VARCHAR2(150)
    CTX   OWNER             VARCHAR2(6)
    TRANS MEANING          VARCHAR2(80)
    TRANS DESCRIPTION      VARCHAR2(240)
  END FND_LOOKUP_VALUE
END FND_LOOKUP_TYPE
```

Profile Options and Profile Values Configuration File

Use the file afscprof.lct for loading profile options and profile values.

The following table lists the entities, sub-entities (if any), and download parameters for this configuration file.

Entity	Sub-entities, if any	Download Parameters
PROFILE	FND_PROFILE_ OPTION_ VALUES	PROFILE_NAME APPLICAT ION_SHORT_NAME

The entity definition is:

```
DEFINE PROFILE
  KEY   PROFILE_NAME      VARCHAR2(80)
  CTX   OWNER         VARCHAR2(7)
  CTX   APPLICATION_SHORT_NAME   VARCHAR2(50)
  TRANS USER_PROFILE_OPTION_NAME  VARCHAR2(240)
  TRANS DESCRIPTION       VARCHAR2(240)
  BASE  USER_CHANGEABLE_FLAG   VARCHAR2(1)
  BASE  USER_VISIBLE_FLAG    VARCHAR2(1)
  BASE  READ_ALLOWED_FLAG    VARCHAR2(1)
  BASE  WRITE_ALLOWED_FLAG    VARCHAR2(1)
  BASE  SITE_ENABLED_FLAG    VARCHAR2(1)
  BASE  SITE_UPDATE_ALLOWED_FLAG  VARCHAR2(1)
  BASE  APP_ENABLED_FLAG     VARCHAR2(1)
  BASE  APP_UPDATE_ALLOWED_FLAG  VARCHAR2(1)
  BASE  RESP_ENABLED_FLAG    VARCHAR2(1)
  BASE  RESP_UPDATE_ALLOWED_FLAG  VARCHAR2(1)
  BASE  USER_ENABLED_FLAG    VARCHAR2(1)
  BASE  USER_UPDATE_ALLOWED_FLAG  VARCHAR2(1)
  BASE  START_DATE_ACTIVE    VARCHAR2(11)
  BASE  END_DATE_ACTIVE     VARCHAR2(11)
  BASE  SQL_VALIDATION      VARCHAR2(2000)

  DEFINE FND_PROFILE_OPTION_VALUES
    KEY   LEVEL        VARCHAR2(50)
    KEY   LEVEL_VALUE     VARCHAR2(100)
    KEY   LEVEL_VALUE_APP    VARCHAR2(50)
    CTX   OWNER       VARCHAR2(7)
    BASE  PROFILE_OPTION_VALUE  VARCHAR2(240)
  END FND_PROFILE_OPTION_VALUES
END PROFILE
```

Flexfields Setup Data Configuration File

Use the file afffload.lct for loading flexfields data.

> **Warning:** Do not modify the data files you download using the flexfields configuration file. You risk corrupting your flexfields data. Oracle Applications does not support any changes you make to the data files.

The configuration file includes the following entities:

- Value sets
- Descriptive flexfields
- Key flexfields

Flexfield Value Sets

The entity VALUE_SET includes the following table details of table validated value sets, and user exit details of special/pair validated value sets. It includes the values, the normalized value hierarchy, value qualifier values, security rules, security rule lines, security rule usage details, rollup groups, or value hierarchies for the value set.

The key for this entity is FLEX_VALUE_SET_NAME.

Example

```
>FNDLOAD apps/apps 0 Y DOWNLOAD @FND:admin/import/afffload.lct ou
t.ldt \
 VALUE_SET FLEX_VALUE_SET_NAME="Loader_Test"

>FNDLOAD apps/apps 0 Y UPLOAD @FND:admin/import/afffload.lct out.l
dt -
```

Descriptive Flexfields

The entity DESC_FLEX includes context column, attribute columns, context, and segment details. This entity references the VALUE_SET for the value set used by a given SEGMENT.

The key is composed of APPLICATION_SHORT_NAME and DESCRIPTIVE_FLEXF IELD_NAME.

Example

```
>FNDLOAD apps/apps 0 Y DOWNLOAD @FND:admin/import/afffload.lct ou
t.ldt \
DESC_FLEX APPLICATION_SHORT_NAME="FND" DESCRIPTIVE_FLEXFIELD_NAME=
"FND_FLEX_TEST"

>FNDLOAD apps/apps 0 Y UPLOAD @FND:admin/import/afffload.lct out.l
dt -
```

Key Flexfields

The entity KEY_FLEX includes the unique ID column, structure column, segment columns, flexfield qualifier, segment qualifier, structure, Account Generator workflow process, shorthand alias, cross-validation rule, cross-validation rule line, segment, flexfield qualifier assignment, and segment qualifier assignment details.

References VALUE_SET for the value set used by the given segment.

The key is composed of APPLICATION_SHORT_NAME and ID_FLEX_CODE.

Example

```
>FNDLOAD apps/apps 0 Y DOWNLOAD @FND:admin/import/afffload.lct ou
t.ldt \
KEY_FLEX APPLICATION_SHORT_NAME="SQLGL" ID_FLEX_CODE="GL#"

>FNDLOAD apps/apps 0 Y UPLOAD @FND:admin/import/afffload.lct out.l
dt -
```

Attachments Setup Data Configuration File

Use the file afattach.lct for loading attachments setup data.

The following table lists the entities, sub-entities (if any), and download parameters for this configuration file.

Entity	Sub-entities, if any	Download Parameters
FND_ATTACHMENT_ FUNCTIONS	FND_ATTACHMENT_ BLOCKS FND_ ATTACHMENT_BLOCK_ ENTITIES FND_DOC_ CATEGORY_USAGES	APPLICATION_ SHORT_ NAME FUNCTION_NAME FUNCTION_TYPE

The entity definitions are:

```
DEFINE FND_ATTACHMENT_FUNCTIONS
  KEY   FUNCTION_NAME    VARCHAR2(30)
  KEY   FUNCTION_TYPE    VARCHAR2(1)
  KEY   APP_SHORT_NAME    VARCHAR2(50)
  CTX   OWNER       VARCHAR2(7)
  BASE  SESSION_CONTEXT_FIELD  VARCHAR2(61)
  BASE  ENABLED_FLAG      VARCHAR2(1)

  DEFINE FND_ATTACHMENT_BLOCKS
    KEY   BLOCK_NAME    VARCHAR2(30)
    CTX   OWNER     VARCHAR2(7)
    BASE  QUERY_FLAG    VARCHAR2(1)
    BASE  SECURITY_TYPE   VARCHAR2(50)
    BASE  ORG_CONTEXT_FIELD VARCHAR2(61)
    BASE  SET_OF_BOOKS_CONTEXT_FIELD  VARCHAR2(61)
    BASE  BUSINESS_UNIT_CONTEXT_FIELD  VARCHAR2(61)
    BASE  CONTEXT1_FIELD   VARCHAR2(61)
    BASE  CONTEXT2_FIELD   VARCHAR2(61)
    BASE  CONTEXT3_FIELD   VARCHAR2(61)

    DEFINE FND_ATTACHMENT_BLK_ENTITIES
  KEY   BLK_ENTITY  REFERENCES
FND_DOCUMENT_ENTITIES
  BASE  DISPLAY_METHOD     VARCHAR2(1)
  BASE  INCLUDE_IN_INDICATOR_FLAG  VARCHAR2(1)
  CTX   OWNER      VARCHAR2(7)
  BASE  PK1_FIELD     VARCHAR2(61)
  BASE  PK2_FIELD     VARCHAR2(61)
  BASE  PK3_FIELD     VARCHAR2(61)
  BASE  PK4_FIELD     VARCHAR2(61)
  BASE  PK5_FIELD     VARCHAR2(61)
  BASE  SQL_STATEMENT   VARCHAR2(2000)
  BASE  INDICATOR_IN_VIEW_FLAG  VARCHAR2(1)
  BASE  QUERY_PERMISSION_TYPE   VARCHAR2(1)
  BASE  INSERT_PERMISSION_TYPE  VARCHAR2(1)
  BASE  UPDATE_PERMISSION_TYPE  VARCHAR2(1)
  BASE  DELETE_PERMISSION_TYPE  VARCHAR2(1)
  BASE  CONDITION_FIELD    VARCHAR2(61)
        BASE  CONDITION_OPERATOR VARCHAR2(50)
  BASE  CONDITION_VALUE1   VARCHAR2(100)
  BASE  CONDITION_VALUE2   VARCHAR2(100)
    END FND_ATTACHMENT_BLK_ENTITIES
  END FND_ATTACHMENT_BLOCKS

  DEFINE FND_DOC_CATEGORY_USAGES
    KEY CATEGORY_USAGE REFERENCES
    FND_DOCUMENT_CATEGORIES
```

```
    BASE ENABLED_FLAG VARCHAR2(1)
    CTX  OWNER   VARCHAR2(7)
 END FND_DOC_CATEGORY_USAGES
END FND_ATTACHMENT_FUNCTIONS

DEFINE FND_DOCUMENT_ENTITIES
  KEY   DATA_OBJECT_CODE  VARCHAR2(30)
  BASE  APP_SHORT_NAME   VARCHAR2(50)
  BASE  TABLE_NAME     VARCHAR2(30)
  BASE  ENTITY_NAME   VARCHAR2(40)
  CTX   OWNER     VARCHAR2(7)
  BASE  PK1_COLUMN   VARCHAR2(30)
  BASE  PK2_COLUMN   VARCHAR2(30)
  BASE  PK3_COLUMN   VARCHAR2(30)
  BASE  PK4_COLUMN   VARCHAR2(30)
  BASE  PK5_COLUMN   VARCHAR2(30)
  TRANS USER_ENTITY_NAME  VARCHAR2(240)
  TRANS USER_ENTITY_PROMPT VARCHAR2(40)
END FND_DOCUMENT_ENTITIES

DEFINE FND_DOCUMENT_CATEGORIES
  KEY   CATEGORY_NAME   VARCHAR2(30)
  BASE  APP_SHORT_NAME   VARCHAR2(50)
  CTX   OWNER      VARCHAR2(7)
  BASE  START_DATE_ACTIVE  VARCHAR2(11)
  BASE  END_DATE_ACTIVE   VARCHAR2(11)
  BASE  ATTRIBUTE_CATEGORY VARCHAR2(30)
  BASE  ATTRIBUTE1   VARCHAR2(150)
  BASE  ATTRIBUTE2   VARCHAR2(150)
  BASE  ATTRIBUTE3   VARCHAR2(150)
  BASE  ATTRIBUTE4   VARCHAR2(150)
  BASE  ATTRIBUTE5   VARCHAR2(150)
  BASE  ATTRIBUTE6   VARCHAR2(150)
  BASE  ATTRIBUTE7   VARCHAR2(150)
  BASE  ATTRIBUTE8   VARCHAR2(150)
  BASE  ATTRIBUTE9   VARCHAR2(150)
  BASE  ATTRIBUTE10   VARCHAR2(150)
  BASE  ATTRIBUTE11   VARCHAR2(150)
  BASE  ATTRIBUTE12   VARCHAR2(150)
  BASE  ATTRIBUTE13   VARCHAR2(150)
  BASE  ATTRIBUTE14   VARCHAR2(150)
  BASE  ATTRIBUTE15   VARCHAR2(150)
  BASE  DEFAULT_DATATYPE_ID VARCHAR2(50)
  BASE  APP_SOURCE_VERSION VARCHAR2(255)
  TRANS USER_NAME     VARCHAR2(255)
END FND_DOCUMENT_CATEGORIES

DEFINE FND_DOCUMENT_DATATYPES
  KEY   DATATYPE_ID   VARCHAR2(50)
  KEY   NAME     VARCHAR2(30)
  CTX   OWNER     VARCHAR2(7)
  BASE  START_DATE_ACTIVE VARCHAR2(11)
  BASE  END_DATE_ACTIVE  VARCHAR2(11)
  TRANS USER_NAME    VARCHAR2(30)
END FND_DOCUMENT_DATATYPES
```

Messages Configuration File

Use the file afmdmsg.lct for uploading and downloading messages in a database.

Use the Generic Loader and afmdmsg.lct for transferring messages between databases only. Use the Message Dictionary Generator for moving messages into binary runtime files and readable text files. See: Message Dictionary Generator, page C-18.

The following table lists the entities, sub-entities (if any), and download parameters for this configuration file.

Entity	Sub-entities, if any	Download Parameters
FND_NEW_MESSAGES		APPLICATION_SHORT_ NAME MESSAGE_NAME

The entity definition is:

> **Note:** to change the language you are downloading, set the environment variable NLS_LANG before running the loader.

```
DEFINE FND_NEW_MESSAGES
  KEY   APPLICATION_SHORT_NAME   VARCHAR2(50)
  KEY   MESSAGE_NAME             VARCHAR2(30)
  CTX   OWNER                    VARCHAR2(7)
  CTX   MESSAGE_NUMBER           VARCHAR2(50)
  TRANS MESSAGE_TEXT             VARCHAR2(2000)
  CTX   DESCRIPTION              VARCHAR2(240)
  CTX   TYPE                     VARCHAR2(30)
  CTX   MAX_LENGTH               NUMBER
END FND_NEW_MESSAGES
```

Security Information Configuration File

Use the file afsload.lct for downloading and uploading forms, functions, menus, and menu entries.

The following table lists the entities, sub-entities (if any), and download parameters for this configuration file.

Entity	Sub-entities, if any	Download Parameters
FORM		FORM_APP_SHORT_NAME, FORM_NAME
FUNCTION		FUNC_APP_SHORT_NAME FUNCTION_NAME
MENU	ENTRY	MENU
ENTRY		[None]

The entity definition is:

```
DEFINE FORM
  KEY   APPLICATION_SHORT_NAME VARCHAR2(50)
  KEY   FORM_NAME VARCHAR2(30)
  TRANS USER_FORM_NAME VARCHAR2(80)
  TRANS DESCRIPTION VARCHAR2(240)
  CTX   OWNER VARCHAR2(7)
END FORM

DEFINE FUNCTION
  KEY   FUNCTION_NAME VARCHAR2(30)
  BASE  FORM REFERENCES FORM
  BASE  TYPE VARCHAR2(30)
  BASE  PARAMETERS VARCHAR2(2000)
  BASE  WEB_HOST_NAME VARCHAR2(80)
  BASE  WEB_AGENT_NAME VARCHAR2(80)
  BASE  WEB_HTML_CALL VARCHAR2(240)
  BASE  WEB_ENCRYPT_PARAMETERS VARCHAR2(1)
  BASE  WEB_SECURED VARCHAR2(1)
  BASE  WEB_ICON VARCHAR2(30)
  TRANS USER_FUNCTION_NAME VARCHAR2(80)
  TRANS DESCRIPTION VARCHAR2(240)
  CTX   OWNER VARCHAR2(7)
END FUNCTION

DEFINE MENU
  KEY   MENU_NAME VARCHAR2(30)
  TRANS USER_MENU_NAME VARCHAR2(80)
  TRANS DESCRIPTION VARCHAR2(240)
  CTX   OWNER VARCHAR2(7)

 DEFINE ENTRY
    TRANS PROMPT VARCHAR2(60)
    TRANS DESCRIPTION VARCHAR2(240)
    CTX   SUBMENU REFERENCES MENU
    CTX   FUNCTION REFERENCES FUNCTION
    CTX   OWNER VARCHAR2(7)
  END ENTRY
END MENU
```

Message Dictionary Generator

The Message Dictionary Generator (FNDMDGEN) is a concurrent program that generates binary runtime files from the database for Oracle Applications Message Dictionary messages. The following sections describe the operation of the Message Dictionary Generator.

For more information on using the Message Dictionary and creating messages, see the *Oracle Applications Developer's Guide*.

> **Note:** Use the Generic Loader and corresponding configuration file for uploading and downloading message text files into a database.

Message Repositories

Message information is stored in two different repositories, each of which has its own format and serves a specific need. Following is a description for each of the message repositories, including the message attributes they store.

Database

The FND_NEW_MESSAGES table in the database stores all Oracle Applications messages for all languages. Database messages are directly used only by the stored procedure Message Dictionary API. Database message data can be edited using the Messages form.

Database Attributes are: APPLICATION, LANGUAGE, NAME, NUMBER, TEXT, DESCRIPTION

Runtime

A runtime binary file stores the messages for a single application and a single language. The file is optimized for rapid lookup of individual messages by message NAME.

A runtime file is located in:

```
<APPL_TOP>/$APPLMSG/<LANGUAGE>.msb
```

where <APPL_TOP> is the application basepath, APPLMSG is an environment variable whose usual value is "mesg", and <LANGUAGE> is the NLS language code (for example: 'US', or 'F'.). A typical message file would be $FND_TOP/mesg/US.msb.

Runtime Attributes are: NAME, NUMBER, TEXT

Usage

The help that you get when you invoke the Message Dictionary Generator without any program arguments (i.e., FNDMDGEN dbuser/dbpassword 0 Y) is:

```
FNDMDGEN <Oracle ID/password> 0 Y <language codename> [applicatio
n shortname] [mode] [filename] \
```

where mode is:

DB_TO_ RUNTIME
From Database to Runtime file (.msb)

> **Note:** In Release 11*i* the mode DB_TO_RUNTIME only is supported, unlike in previous releases.

Wildcards

Either <language codename> or [application shortname] can be wildcarded by passing the value "ALL". The following describes how wildcards are used:

From DB
Messages come from the FND_NEW_MESSAGES table. Wildcards match all the messages in the database.

To RUNTIME
In the case of wildcards, separate runtime files are created for each combination of language and application.

Generic File Manager Access Utility (FNDGFU)

The Generic File Manager (GFM) is a set of PL/SQL procedures that leverages Oracle HTTP Server functionality to provide generic upload and download capabilities of unstructured data between a client (usually a web browser) and a database.

FNDGFU is an access utility that allows the upload of files from the local file system to the GFM database file system. It supports simple uploads of single files as well as bulk uploads of many files. FNDGFU also offers a download option that provides a convenient and quick means of retrieving the contents of large objects (LOBs) if the file identifier is known.

To delete files loaded to the database run the Purge Obsolete Generic File Manager Data concurrent program.

Usage

FNDGFU is located in the $FND_TOP/bin directory. Putting this directory on your path will allow you to invoke FNDGFU easily.

Upload files to the GFM

To upload files using FNDGFU use the following syntax:

```
FNDGFU <logon> [param] <filenames>
```

where

<logon>
Specifies a standard Oracle logon string of the form username/password. To specify a particular database, append an @ sign and the database SID (@database).

[param]
Includes the following parameters (in any order) as appropriate:

PROGRAM_NAME=<name> specifies the name of the program on whose behalf the LOB is to be maintained.

PROGRAM_TAG=<name> specifies the program tag, which is a string used by the GFM client program to further categorize the LOB.

LANGUAGE=<language_code> specifies the language of the file.

PLS_CALLBACK=<plsql procedure> specifies the procedure to execute once for each uploaded file. The procedure must accept file_id as its only parameter. FNDGFU will call the specified procedure after each uploaded file, passing in the new file identifier, for example: PLS_CALLBACK=mypackage.myprocedure.

CONTENT_TYPE=<mime_type> specifies the default mime type to use for uploaded files not qualified by a content map.

CONTENT_MAP=<contentmapfile> specifies a text file that maps filename suffixes onto content types. The text file consists of lines of the form <suffix>=<mime_type> where suffix is any string matched against the end of the filename. For example: ".txt = text/plain", ".html = text/html", and ".ps = application/postscript".

<filenames>
Specifies the files to upload. Any number of files may be uploaded.

Download files from the GFM

To download a file using the FNDGFU utility, use the following syntax:

```
FNDGFU <logon> DOWNLOAD=<fileid> [LINE_BREAKS=<mode>] [filename]
```

where

<logon>
Specifies a standard Oracle logon string of the form username/password. To specify a particular database, append an @ sign and the database SID (@database).

<fileid>
Specifies the identifier of the large object (LOB) to download.

<mode>
Specifies how to treat line breaks for a text document. This parameter is ignored for nontext content. The following values are valid:

LF - Line breaks will be represented using "/n" in the downloaded output. This is the default mode if the LINE_BREAK parameter is omitted.

CRLF - Line breaks will be left in the canonical format.

[filename]
Specifies the file into which to download. If omitted, downloaded contents are streamed to the standard output.

Example of FNDGFU Upload

The FNDGFU utility can be used to upload new or changed help files. Use the following arguments to upload help files:

```
FNDGFU <apps/pwd> 0 Y PROGRAM_NAME=FND_HELP PROGRAM_TAG=<applicat
ion>:<custom_level> CONTENT_TYPE=<mime_type> LANGUAGE=<language_co
de> <filenames>
```

where

<apps/pwd>
is the APPS schema username/password. To specify a particular database, append an @ sign and the database SID (@database).

<application>
is the Application short name.

<custom_level>
is the files'customization level. Use the number 100 or above for customized help files. To replace previously uploaded files, use the same customization level when uploading the new files. To override previously uploaded files without deleting them from the database, use a higher customization level.

<mime_type>
is the files' MIME type.

<language_code>
is the files' language code.

<filenames>
is a space-separated list of files to upload, or a filename glob in the current directory.

Enter all arguments on a single command line. They may appear on separate lines here and in the examples that follow depending on the display medium.

Example 1
FNDGFU apps/apps@devdb 0 Y PROGRAM_NAME=FND_HELP PROGRAM_TAG=
GL:100 CONTENT_TYPE=text/html LANGUAGE=US file1.htm file2.htm

- connects to apps/apps@devdb
- identifies uploaded files as part of Oracle General Ledger (GL) help
- identifies the uploaded files' customization level as 100
- identifies their MIME type as text/html
- identifies their language as US English (US)
- uploads the two specified .htm files in the current directory (in UNIX)

Example 2
FNDGFU apps/apps@custdb 0 Y PROGRAM_NAME=FND_HELP PROGRAM_TAG=
FND:100 CONTENT_TYPE=image/gif *.gif

- connects to apps/apps@custdb
- identifies uploaded files as part of Application Object Library (FND) help
- identifies the uploaded files' customization level as 100
- identifies their MIME type as image/gif
- does not identify their language, which defaults to userenv('LANG')
- uploads all .gif files in the current directory (in UNIX)

Purging Generic File Manager Data

To purge uploaded files from the Generic File Manager, run the concurrent program, Purge Obsolete Generic File Manager Data.

This concurrent program should also be used to periodically expunge expired data. It is recommended that you schedule this program to run every day or so, using the default parameter values.

Purge Obsolete Generic File Manager Data

To purge uploaded files from the Generic File Manager, run the concurrent program, Purge Obsolete Generic File Manager Data.

This concurrent program should also be used to periodically delete obsolete data. It is recommended that you schedule this program to run every day or so, using the default parameter values.

Program Parameters

Expired

Enter "Y" if you want to purge expired data only. Enter "N" if you want the purge to include all data. The default is "Y."

Program Name

Enter the program name(s) to process. Leave blank to process all programs.

Program Tag

Enter the program tag(s) to process. Leave blank to process all program tags.