Customization Survival Guide: How to Use E-Business Utilities to Migrate Your Custom Code

Brad Simmons and Donna Campbell
Los Alamos National Laboratory

Abstract -- This paper examines a configuration management process using Oracle E-Business Suite provided utilities like FNDLOAD. We will briefly examine concepts for configuration, customization, and promotion processes. This paper will then review how to use the FNDLOAD and java XMLImporter and XDOLoader utilities. Along with example commands and sample unix scripts, we will review how to download and upload Oracle Application data using Oracle utilities for custom code promotion, in conjunction with the use of configuration management processes for code control, versioning, and release management.

Background -- Los Alamos National Laboratory (LANL) is a premier national security research institution, delivering scientific and engineering solutions for the nation's most crucial and complex problems. To ensure manufacturing program control, execution predictability, cost management, and safe and secure operations, LANL has deployed an integrated Manufacturing System that is based on Oracle's E-business Suite for Discrete Manufacturing.

Configuration Management Process -- Extensive customization or extension of Oracle's E-Business suite can result in an investment of thousands to millions of dollars on third-party configuration management tools to automate and track promotion of customizations from development through production. A small company may not have the expertise or the budget, for expensive configuration management tools which require add-on software, usually at additional cost, to automate the migration of Oracle Application setup and custom objects. Many

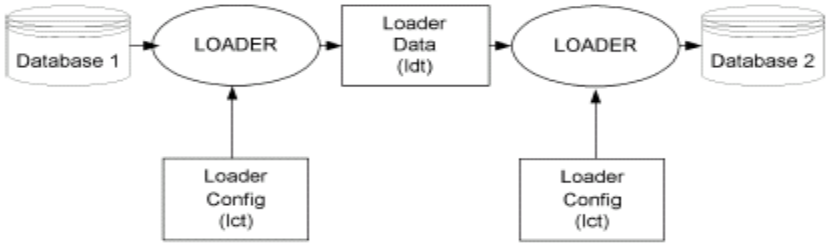
---

companies initially mandate that customizations not be allowed, yet over time, they find they have many customizations and no way to track, migrate or reapply them. In this customization environment, companies must use an effective configuration management process, with good code control and approval processes, in order to meet requirements for documenting, tracking, promoting, and implementing custom code. The configuration management process does not need to be complicated – it should be able to identify what versions of what customizations reside in which instances –typically development, quality / testing, and production. It encompasses promotion steps that allow a repeatable process of moving customizations through environments. It should provide a restore capability where promoted objects and code can be removed from an environment. Designing, planning and developing your company's configuration management processes by taking advantage of Oracle provided utilities, as outlined in this paper, can provide an effective, cost efficient, auditable code promotion capability.

Oracle Application Customizations -- In the case of the E-business Suite, the configuration management process must be able to manage customizations that range from simple custom reports to complex extensions. Examples include: OA framework pages, database packages, built-in customizations like descriptive flexfields, value sets, alerts, etc. Typical E-Business Suite customizations involve coordination of both technical and functional application-related steps. A developer will use Report Builder or XML Publisher to develop a report, or Forms Builder or JDeveloper to develop a screen, but Oracle Applications also requires that a report or screen be "registered" to Oracle Applications, which is a more functional, administrative task. For example, an Oracle customization may include a new custom form created using the Oracle provided template.fmb or modification of an Oracle application seeded form (.fmb) , or extensions to the newer Oracle Framework pages, a custom report (.rdf), or XML reports. After customization of the Application object, and depending on the type of customization, registration to Oracle Applications instance is usually completed manually using the System Administrator responsibility to register forms, functions, value sets, descriptive flex fields, concurrent programs, responsibilities, menus, request groups, messages, XML web pages, etc. Once this customization has been developed and tested, a code promotion process for the file system objects and the Oracle Application registrations is needed to promote it to other instances, e.g., from development to beta to production. Manually repeating this registration process in each instance is time consuming and error prone. Using the Oracle's provided utilities, like the FNDLOAD utility, it is possible to quickly download application-related data and upload it to the next instance. Define your promotion process to what best fits your company needs. You may need source code control at a very discrete level, or you may not need any source code control. Whether the objects and/or code is promoted manually by re-entering the application data, or by using FNDLOAD-type utilities, or using another configuration management tool, a consistent, repeatable and reliable promotion process needs to be followed.

FNDLOAD -- The FNDLOAD utility is documented in Oracle Applications 11.5.7, 11.5.8, 11.5.9 System Administration Guides, Appendix C. Starting with 11.5.10, the documentation has been expanded in the System Administration Guide – *Configuration* manual, Appendix B. The documentation for Release 12 has 3 new pages for Folders Configuration File. Metalink notes are useful when you encounter an error or are not sure of the correct syntax or format for a specific use of FNDLOAD. Other good sources of documentation are Oracle blogs which can easily be found by entering the FNDLOAD command or error in google.

"FNDLOAD is a Generic Loader" (oracle's definition) – in this paper we will refer to it as just FNDLOAD -- that can be used to download application data from an Oracle Application instance into a portable, editable text file(.ldt file). The data, in the FNDLOAD .ldt file can then be uploaded into another Oracle Application instance. This .ldt file contains not only the actual data representing the object(s) being downloaded, but it also contains metadata about the structure of that data. The following diagram, taken from the System Administrator's Guide - Configuration, Loaders in Appendix B or C (even this location varies by version), illustrates how FNDLOAD downloads data from an Oracle Application database according to a configuration (lct) file, and converts the data into a data file (ldt file) for subsequent uploading to another Application database.



Some of the terminology associated with FNDLOAD can get confusing. The utility is designed to download and upload FND types of objects, for example: menu definitions, concurrent program definitions, responsibility definitions, etc. These FND object types have also been called 'seed data types', 'entities', and simply 'objects'. Remember, each of these terms refers to the same thing – a type of Application object.

FNDLOAD 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. In both downloading and uploading, the structure of the data involved is described by the configuration file (lct file) and also the access methods to use to copy the data into or out of the database file. The same lct file is used for both uploading and downloading. Data structures supported by the loader include master-detail relationships and foreign key reference relationships. These configuration files are readable text and they often have additional comments and parameter usage information.

The FNDLOAD executable can be found in $FND_TOP/bin and associated lct files are located in $FND_TOP/admin/import/ or $FND_TOP/patch/115/import for the most current version. Remember, Oracle Application patches or upgrades can deliver new versions of the FNDLOAD executable and/or lct files, which can potentially change the behavior and/or parameters of the FNDLOAD program. Each FNDLOAD download execution will create .ldt, and .log files. Upon execution of the FNDLOAD program, the created log file needs to be checked to verify that data has been downloaded or uploaded correctly.

FNDLOAD can be executed as a Concurrent Program or as a UNIX command line utility with the following format and parameters, as documented in the Oracle Application System Administrator's Guide - Configuration:

FNDLOAD apps/pwd 0 Y mode configfile datafile entity [parameter....]
Where:

<apps/pwd>	Apps schema / password
< 0 Y >	Concurrent Program flags
mode	UPLOAD or DOWNLOAD
<config file>	Configuration lct file to use. See the FNDLOAD configuration table below for names of the lct files by seed data types.
<datafile>	Name of ldt data file written out when the FNDLOAD runs. This file is then used in an FNDLOAD upload to load the downloaded data into another application instance.
<entity>	Entities to upload or download. Specify a "-" to upload all entities on uploads.

<[param]> Zero or more additional parameters used to provide bind values in the access SQL. Each parameter is in the form NAME=Value (Look at the specific .lct file being used for more details for these parameters.)

Depending on the entity, FNDLOAD uses a different lct configuration file. The following table lists some of the most common entities that can be downloaded and/or uploaded and the associated FNDLOAD configuration (lct) file. Note that some lct file formats support multiple entity types:

Entity	**.lct file name**
Menus, Forms, Functions	afsload.lct
Responsibilities / Users	afscursp.lct
Profile Options	afscprof.lct
Concurrent Programs	afcpprog.lct
Request Groups	afcpreqg.lct
Request Sets	afcpreset.lct
Messages	afmdmsg.lct
Printer Styles	afcppstl.lct
Lookups	aflvmlu.lct
KeyFlexfields, Descriptive Flexfields, Value Sets	afffload.lct
Personalizations	affrmus.lct
Web ADI integrators, layers, mappings, content	bneint.lct, bnelay.lct, bnemap.lct, bnecont.lct
XML Templates	xdotmpl.lct
Folders (Release 12 only)	fndfold.lct

With knowledge of the FNDLOAD syntax and the appropriate configuration file, you can run FNDLOAD at the command line. The following are some sample 11.5.10 FNDLOAD commands, by entity, using the "DOWNLOAD" parameter. Uploads usually follow the same format only with the "UPLOAD" parameter. The FNDLOAD command is issued as 1 line. Some of the examples use literal values for all parameters, but other examples use unix variables as references to some parameter values. Please check the metalink notes referenced at the end of the paper for additional detailed examples.

MENU:

```
FNDLOAD apps/apps 0 Y DOWNLOAD $FND_TOP/patch/115/import/afsload.lct menu.ldt
MENU MENU_NAME=BOM_NAV
```

FUNCTION:

```
FNDLOAD apps/apps O Y DOWNLOAD $FND_TOP/patch/115/import/afsload.lct
function.ldt FUNCTION FUNCTION_NAME=INV_INVTTMTX
```

FORM:

```
FNDLOAD apps/apps O Y DOWNLOAD $FND_TOP/patch/115/import/afsload.lct form.ldt
FORM FORM_NAME=${shortname}
```

PERSONALIZATION:

```
FNDLOAD apps/apps O Y DOWNLOAD $FND_TOP/patch/115/import/affrmcus.lct
formpersonalization.ldt FND_FORM_CUSTOM_RULES function_name=${shortname}
```

PRINTER:

```
FNDLOAD apps/apps O Y DOWNLOAD $FND_TOP/patch/115/import/afcppstl.lct
printer.ldt STYLE PRINTER_STYLE_NAME=${shortname}
```

CONCURRENT:

```
FNDLOAD apps/apps 0 Y DOWNLOAD $FND_TOP/patch/115/import/afcpprog.lct
concprog.ldt PROGRAM CONCURRENT_PROGRAM_NAME=${shortname}
```

LOOKUP:

```
FNDLOAD apps/apps O Y DOWNLOAD $FND_TOP/patch/115/import/aflvmlu.lct
lookup.ldt FND_LOOKUP_TYPE APPLICATION_SHORT_NAME="FND"
LOOKUP_TYPE=${shortname}
```

RESPONSIBILITY:

```
FNDLOAD apps/apps O Y DOWNLOAD $FND_TOP/admin/import/afscursp.lct
responsible.ldt FND_RESPONSIBILITY RESP_KEY="${respkey}"
```

DESCFLEX:

```
FNDLOAD apps/apps O Y DOWNLOAD $FND_TOP/patch/115/import/afffload.lct
${shortname}_flex.ldt DESC_FLEX DESCRIPTIVE_FLEXFIELD_NAME=${shortname}
```

KEYFLEX:

```
FNDLOAD apps/$PASSWD O Y DOWNLOAD $FND_TOP/patch/115/import/afffload.lct
keyflex.ldt KEY_FLEX
P_LEVEL='COL_ALL:FQL_ALL:SQL_ALL:STR_ONE:WFP_ALL:SHA_ALL: CVR_ALL:SEG_ALL'
APPLICATION_SHORT_NAME=${appshortname} ID_FLEX_CODE=MCAT
P_STRUCTURE_CODE=${shortname}
```

VALUE:

```
FNDLOAD apps/apps O Y DOWNLOAD $FND_TOP/patch/115/import/afffload.lct
valset.ldt VALUE_SET FLEX_VALUE_SET_NAME=${shortname}
```

PROFILE:

```
FNDLOAD apps/apps O Y DOWNLOAD $FND_TOP/patch/115/import/afscprof.lct
profile.ldt PROFILE PROFILE_NAME=${shortname}
```

MESSAGE:

```
FNDLOAD apps/apps O Y DOWNLOAD $FND_TOP/patch/115/import/afmdmsg.lct
message.ldt FND_NEW_MESSAGES MESSAGE_NAME=${shortname}
```

REQUEST GROUP:

```
FNDLOAD apps/apps O Y DOWNLOAD $FND_TOP/patch/115/import/afcpreqg.lct
reqgrp.ldt REQUEST_GROUP REQUEST_GROUP_NAME=${shortname}
```

REQUEST SET:

```
FNDLOAD apps/apps 0 Y DOWNLOAD $FND_TOP/patch/115/import/afcprset.lct
rqstset.ldt REQ_SET APPLICATION_SHORT_NAME="FND"
REQUEST_SET_NAME=${shortname}
```

WEBADI:

```
FNDLOAD apps/apps 0 Y DOWNLOAD $BNE_TOP/admin/import/bnelay.lct
XX_C_O_F_T.ldt BNE_LAYOUTS LAYOUT_ASN="PER" LAYOUT_CODE="XX_C_O_F_T"
```

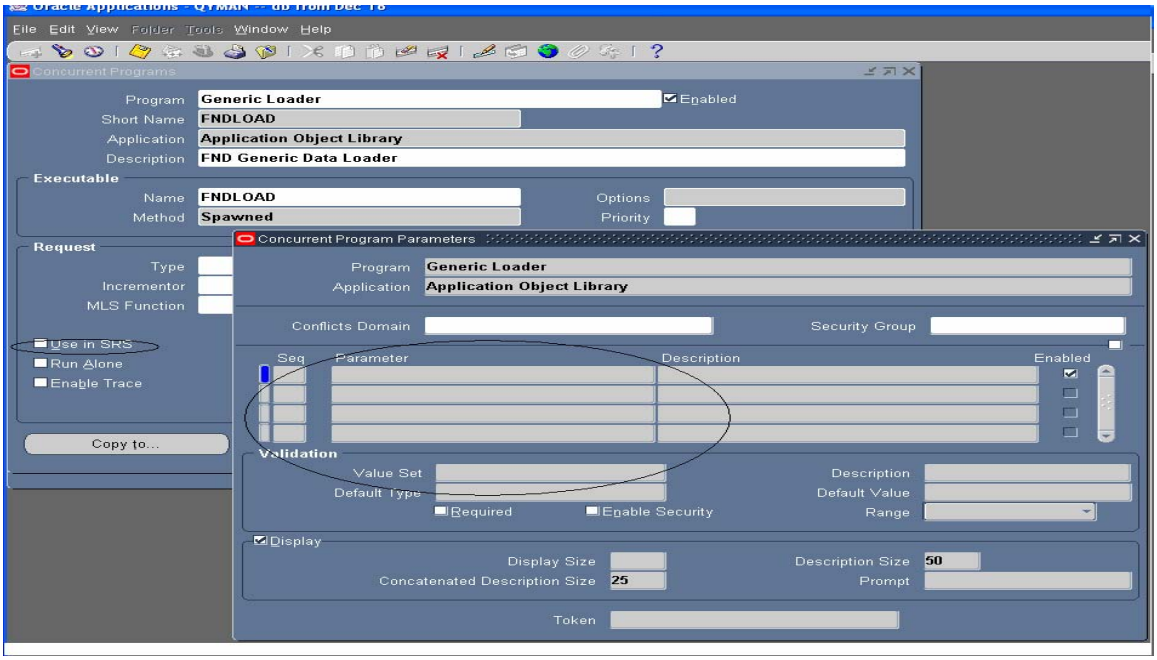
FNDLOAD: Don't shoot yourself in the foot! – Be careful to consider data preservation when using FNDLOAD in the upload mode. FNDLOAD may insert, update, or even delete rows in the underlying FND tables! When uploading and a row exists, but with different attributes, the row will be updated. If the row does not exist, a new row is inserted. Additionally, 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. Determining which key is used to identify records can be challenging. For example, PROFILE_OPTION_NAME, not PROFILE_OPTION_ID, is used identify records in the profiles configuration file. You can examine the configuration file to determine additional behavior. FNDLOAD also uses OWNER and LAST_UPDATE_DATE attributes to determine whether to overwrite pre-existing data. See the Sys Admin Guide – Configuration, Appendix B for more details.

FNDLOAD: Failures – Several different errors can occur when using FNDLOAD, so use diligence to review results and logs when using this utility. The FNDLOAD program can fail when the ldt file used to download and the corresponding lct file used to upload are not compatible. In later versions of the FNDLOAD program, configuration (lct) files have a heading line, LDRCONFIG = "afscprof.lct 115.xx", which shows the file version needed to successfully load the ldt file. Usually, you should use the same version of the lct file for the download and upload. Be aware, Oracle patches use FNDLOAD and will sometimes deliver a new version of an lct file with a patch. Another, "hidden", failure can occur when an FNDLOAD execution generates an ldt file with the data template information but no actual data, so it is important to review the log file to ensure a successful download or upload. ALWAYS check your log files, on downloads, and especially on uploads. Other errors can be encountered, like missing data dependencies, e.g., a responsibility upload will report errors in the log file if a referenced menu is missing. In this case, you must upload the menu before the responsibility. Finally, even with a successful upload, some entities may require some additional actions like recompiling menu structures, etc, via the adadmin utility.

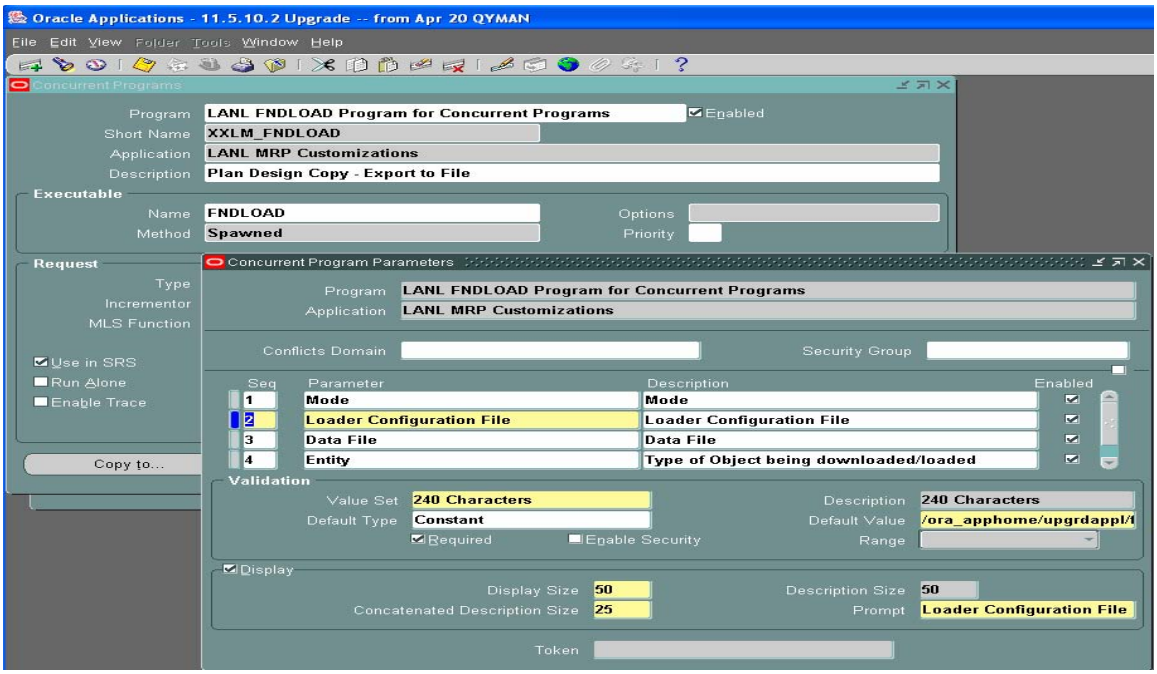
Note the warning in the Oracle Application System Administrator Guide – Configuration Release 12 Part No. B31453-01 Appendix B: "Warning: Use only the loader files provided by Oracle Applications. If you use files not provided by Oracle Applications or modify the provided scripts you risk corrupting your database. Oracle does not support the use of custom loader files or modified Oracle Applications loader files."

FNDLOAD: Great Tip! "Cloning" Objects -- FNDLOAD can also be used to easily copy application data, like responsibilities and menus. The key is in the .ldt file. It's editable. For example, an Oracle Application seeded menu structure can be downloaded, like the seeded top-level Inventory menu INV_NAVIGATE (all submenus will download too). You can edit the .ldt file to prefix all menu names with your site's prefix, e.g., LANL. Then run FNDLOAD upload and, *eureka*, you have a set of custom Inventory menus! This is a fast, easy, accurate way to create custom menus and other custom entities, eg, responsibilities, request groups, etc.

FNDLOAD: Concurrent Program – FNDLOAD can also be used as "traditional" concurrent program to move Oracle Application data between data file and text file representations. To see the FNDLOAD concurrent program definition, navigate to the Define Program screen under the System Administrator responsibility and you will find the registered spawned program but you *will not* find the program listed in the request groups making it available to run by the Submit Request form. The seeded Generic Loader program does *not* have the "use in SRS" checkbox checked and has *no* parameters defined in the Define Program parameters screen.



Seeded Generic Loader (FNDLOAD) has no parameters defined, Use in SRS is not checked



Custom Generic Loader (FNDLOAD) version, Parameters defined, Use in SRS checked

As with any seeded Oracle Application object, you would copy the seeded program definition to make a custom Generic Loader program (see above screen print). Then check the "Use in SRS" checkbox, define the FNDLOAD parameters, and add the concurrent program to a request group for a responsibility to make the program available to run from the Submit Request form in Oracle Applications. It will run the same as if running FNDLOAD from the Unix command line and will create the ldt data file in the directory you normally find your report output unless you include a different output directory path on the data file parameter. An advantage of running FNDLOAD as a

concurrent program is that your users do not have to be UNIX aware. In contrast, running FNDLOAD from a command line allows script integration with other configuration management components.

Other Oracle Utilities – Similar to FNDLOAD, other Oracle Development tools, like the XML Publisher and JDeveloper, also have Oracle Application utilities that can be used to download and/or upload application data. These tools are supported by the java XML and XDO Loader utilities, which can be used to migrate additional types of Application entities among your instances. Finally, AD administration tasks are sometimes needed to support post-upload requirements for some entities.

Developers can use the XML Publisher to create XML reports for Oracle Applications. In the Apps, navigate to the XML Publisher Administrator responsibility, then to Templates and Data Definition web pages. Use these web pages to register custom XML reports and upload your report's physical template file from your client, *or* you can use two command line utilities to automate moving XML report registration and the report's physical files among instances. FNDLOAD will download and upload XML report data definitions and template metadata. The XDOLoader utility is a Java-based command line program to download and/or upload the physical template (RTF, PDF, XSL-FO, XML, and XSD) files from/to the XML Publisher database tables. The following examples show the FNDLOAD and XDO Loader commands to download/upload XML report data:

Data definition and template registration:

```
FNDLOAD apps/apps O Y DOWNLOAD $XDO_TOP/patch/115/import/xdotmpl.lct tmpl.ldt
XDO_DS_DEFINITIONS APPLICATION_SHORT_NAME=custom application name
DATA_SOURCE_CODE=code name
```

Physical template files:

```
java oracle.apps.xdo.oa.util.XDOLoader DOWNLOAD -DB_USERNAME apps -
DB_PASSWORD xxxx  -JDBC_CONNECTION
'(DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=hostname)(PORT=1561))(CONNECT_DATA
=(SERVICE_NAME=SID)))' -LOB_TYPE TEMPLATE -APPS_SHORT_NAME XXLM -LANGUAGE en
-lct_FILE $XDO_TOP/patch/115/import/xdotmpl.lct
```

Developers can use JDeveloper to modify existing OA Framework pages or create new custom web pages. The web pages can then be downloaded and/or uploaded from one Oracle Application instance to another using the Java Exporter and/or Importer utility. OA Framework pages also have file system class and jar files which can be migrated by zipping your custom directories, using ftp to copy these files, and unzipping them into your other instance's $APPL_TOP before running the XML Importer.

The following is a sample Java XML Importer command to load xml pages:

XML PAGES:

```
java oracle.jrad.tools.xml.importer.XMLImporter
$XXLM_TOP/oracle/apps/xxlm/${shortname}/webui/ -username apps -password
$password -rootdir $XXLM_TOP -DBCONNECTION "hostname:dbport:sid" -jdk13 -
mmddir $OA_HTML/jrad
```

Once uploaded and prior to being used, some entities require AD administration tasks to be executed. For example, uploading messages requires the maintenance step of 'Generate Message Files'. Before AD administration can be used non-interactively, a defaults file must be created interactively. eg:

```
adadmin defaultsfile=$APPL_TOP/admin/testdb1/adadmindef.txt
```

Once created, this defaults file is used in the non-interactive call to adadmin:

```
adadmin interactive=n defaultsfile=$APPL_TOP/admin/testdb1/adadmindef.txt
```

For details, see “Oracle Applications Maintenance Procedures.”

Utility download script – Running FNDLOAD, and other similar utilities, from a UNIX script allows you to easily integrate the use of the download/upload utilities for your Oracle Applications data, e.g., registration of forms/reports/value sets, etc., for code promotion with the configuration management and promotion of your UNIX file system objects, e.g, your forms (fmb), reports (rdf), etc. Any execution of FNDLOAD assumes the Oracle Application environment has previously been set. The code segment below is one example of how to call FNDLOAD within a korn shell script.

```
##
## assumes APPS environment is properly set
##   and other shell variables previously defined
##
##
## Prompt for type and parameter value
##
echo "Enter a category
(MENU,FUNCTION,PERSONALIZATION,PRINTER,CONCURRENT,LOOKUP,RESPONSIBILITY,FORM,
DESCFLEX, KEYFLEX, VALUE, PROFILE, REQGROUP,
REQSET,MESSAGE,XML,XDO,IMPORTER): "
read category
if [ -z "$category" ] ; then
  exit
else
  typeset -u category
  echo "Using category" $category | tee -a ${LOGFILE}
fi

##
## determine appropriate user input prompt
case $category in
   'REQGROUP')
      parm_prompt="Please enter a request group."
   ;;
   'RESPONSIBILITY')
      parm_prompt="Please enter a responsibility key (include _ underscore
for spaces)."
   ;;
   *)
      parm_prompt="Enter a short name."
esac
## Get input from user
echo $parm_prompt
read shortname
if [ -z "$shortname" ] ; then
  echo $parm_prompt
  exit
fi
respkey=$shortname

## log files from FNDLOAD execution are not ‘well named’
## this is one attempt to identify a new file
```

```
##  Get initial list of *log files
ls *log > ${WORK_FILE1} 2>/dev/null


##
## execute appropriate form of FNDLOAD
##
case $category in
   'MENU')
     FNDLOAD apps/$PASSWD 0 Y DOWNLOAD $FND_TOP/patch/115/import/afsload.lct
${shortname}_menu.ldt MENU MENU_NAME=${shortname}
   ;;
   'PERSONALIZATION')
   FNDLOAD apps/$PASSWD 0 Y DOWNLOAD $FND_TOP/patch/115/import/affrmcus.lct
${shortname}_formpersonalization.ldt FND_FORM_CUSTOM_RULES
function_name=${shortname}
   ;;
   'XML')
     FNDLOAD apps/$PASSWD 0 Y DOWNLOAD $XDO_TOP/patch/115/import/xdotmpl.lct
${shortname}_tmpl.ldt XDO_DS_DEFINITIONS APPLICATION_SHORT_NAME=XXLM
DATA_SOURCE_CODE=${shortname}
   ;;
   'XDO')
java oracle.apps.xdo.oa.util.XDOLoader DOWNLOAD -DB_USERNAME apps -
DB_PASSWORD $PASSWD -JDBC_CONNECTION
'(DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=$EMACH)(PORT=$EPORT))(CONNECT_DATA
=(SERVICE_NAME=$TWO_TASK)))' -LOB_TYPE TEMPLATE_SOURCE -APPS_SHORT_NAME XXLM
-LOB_CODE ${shortname} -LANGUAGE en -TERRITORY US -XDO_FILE_TYPE RTF -
FILE_CONTENT_TYPE 'application/rtf' -FILE_NAME ${shortname}.rtf -NLS_LANG
ENGLISH_UNITED_STATES.WE8ISO8859P1  $XDO_TOP/patch/115/import/xdotmpl.lct
   ;;
##
## additional FNDLOAD command lines for each of the different
##     categories of applications entities
##
   *)
     echo
     echo You entered an incorrect category. Valid categories are:
     echo MENU, FUNCTION, PRINTER, CONCURRENT, LOOKUP, RESPONSIBILITY
DESCFLEX, KEYFLEX, VALUE, PROFILE, REQGROUP, REQSET, XML, XDO
   ;;
esac
sleep 1

## Additional work to attempt to identify the most recently
##   created log file
##
## get new list of log files
ls *log > ${WORK_FILE2}
## find name of newly created log file  (from concurrent mgr process)
FND_LOG=`diff ${WORK_FILE1} ${WORK_FILE2} | grep \> | cut -c3-`
echo "Logfile is:" ${FND_LOG} | tee -a ${LOGFILE}
## look for obvious success or failure tokens
grep "successfully" ${FND_LOG} | tee -a ${LOGFILE}
grep "ORA-" ${FND_LOG} | tee -a ${LOGFILE}
```

This code interactively prompts the user for the type and name of the entity that will be downloaded. It also uses the name ('shortname' parameter) to help label the created .ldt file along with its entity type. This naming convention helps to keep track of which ldt files contain which entities. An upload script for these files would look very similar. The script could be extended or used in conjunction with other scripts to implement a simple configuration process.

Using utilities within a configuration management process – A configuration management tool is used to ensure you have version control and rollback capability for all file system objects, eg., forms(fmb) , reports(rdf), as well as database objects, e.g., tables, indexes, packages, etc. The Oracle convention of using a $Header remark line in every object provides a basic version control that follows the object wherever delivered, as an operating system file, or in the database. Use of a configuration management tool will version an object on check-in and will save all versions of a particular object, providing rollback capability when needed. Use of the FNDLOAD utilities, in conjunction with a configuration management tool, gives you the ability to automate the version control, as well as the migration of the functional data needed for registering and promoting Oracle Application objects.

Simple Process **–** Changes are developed within one environment. This may consist of file system objects like sql scripts, or form .fmb files, etc., as well as new or changed application entities like a concurrent program definition, or a menu structure. These changes are documented and the entities composing the change are bundled into a 'promotion package'. Creating the promotion package can be as simple as creating a staging directory and copying all UNIX file system objects that need to be promoted into that directory, then running FNDLOAD to download application data for all associated objects into the same directory. This could be done by the developer of the change. The final stage directory can either be uniquely named, so it acts as its own historical reference, or it can be checked into a version control system. A typical configuration management activity, approval of the promotion package, can be done via email. Implementing the change consists of ftp'ing the staging directory to the location of your target instance, copying objects to the appropriate target $APPL_TOP directories and running an upload script. Another configuration management task, tracking where promotion packages have been applied, can be done via a spreadsheet (or even a table in a database – gasp!) For example, to promote a newly developed report, copy the rdf to a staging directory, then run the FNDLOAD command to download the Concurrent program definition and parameters for that report. You may also need to download the request group, menu, value sets, etc., associated with this report. All of downloaded files should be put in the same staging directory. To move this report to a new instance, the staging directory is copied to the new instance, then the rdf file is copied to the new instance's $APPL_TOP. Finally, an upload script is run for the Concurrent program definition, and any other related entities.

Summary -- This paper introduced some basic concepts for configuration management within the Oracle Applications E-Business suite. Customization and changes in this environment not only require typical objects like new sql code or new form / report files, but often require changes to Application entities like Concurrent programs, value sets, and menus. These entity changes can be done manually, but this process is error-prone and time consuming. To improve this process, we reviewed the functionality of the FNDLOAD utility as a tool to move entities between Applications instances. We described the technical details of using this utility, including syntax and parameters. Furthermore, we showed how to run the FNDLOAD utility from the UNIX command line and from within Oracle Applications. We then briefly mentioned other utilities that are available for downloading and uploading specific Oracle Application entities and provided examples of their use. To assist with understanding the practical implementation of these utilities, we provided an example of code that could be used. Finally, we reviewed how to use the Oracle utilities in conjunction within a configuration management process to promote code in an auditable, repeatable, efficient process, thereby saving you and your company many hours of time and effort.

References:

Note: 274667.1 FNDLOAD Commands to Download Different Seed Data Types.
Note: 287417.1 Parameters Of Different Configuration Files To Download And Upload Seed Data Types.

Web Blogs:
http://oracle.anilpassi.com/