

Oracle Tuxedo Application Rehosting Workbench

User Guide

11g Release 1 (11.1.1.1.0)

March 2010

ORACLE®

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

If this software or related documentation is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, the following notice is applicable:

U.S. GOVERNMENT RIGHTS Programs, software, databases, and related documentation and technical data delivered to U.S. Government customers are "commercial computer software" or "commercial technical data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, the use, duplication, disclosure, modification, and adaptation shall be subject to the restrictions and license terms set forth in the applicable Government contract, and, to the extent applicable by the terms of the Government contract, the additional rights set forth in FAR 52.227-19, Commercial Computer Software License (December 2007). Oracle USA, Inc., 500 Oracle Parkway, Redwood City, CA 94065.

This software is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications which may create a risk of personal injury. If you use this software in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure the safe use of this software. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software in dangerous applications.

Oracle is a registered trademark of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.

This software and documentation may provide access to or information on content, products and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services.

Contents:

1. Introduction

Oracle Tuxedo Application Rehosting Workbench	1-1
Sequence of the Rehosting Workbench processes	1-1

2. Oracle Tuxedo Application Rehosting Workbench Cataloger

Overview	2-1
Purpose	2-1
Skills	2-2
Migration process and concepts	2-2
UNIX/Linux skills	2-2
z/OS skills	2-2
See also	2-2
Organization	2-3
Requirements & Prerequisites	2-3
Preparing the migration platform	2-3
Overview of the Cataloger in the replatforming process	2-3
Simple Sample Application	2-3
Cataloging migration steps	2-3
Cataloging steps	2-4
Building-up an asset	2-4
Initialization and configuring the working environment	2-5
Configuration objectives	2-5

Recommended file structure	2-5
Setting environment and working variables	2-6
Project initialization summary	2-8
Configuration	2-8
System description file	2-9
Global options	2-10
Executing the Cataloger	2-10
One operation	2-10
Step by step	2-11
Parsing	2-11
Analysis	2-11
Print reports	2-12
Result analysis and validation	2-12
Inventory	2-13
Anomaly analysis	2-13
Completion criteria	2-14
Using the make utility	2-14
Make configuration	2-15
Version.mk	2-15
Makefile	2-16
Parsing	2-16
Parsing all programs	2-16
Check need to be parsed	2-16
Parsing of one program	2-16
Cataloging	2-16
Advanced usage of make	2-17
To add or update a target	2-17
Debugging the make target	2-17

Using the command with option VERIF=TRUE 2-17

3. Oracle Tuxedo Application Rehosting Workbench File-to-File Converter

Overview	3-1
Purpose	3-1
Skills	3-1
See also	3-1
Organization	3-1
The File-to-File Migration Process	3-2
File Organizations Processed	3-2
PDS file organization	3-3
GDG file organization	3-3
Migration Process Steps	3-3
Initializing the process	3-3
Requirements	3-4
Listing the files to be migrated	3-4
File descriptions and managing files with the same structure	3-4
COBOL description	3-4
COBOL description format	3-5
COBOL description and related discrimination rules	3-6
Redefinition examples	3-8
Non-equivalent redefinition	3-8
Equivalent redefinition called technical redefinition	3-9
Preparing the environment	3-11
Initializing environment variables	3-11
Implementing the configuration files	3-12
Configuring the files	3-12

Database Parameter File (db-param.cfg)	3-12
Datamap parameter file (Datamap-<configuration name>.re)	3-13
Mapping parameter file (mapper-<configuration name>.re)	3-15
Installing the copy files	3-17
Generating the components	3-17
file.sh	3-18
Locations of generated files	3-19
Modifying generated components	3-19
Using the make utility	3-19
Configuring a make file	3-20
Version.mk	3-20
Using a makefile with the Rehosting Workbench File-To-File Converter . .	3-21
Performing the migration	3-21
Preparation	3-21
Configuring the environments and installing the components	3-21
Installing the unloading components under z/OS	3-21
Installing the reloading components on target platform	3-21
Unloading JCL	3-22
Transferring the files	3-22
Compiling the transcoding programs	3-22
Sequential files	3-22
VSAM KSDS files	3-23
Executing the transcoding and reloading scripts	3-24
Checking the transfers	3-24
Troubleshooting	3-25
Overview	3-25
Common problems and solutions	3-25
Error: Unknown file organization *UNDEFINED*	3-25

Error: Record... not found	3-26
Error: Record... not found	3-26
Error: External Variable PARAM is not set	3-27

4. Oracle Tuxedo Application Rehosting Workbench DB2 to Oracle Converter

Overview	4-1
Purpose	4-1
Skills	4-1
See also	4-1
Organization	4-1
The DB2- to-Oracle Migration Process	4-2
File Organizations Processed	4-2
Migration Process Steps	4-2
Interaction with other Oracle Tuxedo Application Rehosting Workbench tools	4-3
Reengineering rules to implement	4-3
Migration rules applied	4-3
DB2-to-Oracle data type migration rules	4-4
DB2-to-Oracle column property migration rules	4-4
Preparing and implementing renaming rules	4-5
Example of a migration of DB2 objects	4-6
Preparing the environment	4-8
Implementing the cataloging of the DB2 DDL source files	4-8
system.desc file parameters	4-9
Schemas	4-9
Implementing the configuration files	4-9
Initializing environment variables	4-10
Generation parameters	4-10

Generating the components	4-11
rdbms.sh	4-11
Generation options	4-12
Modification options.....	4-12
Installation Option	4-13
Generate configuration files for COBOL conversion.....	4-13
Using the make utility.....	4-13
Configuring a make file	4-13
Version.mk	4-13
Using a makefile with the Rehosting Workbench DB2-To-Oracle Converter	4-14
Locations of generated files	4-15
Modifying generated components.....	4-16
Performing the migration	4-16
Preparation	4-16
Configuring the environments and installing the components	4-16
Installing the unloading components under z/OS.....	4-16
Installing the reloading components on the target platform.....	4-16
Unloading JCL.....	4-17
Transferring the files	4-18
Creating the Oracle objects.....	4-18
Compiling the transcoding programs	4-19
Executing the transcoding and reloading scripts.....	4-20
Transcoding and reloading command	4-20
Checking the transfers.....	4-21
Troubleshooting	4-21
Overview.....	4-21
Common problems and solutions	4-22
Error: RDBMS-0105	4-22

Error: conversion aborted. Can not read	4-22
Error: Configuration file /db-param.cfg is missing!.....	4-22
Error: Target output directory... is missing.....	4-23

5. Oracle Tuxedo Application Rehosting Workbench File-to-Oracle Converter

Overview.....	5-1
Purpose	5-1
Skills	5-1
See also	5-1
Organization	5-1
The File-to-Oracle migration process.....	5-2
File organizations processed	5-2
Migration Process Steps	5-2
Interaction with other Oracle Tuxedo Application Rehosting Workbench tools... ..	5-3
Initializing the process	5-3
Listing the files to be migrated	5-3
File descriptions and managing files with the same structure	5-3
COBOL description.....	5-4
COBOL description format	5-4
COBOL description and related discrimination rules	5-5
Redefinition examples	5-7
Non-equivalent redefinition	5-7
Equivalent redefinition called technical redefinition	5-8
Reengineering rules to implement	5-10
Migration rules applied	5-10
Rules applied to Picture clauses	5-11
Rules applied to Occurs and Redefines clauses.....	5-12

Example VSAM file migration to Oracle table	5-13
Preparing the environment	5-14
Initializing environment variables.	5-15
Implementing the configuration files	5-15
Configuring the files	5-15
Database Parameter File (db-param.cfg)	5-15
Datamap parameter file (Datamap-<configuration name>.re)	5-17
Mapping parameter file (mapper-<configuration name>.re)	5-18
Installing the copy files	5-19
Generating the components	5-20
file.sh	5-20
Using the make utility	5-21
Configuring a make file	5-21
Version.mk	5-21
Using a makefile with the Rehosting Workbench File-To-Oracle Converter	5-22
Locations of generated files	5-22
Examples of generated components.	5-23
Modifying generated components.	5-24
Performing the migration	5-24
Preparation	5-24
Configuring the environments and installing the components	5-24
Installing the unloading components under z/OS.	5-24
Installing the reloading components under UNIX	5-25
Installing the Oracle object creation components.	5-25
Setting the target platform environment variables	5-25
Unloading the JCL	5-25
Transferring the files	5-26
Compiling the transcoding programs	5-26

Executing the Oracle object creation scripts	5-26
Executing the transcoding and reloading scripts	5-26
Checking the transfers	5-27
Compiling the access routines and utilities	5-27
Troubleshooting	5-28
Overview	5-28
Common problems and solutions	5-28
Error: External Variable PARAM is not set	5-28
Error: Target directory does not exist	5-28
Error: Unknown file organization	5-29
Error: The illegal text is: "converted"	5-29
Error: Can't find file in file table named PJ01AAA.SS.VSAM.CUSTOMERS	5-31

6. the Rehosting Workbench Cobol Converter

Overview	6-1
Purpose	6-1
Skills	6-1
See also	6-1
Organization	6-2
Requirements & Prerequisites	6-2
Cataloguing requirements	6-2
Data Conversion	6-2
Overview of the Cobol Converter in the replatforming process	6-2
Conversion steps	6-3
Building and setting the configuration files	6-3
Configuring config-cobol	6-3
keywords-file	6-4

tr-hexa.map	6-5
rename-call-map-file	6-6
Conversion	6-7
Translation of Batch, CICS programs and sub-programs	6-7
Reconciling copybooks	6-8
Checking results	6-9
Compiling	6-9
Compilation options and settings	6-10
Compilation command	6-11
Using a make file	6-12
Configuration	6-12
Cobol conversion	6-14
Reconciling copybooks	6-14
Troubleshooting the Cobol Converter	6-14
Configuration file does not exist	6-14
Message	6-14
Solution	6-15
Missing POB file	6-15
Message	6-15
Solution	6-15
POB file too old	6-15
Message	6-15
Solution	6-16
Parsing error encountered	6-16
Message	6-16
Solution	6-16

7. Oracle Tuxedo Application Rehosting Workbench JCL Converter

Overview	7-1
Purpose	7-1
Skills	7-1
See also	7-1
Organization	7-2
Requirements & Prerequisites	7-2
Cataloguing requirements	7-2
Data Conversion	7-2
Overview of the JCL Translator in the replatforming process	7-2
Translation steps	7-3
Building and setting the configuration files	7-3
Top skeleton and Bottom skeleton	7-4
List of data files converted to Oracle tables	7-5
Full example configuration	7-5
Full example configuration	7-7
Translation steps	7-10
Set environment variables	7-10
Translation commands	7-10
Command line to translate one JCL file	7-10
Command line to translate a list of JCL files	7-11
Command line to translate all JCL files	7-11
Checking results	7-11
Using a make file	7-12
Configuration	7-12
JCL translation	7-12

Introduction

Oracle Tuxedo Application Rehosting Workbench

Oracle Tuxedo Application Rehosting Workbench is part of a packaged and comprehensive solution that enables its users:

- To perform a replatforming project with minimum risk and cost;
- To run the replatformed applications in a standardized UNIX/Linux, Tuxedo, Oracle environment.

The Oracle Tuxedo Application Rehosting Workbench that is used only during the replatforming project itself is composed of several tools, the principal tools are the:

- Cataloger,
- DB2-to-Oracle convertor
- File-to-Oracle converter,
- File-to-File converter
- Cobol convertor
- JCL translator.

Sequence of the Rehosting Workbench processes

The tools should be used in the following sequence as shown in the following figure:

1. Catalog the asset

2. Convert data

Note: Cataloguing is a prerequisite to the data conversion process, whether it be file conversion or database conversion.

3. Translate language components

When converting data from VSAM files to Oracle tables, File-to-Oracle data conversion is a prerequisite to JCL and Cobol translation

When converting data from DB2 databases to Oracle data conversion, DB2-to-Oracle data conversion is a prerequisite to Cobol translation.

Oracle Tuxedo Application Rehosting Workbench Cataloger

Overview

Purpose

This chapter aims to:

- Provide information and instructions about how to use the the Rehosting Workbench Cataloger.
- Help you to configure and run the Rehosting Workbench Cataloger tools.
- Show how to verify the results by interpreting messages and status information.

This guide includes information about the following:

- How to use the Rehosting Workbench Cataloger within the whole migration process.
- Best practices on how to prepare assets to be migrated, organize work spaces and effectively use the Rehosting Workbench tools.

Note: These practices and methodologies are not part of the the Rehosting Workbench product but are presented in this guide to help new users. You can proceed otherwise, customize the Rehosting Workbench in your own way and develop your own methods.

Skills

Migration process and concepts

You should understand the processes, concepts and terminology used by the Rehosting Workbench Cataloger; know the inputs, outputs and configuration expected by the Cataloger and how it analyzes all the components separately and together to determine whether the asset is consistent and can be migrated.

The Oracle Tuxedo Application Rehosting Workbench Reference Guide describes precisely all features of the Cataloger. Read at least the three first sections of the Cataloger chapter for an introduction to the concepts, terminology, inputs and outputs, and configuration.

UNIX/Linux skills

UNIX/Linux skills are required to work correctly on the migration platform environment and perform certain actions accompanying the cataloguing process. You need to know:

- The standard Unix/Linux commands to create, modify, display and delete configuration files and output files.
- The standard Unix/Linux commands and specialized scripts (shells, makefile, cvs) to store and manage the original z/OS asset before and after cataloging.
- A set of commands (shell, makefile) to start, monitor and terminate the execution of the Rehosting Workbench Cataloger.

z/OS skills

You should be able to identify and understand z/OS components and programming languages. General skills in z/OS environment (COBOL, files, DB2, CICS, JCL, utilities) are sufficient.

See also

For more information, see:

- Oracle Tuxedo Application Rehosting Workbench Reference manual
- Oracle Tuxedo Application Runtime Process Guide.

Organization

The Oracle Tuxedo Application Rehosting Workbench Cataloger is described in the following sections:

- [Requirements & Prerequisites](#)
- [Overview of the Cataloger in the replatforming process](#)
- [Cataloging steps](#)
- [Using the make utility](#)

Requirements & Prerequisites

Preparing the migration platform

The migration platform is the platform on which the migration tools (the Rehosting Workbench) execute, including the Cataloger. This platform is based on Linux running on an Intel-compatible hardware platform.

Before performing any action, the Rehosting Workbench should be installed and configured according to specifications and requirements detailed in the Oracle Tuxedo Application Rehosting Workbench Installation Guide.

Overview of the Cataloger in the replatforming process

This section describes the inputs and outputs of the Cataloging step and dependencies with other migration steps in the replatforming process.

Simple Sample Application

In order to illustrate all of the migration activities performed and how to use the the Rehosting Workbench Cataloger to determine whether the asset is consistent and can be migrated to the target platform, the Simple Application `STFILEORA` will be used. `STFILEORA` is provided with the Rehosting Workbench set of tools.

Cataloging migration steps

Description of the cataloging operations shown in the graphic:

1. Read the configuration files and the description of the source files prepared for cataloging

2. Parse each of the components in the source asset and create an internal representation called packed object base (with extension .pob). Identify internal inconsistencies in these components (e.g. syntax error or reference to an undeclared variable);
3. Compute cross-reference relationships between these components, such as a CALL statement from a Cobol program to a Cobol sub-program;
4. Identify mutual inconsistencies in these relationships, such as a CALL statement referencing a sub-program missing from the asset, or an SQL table referenced by none of the Cobol programs in the asset (unused component);
5. Create CSV-format reports listing all the components in the asset, together with their status (missing, unused or correct) and all the anomalies (inconsistencies) detected above.

The operations described above are explained in more detail in the next sections.

Cataloging steps

Building-up an asset

This step is a pre-requisite for the Cataloger and, as mentioned in the Oracle Tuxedo Application Process Guide that gives an overview of the whole migration process, it is up to the user of the Rehosting Workbench to gather these source files on the source platform, transfer them to the migration platform, install them in an appropriate file structure and prepare them for migration.

Building up the asset consists of several steps from getting sources to organizing them to be available as a valid input to the Cataloger tool:

1. Gathering sources to be migrated from z/OS
2. Transfer the sources to the Rehosting Workbench platform:
 - a. The source files must be converted to the ASCII code-set in order to be understood on the migration platform. This conversion can be made either by using the file transfer utility or by applying a separate EBCDIC/ASCII conversion step.
 - b. You have to be sure that source content is not corrupted due to transfer procedures.
3. Dispatch and prepare sources by types as they are expected by the Cataloguer
 - a. Put all components in UNIX format (using dos2unix command for example)

- b. Put the Cobol components in free format as it will be requested in that format by the Cobol Converter. The Cataloger itself accepts both formats, you have only to configure the `cobol-left-margin` and `cobol-right-margin` options.
 - c. Organize components by type: put all CICS programs in one directory that could contain sub-directories, the same applies for Cobol includes, Batch programs, Jobs, etc.
 - d. Add file extensions to each type: `.cbl` for Cobol programs, `.cpy` for Cobol includes and `.jcl` for Jobs
4. Some other actions may be required: such as clean components from certain characters or lines added after the extraction from z/OS or during the transfer.

Initialization and configuring the working environment

As the the Rehosting Workbench Cataloger is the first tool of the Oracle Tuxedo Application Workbench suite used in the migration process, a general configuration step for the whole project to be performed is explained here.

Configuration objectives

- Reduce the time needed to prepare the working environment and avoid further possible errors due to configuration.
- Initialize the configuration for other WorkBench tools not only for the Cataloger.
- Recommend a standard organization to facilitate team work.

Recommended file structure

The more organized and standardized the working space, the more migration tasks will be easier and automated.

In the following sample we illustrate a typical organization and we recommend working with the same structure.

Listing 2-1 The hierarchy of Sample Application

```
SampleApp
|-- CVS_DELIVERY
|-- Logs
```

```
|-- param
|   `-- system.desc
|-- source
|   `-- makefile
|-- tools
```

The contents of each directory are:

- Source

Contains all sources to be migrated and prepared as input for the Rehosting Workbench tools.

Workspace where all the Rehosting Workbench processes can find the source code.

- param

Contains all configuration files (parameters and hints for use by the Rehosting Workbench tools).

- Logs

Working directory where all the Rehosting Workbench tools are launched, all log files are generated

- tools

Directory to place specific tools developed by the user during the migration process

Setting environment and working variables

It is recommended that each time you work on a project using the Rehosting Workbench to set certain environment variables that will be useful later. The only environment variable that is mandatory is `REFINEDISTRIB`; others can be used for simplification reasons.

The following table explains the usage of proposed variables.

Table 2-1

Variable	Value	Usage
REFINEDISTRIB	Linux64, or Linux32	Specifies the architecture of the the Rehosting Workbench binary files.
PROJECT	\${HOME}	Project root within the workspace. HOME is used for training. But value should be something like /project/SampleApp
CVSROOT	\${PROJECT}/CVS_DELIVRY	CVS repository to use containing the source and configuration files.
LOGS	\${PROJECT}/Logs	Structure where tthe Rehosting Workbench logs are stored
PARAM	\${PROJECT}/param	Directory where the parameter files are stored
SOURCE	\${PROJECT}/source	Structure where the source files to translate are stored
REFINEDIR	/ <InstallDir> /refine	Root of the Rehosting Workbench binaries and utilities
PHOENIX	\${REFINEDIR}	Variable used by the Rehosting Workbench
TMPPROJECT	\${PROJECT}/tmp	Temporary structure for R4Z the Rehosting Workbench
PATH	\${PATH}:/platform/Tools	PATH complement

In the Simple Application example project, we use a setting file named \$PROJECT/.project that contains all initializations using an export Linux command. This file is to be executed each time a new Linux session is opened to work in the project

Listing 2-2 Extract from Simple Application .project file:

```
echo "Wellcome to SampleApp"
export GROUP=refine
```

```
export PROJECT=${HOME}/SampleApp
export CVS_LIV=${PROJECT}/CVS_DELIVERY
export CVSROOT=${PROJECT}/CVS_DELIVERY
export LOGS=${PROJECT}/Logs
export SOURCE=${PROJECT}/source
export PARAM=${PROJECT}/param
export REFINEDIR=/product/art_wb1lgR1/refine
export PHOENIX=${REFINEDIR}
export TMPPROJECT=${PROJECT}/tmp
export REFINEDISTRIB=Linux64
```

Project initialization summary

To initialize a new project, proceed as follow:

1. Create a file structure as described above.
2. Copy a sample of the configuration (system.desc, version.mk, options-catalog, etc) files from Simple Application to `$PROJECT/param` directory to make any necessary modifications.
3. Copy the makefile from the source (SimpleApp) into `$PROJECT/source`.
4. Create a file `.project` under `$PROJECT` and initialize it with variables listed in [Setting environment and working variables](#). Variables in the file are to be exported each time you work on the project.
5. Copy the prepared source files to `$PROJECT/source`
6. Optional: Manage the source and configuration files under CVS.

Configuration

You need to set at least two configuration files, additional configuration files are described in the advanced usage section.

The two configuration files needed by the Cataloger are:

- System description file
- General option file

System description file

The system description file is the main configuration file for the Rehosting Workbench tools.

For the Simple Application example:

- The system-name is SampleApp
- The path where sources to be migrated are stored is `../source`
- To inform the Cataloger of the location of the global options file you can use relative or absolute naming but it is recommended to use an absolute path when you have multiple source environments to migrate.

```
global-options catalog = "../param/options-catalog.desc".
```

- Then add entries for each directory containing components to be catalogued.

Listing 2-3 system description for Simple Application

```
system SampleApp root "../source"

global-options
    catalog="../param/options-catalog.desc",
    no-END-Xxx.

DBMS-VERSION="8".

% Copies

directory "COPY" type Cobol-Library files "*.cpy".

% DDL

directory "DDL" type SQL-SCRIPT files "*.ddl".

% Batch

directory "BATCH" type Cobol-Batch files "*.cbl" libraries "COPY". %,
"INCLUDE".

% Cics Cobol Tp
```

```
%  
directory "CICS" type Cobol-TPR files "*.cbl" libraries "COPY". %,  
"INCLUDE".
```

Global options

The purpose of the Cataloger options file is to give the Cataloger additional information that will influence its behavior.

In the Simple Application example we use only three options, of course other options can be used; see the Oracle Tuxedo Application Rehosting Workbench Reference Guide for a full list of options.

Listing 2-4 Global Options file for the Simple Application

```
%% Options for cataloging the system  
job-card-optional.
```

Executing the Cataloger

One operation

You can launch the Cataloging with one command; all operations are performed sequentially.

The example command line syntax is:

```
${REFINEDIR}/refine r4z-catalog -s $PARAM/system.desc
```

Where:

- \${REFINEDIR} is the directory where the Rehosting Workbench tools are installed.
- \$(CATALOG) is the version of the Cataloger to be used
- \$(SYSTEM) is the path of the system description file.

In our sample the command will be:

From directory \$LOGS/catalog execute the command:

```
{REFINEDIR}/refine r4z-catalog -s $PARAM/system.desc
```

The execution log is printed on the screen and at the same is redirected in log file under the directory from which you lunched the command.

Step by step

Parsing

Running this step is useful when you want to check specific programs without waiting for the whole cataloging to see the results.

Listing 2-5 Parsing examples

```
# parse only one program

{REFINEDIR}/refine r4z-preparse-files -s $PARAM/system.desc
CICS/PGMM000.cbl

# parse a list of programs

# build a list that contains programs with path from source
(CICS/PGMM000.cbl)

{REFINEDIR}/refine r4z-preparse-files -s $PARAM/system.desc -f
list-of-file
```

Analysis

The result of this step is the binary file named symtab-SampleApp.pob that represents inter-component information.

Listing 2-6 Analysis example

```
cd $LOGDIR/catalog

{REFINEDIR}/refine r4z-analyze -s $PARAM/system.desc
```

Print reports

In this step information stored in binary files is collected and printed in CSV format reports.

Listing 2-7 Print Reports example

```
cd $LOGDIR/catalog
$REFINEDIR/refine r4z-fast-final -v M2_L3_3 -s $PARAM/system.desc
```

In the Simple Application reports are generated in \$PROJECT/source/Reports-SampleApp:

- report-SAMPLEAPP-Anomalies
- report-SAMPLEAPP-Cobol-Copy
- report-SAMPLEAPP-Cobol-Programs
- report-SAMPLEAPP-JCL-Files
- report-SAMPLEAPP-JCL-Jobs
- report-SAMPLEAPP-JCL-Sub-Files
- report-SAMPLEAPP-SQL-Tables
- report-SAMPLEAPP-SQL-Views
- report-SAMPLEAPP-Transactions

The different reports are described in the Oracle Tuxedo Application Rehosting Workbench Reference guide.

Result analysis and validation

Using the cataloging reports, you can perform different actions in order to create the exact asset set to be migrated and give launch the other the Rehosting Workbench steps: Cobol conversion, JCL translation and Data conversion.

Note: This does not mean that cataloging and conversion activities are strictly sequential, they may overlap.

Inventory

After cataloging each Program, Copy, Job, Include is given a status depending on its use.

Table 2-2 Inventory status

Status	Description
CORRECT	Components are present in source and used: <ul style="list-style-type: none">• Copybooks used at least by one program,• Programs used at least by one transaction, by at least one jcl or by at least one program• All jobs are considered as used.• JCL sub-file use at least by one Job. Action: Accept it
UNUSED	Component present in source but not used: <ul style="list-style-type: none">• Copybooks that are not called by a program• Programs that are not called by a transaction, or by a Job and that are not called as a sub-program by a program,• JCL sub-files that are not used by a Job Action: Decide whether to keep the component or to delete it from asset
MISSING	Component is not present in the asset but at least one reference to it has been found: <ul style="list-style-type: none">• Copybooks used at least by one program,• Programs used at least by one transaction, by at least one JCL or by at least one program.• JCL sub-files used by at least one Job Action: Decide whether to add missing components or delete calling components that could be unused or unnecessary.

Anomaly analysis

All the errors reported in the Anomalies Report must be analyzed:

- FATAL and ERROR severity level anomalies make the component or asset unsuitable for conversion, they must be analyzed and fixed.
- NOTICE and WARNING severity level anomalies should also be analyzed and they can be kept if there is no impact on the conversion.

The asset should be free from severe errors before proceeding to conversion.

Completion criteria

Cataloging can be considered complete once all expected outputs are generated (pob files and cataloging reports).

As a process, it depends on the context of the project, but generally cataloging is considered as completed when:

- All Unused and Missing components are either fixed or justified.
- All errors are fixed or justified.

Using the make utility

`make` is a UNIX utility intended to automate and optimize the construction of targets (files or actions).

We highly recommend using `make` to perform the different operations that compose the migration process because this enables you to:

- Have all (executable) processes documented.
- Effectively manage incremental operations and hence save time.
- Control and check the results of processes.

You should have a descriptor file named `makefile` in the source directory in which all operations are implemented (a `makefile` is prepared in the source directory during the initialization of a project).

The following two sections describe the `make` configuration and how to use the Cataloger functions thru `make`.

Make configuration

Version.mk

The version.mk configuration file in \$PARAM is used to set the variables and parameters required by the make utility.

In this file specify where each type of component is installed and their extensions, and the versions of the different tools to be used. This file also describes how the log files are organized.

Listing 2-8 Extract from version.mj for Simple Application

```
Root = ${PROJECT}

#

# Define directory Projet

#

Find_Jcl = JCL
Find_Prg = BATCH
Find_Tpr = CICS
Find_Spg =
Find_Map = MAP
SCHEMAS = AV

...

#Logs organisation

#

LOGDIR           := $(LOGS)
CATALDIR         := $(LOGS)/catalog
PARSEDIR         := $(LOGS)/parse
TRADJCLDIR      := $(LOGS)/trans-jcl
TRADDIR         := $(LOGS)/trans-cbl
DATADIR         := $(LOGS)/data
```

...

Makefile

The contents of the `makefile` summarize the tasks to be performed:

- Including `vesion.mk`
- Controlling if the main variables are set
- Collecting the value of variables and building the required lists to process.
- Implementing the targets.

The makefile provided with the Simple Application is auto-documented.

Parsing

Parsing all programs

From the `$$SOURCE` directory execute the command:

```
> make pob
```

The log file is generated in `$$LOGS/parse`

Check need to be parsed

```
> make pob VERIF=TRUE
```

```
Check : Parse of BATCH/PGMMB02.cbl Need Process .. To obtain  
BATCH/pob/PGMMB02.cbl.pob
```

Parsing of one program

Sample: parsing the program `BATCH/PGMMB02.cbl`

```
> make BATCH/pob/PGMMB02.cbl.pob
```

The log file is generated in `$$LOGS/parse`

Cataloging

To launch cataloging use:

```
make catalog
```

The log file is generated in \$LOGS/catalog

Advanced usage of make

To add or update a target

You can update the makefile to add some targets or to update existing ones.

For example, if you developed your own script (`report.sh`) to generate a customized report based on basic cataloging reports, place your script in \$TOOLS directory.

The makefile can be modified as follow:

- To add a new target to execute the new script:

```
Reporting:
@${TOOLS}/report.sh
```

- To update existing catalog target

```
catalog:
    @$(REAL_CMD)  ${REFINEDIR}/refine r4z-catalog -v $(CATALOG) -s
    $(SYSTEM)  $(LOG_FILE_FLAGS_CAT)
@make reporting
```

Your customized report will be updated automatically after each catalog execution.

Debugging the make target

Sometimes a command thru make cannot work properly because of a lack of configuration.

Proceed as follow:

1. Copy the used makefile to debug copy (makfile.debug), to not disturb current operations.
2. Modify makefile.debug changing REAL_CMD to COMMENT

```
catalog:
    @$(COMMENT)  ${REFINEDIR}/refine r4z-catalog -v $(CATALOG) -s
    $(SYSTEM)  $(LOG_FILE_FLAGS_CAT)
```

Using the command with option VERIF=TRUE

```
> make catalog VERIF=TRUE -f makefile.debug
```

The command to be executed is printed = after replacement of all variables.

Oracle Tuxedo Application Rehosting Workbench File-to-File Converter

Overview

Purpose

This chapter describes how to install, implement, and configure the Rehosting Workbench File-to-File converter in order to migrate files from the source z/OS environment to the target environment.

Skills

When migrating files, a good knowledge of COBOL, JCL, z/OS utilities and UNIX/Linux Korn Shell is required.

See also

For a comprehensive view of the migration process, see Oracle Tuxedo Application Rehosting Workbench Reference Guide for the chapters Data Conversion and Cobol Conversion as well as the [the Rehosting Workbench Cobol Converter](#) chapter of this guide.

Organization

Migrating data files is described in the following sections:

- [The File-to-File Migration Process](#)

- [Initializing the process](#)
- [Preparing the environment](#)
- [Generating the components](#)
- [Performing the migration](#)
- [Troubleshooting](#)

The File-to-File Migration Process

File Organizations Processed

When migrating from a z/OS source platform to a target platform, the first question to ask, when VSAM is concerned, is whether to keep a file or migrate the data to an Oracle table.

The Oracle Tuxedo Application Rehosting Workbench File-to-File converter is used for those files that keep their source platform format (sequential, relative or indexed files) on the target platform. On the target platform, these files use a Micro Focus COBOL file organization equivalent to the one on the source platform.

The following table lists the file organizations handled by z/OS and indicates the organization proposed on the target platform:

Table 3-1 z/OS to UNIX file organizations

z/OS source file	UNIX ISAM target file
QSAM	Line sequential ISAM
VSAM KSDS	Indexed ISAM
VSAM RRDS	Relative ISAM
VSAM ESDS	Line sequential ISAM

Note: The conversion of z/OS files to ISAM UNIX files has no direct link to the conversion of Cobol application programs or the translation of JCL.

PDS file organization

Files that are part of a PDS are identified as such by their physical file name, for example:
`META00.NIV1.ESSAI(FIC).`

An unloading JCL adapted to PDS is generated in this case. The source and target file organizations as indicated in the above table are applied.

GDG file organization

Generation Data Group (GDG) files are handled specially by the unloading and reloading components in order to maintain their specificity (number of GDG archives to unload and reload). They are subsequently managed as generation files by Oracle Tuxedo Application Runtime Batch (see the Oracle Tuxedo Application Runtime Batch Reference Guide). On the target platform these files have a LINE SEQUENTIAL organization.

Migration Process Steps

The principle steps in the File-to-File migration process, explained in detail in the rest of this chapter, are:

1. Create an overall list of the files to be migrated.
2. List those files that will not be converted to Oracle tables.
3. For each of these files, create a file description.
4. When necessary, create a list of discrimination rules.
5. Prepare the environment to be used to generate the components.
6. Using the Rehosting Workbench generate the components used in the following steps.
7. Unload the data from the source platform.
8. Transfer the data to the target platform.
9. Transcode and reload the data.
10. Check the results.

Initializing the process

This section describes the steps to be performed before starting the file migration.

Requirements

The migration of z/OS files to UNIX/Linux is dependant on the results of the [Oracle Tuxedo Application Rehosting Workbench Cataloger](#). It does not have any impact on the conversion of COBOL components or the translation of JCL components.

Listing the files to be migrated

The first task is to list all of the files to be migrated, for example, the permanent files input to processing units that do not come from an Oracle table.

File descriptions and managing files with the same structure

For each candidate file for migration, its structure should be described in Cobol format. This description is used in a Cobol copy by the the Rehosting Workbench Cobol converter, subject to the limitations described in [COBOL description](#).

Once built, the list of files to migrate can be purged of files with the same structure in order to save work when migrating the files by limiting the number of programs required to transcode and reload data.

Using the purged list of files, a last task consists of building the files:

- `Datamap-<datamap name>.re`
- `mapper-<mapper name>.re`

COBOL description

A COBOL description is related to each file and considered as the representative COBOL description used within the application programs. This description can be a complex COBOL structure using all COBOL data types, including the OCCURS and REDEFINES notions.

This COBOL description will often be more developed than the COBOL file description (FD). For example, an FD field can be described as a PIC X(364) but really contain a three times defined area including, in one case a COMP-3 based numerals table, and in another case a complex description of several characters/digits fields etc.

It is this developed COBOL description which describes the application reality and therefore is used as a base to migrate a specific physical file.

The quality of the file processing execution depends on the quality of this COBOL description. From this point, the COBOL description is not separable from the file and when referring to the file concerned, we mean both the file and its representative COBOL description. The description must be provided in COBOL format, in a file with the following name:

`<COPY name>.cpy`

Note: If a copy book on the source platform provides a detailed description of the file, the copy can be directly used and declared in the Rehosting Workbench.

COBOL description format

The format of the COBOL description must conform to the following rules:

- Only one level 01.
- The word FILLER is not allowed.
- Fields names must be unique.
- Some words are reserved, a list is supplied in the Appendix of the Rehosting Workbench Reference Guide.
- The description should begin in column 1 without any preceding COBOL sequence numbers.
- Comments may be inserted by placing an * in column 1.
- Field level numbers can start from column 2.

Example

Column																									
1	2	3	4	5	6	7	8	9	1	1	1	1	1	1	1	1	1	1	2	2	2	2	2	2	.
									0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	.
*	D	E	S	C	R	I	P	T	I	O	N		O	F		F	I	L	E		X	X	X	X	
	0	1		F	V	I	4	.																	
				0	5		F	V	I	4	x	1		P	I	C		X	.						

COBOL description and related discrimination rules

Within a COBOL description there are several different ways to describe the same memory field, which means to store objects with different structures and descriptions at the same place.

As the same memory field can contain objects with different descriptions, to be able to read the file, we need a mechanism to determine the description to use in order to interpret correctly this data area.

We need a rule which, according to some criteria, generally the content of one or more fields of the record, will enable us to determine (discriminate) the description to use for reading the re-defined area.

In the Rehosting Workbench this rule is called a discrimination rule.

Any redefinition inside a COBOL description lacking discrimination rules presents a major risk during the file transcoding. Therefore, any non-equivalent redefined field requests a discrimination rule. On the other hand, any equivalent redefinition (called technical redefinition) must be subject to a cleansing within the COBOL description (see the example below).

The discrimination rules must be presented per file and highlight the differences and discriminated areas. Regarding the files, it is impossible to reference a field external to the file description.

The following description is a sample of a COPY as expected by the Rehosting Workbench:

Listing 3-1 Cobol COPY sample

```
01 FV14.
    05  FV14-X1      PIC X.
    05  FV14-X2      PIC XXX.
    05  FV14-X3.
        10 FV14-MTMGFA      PIC 9(2).
        10 FV14-NMASMG      PIC X(2).
        10 FV14-FILLER      PIC X(12).
        10 FV14-COINFA      PIC 9(6)V99.
    05 FV14-X4 REDEFINES FV14-X3.
        10 FV14-MTMGFA      PIC 9(6)V99.
        10 FV14-FILLER      PIC X(4).
        10 FV14-IRETCA      PIC X(01).
        10 FV14-FILLER      PIC X(2).
        10 FV14-ZNCERT.
            15 FV14-ZNALEA      COMP-2.
            15 FV14-NOSCP1      COMP-2.
            15 FV14-NOSEC2      COMP-2.
            15 FV14-NOCERT      PIC 9(4)COMP-3.
            15 FV14-FILLER      PIC X(16).
    05  FV14-X5 REDEFINES FV14-X3.
        10 FV14-FIL1      PIC X(16).
        10 FV14-MNT1      PIC S9(6)V99.
    05  FV14-X6 REDEFINES FV14-X3.
        10 FV14-FIL3      PIC X(16).
        10 FV14-MNT3      PIC S9(6).
        10 FV14-FIL4      PIC X(2).
```

The discrimination rules are written in the following format:

Listing 3-2 Cobol COPY discrimination rules

```
Field FV14-X3
Rule if FV14-X1 = "A"           then  FV14-X3
elseif FV14-X1 = "B"           then  FV14-X4
elseif FV14-X1 = "C"           then  FV14-X5
else                             FV14-X6
```

Note: In the COBOL description, the field FV14-X3 must be the first field to be redefined. The order of subsequent fields: FV14-X4, FV14-X5 and FV14-X6 is not important.

The copy name of the COBOL description is: <COPY name>.cpy

Redefinition examples

Non-equivalent redefinition

Listing 3-3 Non-equivalent redefinition example

```
01 FV15.
   05 FV15-MTMGFA           PIC 9(2).
   05 FV15-ZNPCP3.
       10 FV15-NMASMG       PIC X(2).
       10 FV15-FILLER       PIC X(12).
       10 FV15-COINFA       PIC 9(6)V99.
   05 FV15-ZNB2T REDEFINES FV15-ZNPCP3.
       10 FV15-MTMGFA       PIC 9(4)V99.
```

```

10 FV15-FILLER          PIC X(4) .
10 FV15-IRETCA          PIC X(01) .
10 FV15-FILLER          PIC X(2) .
10 FV15-ZNCERT
    15 FV15-ZNALEA      COMP-2 .
    15 FV15-NOSCP1     COMP-2 .
    15 FV15-NOSEC2     COMP-2 .
    15 FV15-NOCERT     PIC 9(4)    COMP-3 .
    15 FV15-FILLER     PIC X(16) .

```

In the above example, two fields (FV15-ZNPCP3 and FV15-ZNB2T) have different structures: an EBCDIC alphanumeric field in one case and a field composed of EBCDIC data and COMP2, COMP3 data in a second case.

The implementation of a discrimination rule will be necessary to migrate the data to a UNIX platform.

Listing 3-4 Related discrimination rules

```

Field FV15-ZNPCP3
Rule if FV15-MTMGFA = 12          then FV15-ZNPCP3
    elseif FV15-MTMGFA = 08 and FV15-NMASMG = "KC" then FV15-ZNB2T

```

Equivalent redefinition called technical redefinition

Listing 3-5 Technical redefinition initial situation

```

01 FV1 .
    05 FV1-ZNPCP3 .

```

```
10 FV1-MTMGFA          PIC 9(6)V99.
10 FV1-NMASMG          PIC X(25).
10 FV1-FILLER          PIC X(12).
10 FV1-COINFA          PIC 9(10).
10 FV2-COINFA REDEFINES FV1-COINFA.
    15 FV2-ZNALEA      PIC 9(2).
    15 FV2-NOSCP1     PIC 9(4).
    15 FV2- FILLER    PIC 9(4).
10 FV15-MTMGFA        PIC 9(6)V99.
10 FV15-FILLER        PIC X(4).
10 FV15-IRETCA        PIC X(01).
10 FV15-FILLER        PIC X(2).
```

Listing 3-6 Technical redefinition potential expected results

```
01 FV1.                                01 FV1.
05 FV1-ZNPCP3.                          05 FV1-ZNPCP3.
10 FV1-MTMGFA PIC 9(6)V99.              10 FV1-MTMGFA PIC 9(6)V99.
10 FV1-NMASMG PIC X(25).                10 FV1-NMASMG PIC X(25).
10 FV1-FILLER PIC X(12).                10 FV1-FILLER PIC X(12).
10 FV1-COINFA PIC 9(10).                10 FV2-COINFA.
10 FV15-MTMGFA PIC 9(6)V99.             15 FV2-ZNALEA PIC 9(2).
10 FV15-FILLER PIC X(4).                15 FV2-NOSCP1 PIC 9(4).
10 FV15-IRETCA PIC X(01).               15 FV2- FILLER PIC X(4).
10 FV15-FILLER PIC X(2).                10 FV15-MTMGFA PIC 9(6)V99.
                                         10 FV15-FILLER PIC X(4).
                                         10 FV15-IRETCA PIC X(01).
                                         10 FV15-FILLER PIC X(2).
```

In the above example, the two descriptions correspond to a simple EBCDIC alphanumeric character string (without binary, packed or signed numeric fields). this type of structure does not require the implementation of a discrimination rule.

Preparing the environment

This section describes the tasks to perform before generating the components to be used to migrate the data files.

Initializing environment variables

Before executing the Rehosting Workbench set the following environment variables:

- `export TMPPROJECT=$HOME/tmp`

— the location for storing temporary objects generated by the process.

- `export PARAM=$HOME/param`

— the location of the configuration files.

Implementing the configuration files

Three files need to be placed in the Rehosting Workbench file structure as described by:

- `$PARAM` for:
 - `db-param.cfg`,
- `$PARAM/file` for:
 - `Datamap-<configuration name>.re`,
 - `mapper-<configuration name>.re`.

For a File-to-File conversion you must create these files yourself.

Note: The Datamap and mapper configuration files must use the same configuration name.

Two other configuration files:

- `file-templates.txt`
- `file-move-assignment.pgm`

are automatically placed in the file structure during the installation of the Rehosting Workbench. If specific versions of these files are required for particular z/OS files they will be placed in the `$PARAM/file` file structure.

Configuring the files

The following examples show the configuration of three files; two QSAM files and one VSAM KSDS file. There are no discrimination rules to implement for these files.

Database Parameter File (`db-param.cfg`)

For the `db-param.cfg` file, the only parameter you may need to modify is the `target_os` parameter.

Listing 3-7 `db-param.cfg` example

```
# This configuration file is used by FILE & RDBMS converter
```

```
# Lines beginning with "#" are ignored
# write information in lower case
# common parameters for FILE and RDBMS
# source information is written into system descriptor file (OS, DBMS=,
# DBMS-VERSION=)
target_rdbms_name:oracle
target_rdbms_version:11
target_os:unix
#
# specific parameters for FILE to RDBMS conversion
file:char_limit_until_varchar:29
# specific parameters for RDBMS conversion
rdbms:date_format:YYYY/MM/DD
rdbms:timestamp_format:YYYY/MM/DD HH24 MI SS
rdbms:time_format:HH24 MI SS
# rename object files
# the file param/rdbms/rename-objects-<schema>.txt is automatically loaded
# by the tool if it exists.
```

Datamap parameter file (Datamap-<configuration name>.re)

Each z/OS file to be migrated must be listed.

The following parameters must be set:

Table 3-2 Datamap parameters

Parameter	Value in example	Set by
configuration name	FTFIL001 (data map FTFIL001-map)	Freely chosen by user.
system name	PROJ001 (system cat::PROJ001)	Determined during cataloging in the System description file .
file name	PJ01DDD.DO.QSAM.KBCOI001	z/OS physical file name.
Organization	Sequential	Source organization

Note: The description of the different parameters used is provided in the Oracle Tuxedo Application Rehosting Workbench Reference Guide - File To File Converter.

In the following example, the first two files are QSAM files, the organization is therefore always sequential. The PJ01AAA.SS.VSAM.CUSTOMER file is a VSAM KSDS file and the organization is therefore indexed. The parameters, `keys offset 1 bytes length 6 bytes primary`, describe the key. In this example, the key is six bytes long starting in position 1.

Listing 3-8 Example datamap file: Datamap-FTFIL001.re

```
%% Lines beginning with "%%" are ignored

data map FTFIL001-map system cat::PROJ001
%%
%% Datamap File PJ01DDD.DO.QSAM.KBCOI001
%%
file PJ01DDD.DO.QSAM.KBCOI001
organization Sequential
%%
%% Datamap File PJ01DDD.DO.QSAM.KBCOI002
```

```

%%
file PJ01DDD.DO.QSAM.KBCOI002
organization Sequential
%%
%% Datamap File PJ01AAA.SS.VSAM.CUSTOMER
%%
file PJ01AAA.SS.VSAM.CUSTOMER
    organization Indexed
    keys offset 1 bytes length 6 bytes primary

```

Mapping parameter file (mapper-<configuration name>.re)

Each z/OS file to be migrated, that is included in the Datamap configuration file, must be listed.

The following parameters must be set:

Table 3-3 Mapping parameters

Parameter	Value in example	Set by
configuration name	FTFIL001 (ufas mapper FTFIL001)	Identical to the name used in the datamap configuration file.
file name	PJ01DDD.DO.QSAM.KBCOI001	Name used in the Datamap file.
include	include "#VAR:RECS-SOURCE#/BCOAC01E .cpy"	During the generation, the string #VAR:RECS-SOURCE# will be replaced by the directory name where the copy files are located: \$PARAM/file/recs-source The name of the copy file BCOAC01E.cpy is freely chosen by the user when creating the file.
map record	map record REC-ENTREE defined in "#VAR:RECS-SOURCE#/BCOAC01E .cpy"	REC-ENTREE corresponds to the level 01 field name in the copy file.

Table 3-3 Mapping parameters

Parameter	Value in example	Set by
logical name	logical name FQSAM01	Chosen by the user, maximum eight characters. This name is used for naming the objects (COBOL, JCL) created by the different tools in the Rehosting Workbench.
converter name	converter name FQSAM01	Same name and use as logical name.

Note: The description of the different parameters used is provided in the Oracle Tuxedo Application Rehosting Workbench Reference Guide - File To File Converter.

Listing 3-9 Example mapper file: mapper-FTFIL001.re

```
%% Lines beginning with "%%" are ignored
ufas mapper FTFIL001
%%
%% Desc file PJ01DDD.DO.QSAM.KBCOI001
%%
file PJ01DDD.DO.QSAM.KBCOI001 transferred
include "#VAR:RECS-SOURCE#/BCOAC01E.cpy"
map record REC-ENTREE defined in "#VAR:RECS-SOURCE#/BCOAC01E.cpy"
source record REC-ENTREE defined in "#VAR:RECS-SOURCE#/BCOAC01E.cpy"
logical name FQSAM01
converter name FQSAM01
%%
%% Desc file PJ01DDD.DO.QSAM.KBCOI002
%%
file PJ01DDD.DO.QSAM.KBCOI002 transferred
include "#VAR:RECS-SOURCE#/BCOAC04E.cpy"
```

```
map record REC-ENTREE-2 defined in "#VAR:RECS-SOURCE#/BCOAC04E.cpy"
source record REC-ENTREE-2 defined in "#VAR:RECS-SOURCE#/BCOAC04E.cpy"
logical name FQSAM02
converter name FQSAM02

%%

%% Desc file PJ01AAA.SS.VSAM.CUSTOMER

%%

file PJ01AAA.SS.VSAM.CUSTOMER transferred

include "COPY/ODCSF0B.cpy"

map record VS-ODCSF0-RECORD defined in "COPY/ODCSF0B.cpy"
source record VS-ODCSF0-RECORD in "COPY/ODCSF0B.cpy"
logical name ODCSF0B
converter name ODCSF0B
```

Installing the copy files

Create a \$PARAM/file/recs-source directory to hold the copy files.

Once the [COBOL description](#) files have been prepared, the copy files described in the `mapper-<configuration name>.re` file should be placed in the \$PARAM/file/recs-source directory.

If you use a COBOL copy book from the source platform to describe a file (see note in [COBOL description](#)), then it is the location of the copy book that is directly used in the mapping parameter file as in the "COPY/ODCSF0B.cpy" example above.

Generating the components

To generate the components used to migrate z/OS files the Rehosting Workbench uses the `file.sh` command. This section describes the command.

file.sh

Name

file.sh — Generate z/OS migration components.

Synopsis

```
file.sh [ [-g] [-m] [-i <installation directory>] <configuration name> | -s  
<installation directory> (<configuration name>,...) ]
```

Description

file.sh generates the components used to migrate z/OS files using the Rehosting Workbench.

Options

-g <configuration name>

Generation option. The unloading and loading components are generated in \$TMPPROJECT using the information provided by the configuration files.

-m <configuration name>

Modification option. Makes the generated shell scripts executable. The COBOL programs are adapted to Micro Focus COBOL fixed format. When present, the shell script that modifies the generated source files is executed.

-i <installation directory><configuration name>

Installation option. Places the components in the installation directory. This operation uses the information located in the file-move-assignment.pgm file.

-s

Not applicable to File-to-File migration.

Example

```
file.sh -gmi $HOME/trf FTFIL001
```

Locations of generated files

The unloading and loading components generated with the `-i $HOME/trf` option are placed in the following locations:

Table 3-4 Location of components

Location	Contents
<code>\$HOME/trf/unload/file/<configuration name></code>	The JCL used for each unloading are generated for each <code><configuration name></code> .
<code>\$HOME/trf/reload/file/<configuration name></code>	The programs and KSH used for each loading are generated for each <code><configuration name></code> .
<code>\$TMPPROJECT/outputs</code>	The generation log files <code>Mapper-log-<configuration name></code> can be used to resolve problems.

Modifying generated components

The generated components may be modified using a project's own scripts. These scripts (`sed`, `awk`, `perl`, ...) should be placed in:

```
$PARAM/file/file-modif-source.sh
```

When present, this file will be automatically executed at the end of the generation process. It will be called using the `<configuration name>` as an argument.

Using the make utility

Make is a UNIX utility intended to automate and optimize the construction of targets (files or actions).

You should have a descriptor file named `makefile` in the source directory in which all operations are implemented (a `makefile` is prepared in the source directory during the initialization of a project).

The next two sections describe configuring a make file and how to use the Rehosting Workbench File-To-File Converter functions with a make file.

Configuring a make file

Version.mk

The `version.mk` configuration file in `$PARAM` is used to set the variables and parameters required by the make utility.

In `version.mk` specify where each type of component is installed and their extensions, as well as the versions of the different tools to be used. This file also describes how the log files are organized.

The following general variables should be set at the beginning of migration process in the `version.mk` file:

- `ROOT_PROJECT`
- `DIR_TRADDATA`
- `TOOLS_DATA`
- `LOGDIR`

In addition, the `FILE_SCHEMAS` variable is specific to file migration, it indicates the different configurations to process.

This configuration should be complete before using the make file.

make file contents

The contents of the makefile summarize the tasks to be performed:

- Including `vesion.mk`
- Controlling if the main variables are set
- Collecting the value of variables and building the required lists to process.
- Implementing the targets.

A `makefile` and a `version.mk` file are provided with the Rehosting Workbench Simple Application.

Using a makefile with the Rehosting Workbench File-To-File Converter

The `make FileConvert` command can be used to launch the Rehosting Workbench File-To-File Converter. It enables the generation of the components required to migrate z/OS files to a UNIX/Linux target platform.

The make file launches the `file.sh` tool with the `-g`, `-m` and `-i` options, for all configurations contained in the `FILE_SCHEMAS` variable.

Performing the migration

This section describes the tasks of unloading, transfer and reloading using the components generated using the Rehosting Workbench (see [Generating the components](#)).

Preparation

Configuring the environments and installing the components

Installing the unloading components under z/OS

The components used for the unloading (generated in `$HOME/trf/unload/file`) should be installed on the source z/OS platform. The generated JCL may need adapting to specific site constraints including JOB cards, library access paths and access paths to input and output files (Data Set Name – DSN).

Installing the reloading components on target platform

The components used for the reloading (generated in `$HOME/trf/reload/file`) should be installed on the target platform (runtime).

The following environment variables should be set on the target platform:

Table 3-5

Variable	Value
DATA_SOURCE	The name of the directory containing the files transferred from z/OS to be reloaded.
BIN	The location of the generic reload and control scripts (<code>\$HOME/trf/reload/bin</code>)
TMPPROJECT	The temporary directory.

Table 3-5

Variable	Value
MT_LOG	Directory to contain execution logs.
DATA	Directory containing the output files from the transcoding and reloading processes. These are the files ready to be used on the target platform.

Unloading JCL

An unloading JCL is generated for each z/OS file listed in the [Datamap parameter file \(Datamap-<configuration name>.re\)](#). These unloading JCLs are named *<logical file name>.jclunload*

Note: The `.jclunload` extension should be deleted for execution under z/OS.

Transferring the files

Files should be transferred between the source z/OS platform and the target platform in binary format using the file transfer tools available at the site (CFT, FTP, ...).

Compiling the transcoding programs

The generated COBOL programs used for transcoding and reloading are named:

RELFILE-<logical file name>

For the example provided in [Mapping parameter file \(mapper-<configuration name>.re\)](#), the generated programs are:

- RELFILE-FQSAM01.cbl,
- RELFILE-FQSAM02.cbl,
- RELFILE-ODCSF0B.cbl.

Sequential files

When migrating a sequential file, a Micro Focus LINE SEQUENTIAL output file will be generated:

Listing 3-10 FILE CONTROL example – extract from program: RELFILE-FQSAM01.cbl:

```
...  
SELECT SORTIE  
    ASSIGN TO "SORTIE"  
        ORGANIZATION IS LINE SEQUENTIAL  
        FILE STATUS IS IO-STATUS.  
...
```

VSAM KSDS files

When migrating a VSAM KSDS file, an INDEXED output file will be generated:

Listing 3-11 FILE CONTROL example – extract from program: RELFILE-ODCSFOB.cbl:

```
...  
SELECT MW-SORTIE  
    ASSIGN TO "SORTIE"  
        ORGANIZATION IS INDEXED  
        ACCESS IS DYNAMIC  
        RECORD KEY IS VS-CUSTIDENT  
        FILE STATUS IS IO-STATUS.  
...
```

These COBOL programs should be compiled with Micro Focus COBOL using the options described in the Cobol converter section of the Rehosting Workbench Reference Guide.

Executing the transcoding and reloading scripts

The transcoding and reloading scripts use the following parameters:

Synopsis

```
loadfile-<logical file name>.ksh [-t/-l] [-c <method>]
```

Options

-t

Transcode and reload the file.

-l

Transcode and reload the file (same action as -t).

-c ftp:<...>:<...>

Implement the verification of the transfer (see [Checking the transfers](#)).

Examples

For the example provided in [Mapping parameter file \(mapper-<configuration name>.re\)](#), the generated scripts are:

- loadfile-FQSAM01.ksh,
- loadfile-FQSAM02.ksh
- loadfile-ODCSF0B.ksh.

Files

By default, the input file is located in the directory indicated by `$DATA_SOURCE`, and the output file is placed in the directory indicated by `$DATA`.

These files are named with the logical file name used in the [Mapping parameter file \(mapper-<configuration name>.re\)](#).

An execution log is created in the directory indicated by `$MT_LOG`.

A return code different from zero is produced when a problem is encountered.

Checking the transfers

This check uses the following option of the `loadfile-<logical file name>.ksh`

```
-c ftp:<name of transferred physical file>:<name of FTP log under UNIX>
```

Note: This option verifies, after the reloading, that the physical file transferred from z/OS and the file reloaded on the target platform contain the same number of records. This check is performed using the FTP log and the execution report of the reloading program. If the number of records is different, an error message is produced.

Troubleshooting

This section describes problems resulting from usage errors that have been encountered when migrating files from the source to target platform.

Overview

When executing any of the Rehosting Workbench tool users should check:

- If any error messages are displayed on the screen.
- If the Mapper-log-<configuration name> file contains any errors (see [Common problems and solutions](#)).

Error messages and associated explanations are listed in the appendix of the Oracle Tuxedo Application Rehosting Workbench Reference Guide.

Common problems and solutions

Error: Unknown file organization *UNDEFINED*

When executing `file.sh -gmi $HOME/trf STFILEORA` the following message appears:
Refine error...

Log

The contents of the Mapper-log-STFILEORA log file include:

```
file PJ01AAA.SS.QSAM.CUSTOMER.REPORT loaded/unloaded
file logical name MW-SYSOUT
*** Unknown file organization : *UNDEFINED*
mapping record MW-SYSOUT
record MW-SYSOUT: logical name MW-SYSOUT
```

```
record MW-SYSOUT: logical name MW-SYSOUT
```

Explanation

A file to be migrated is present in the mapper-`<configuration name>.re` file and absent from the `Datamap.<configuration name>.re` file.

Error: Record... not found

When executing `file.sh -gmi $HOME/trf STFILEORA1` the following message appears:
Refine error...

Log

The contents of the `Mapper-log-STFILEORA1` log file include:

```
file PJ01AAA.SS.QSAM.CUSTOMER.REPORT loaded/unloaded
file logical name MW-SYSOUT
file is sequential: no primary key
*** record `MW-SYSOUT in COPY/MW_SYSOU2T.cpy' not found ***
*** ERROR: all records omitted ***
mapping records
```

Explanation

The copy file is unknown.

Error: Record... not found

When executing `file.sh -gmi $HOME/trf STFILEORA2` the following message appears:
Refine error...

Log

The contents of the `Mapper-log-STFILEORA2` log file include:

```
file PJ01AAA.SS.QSAM.CUSTOMER.REPORT loaded/unloaded
file logical name MW-SYSOUT
file is sequential: no primary key
*** record `MW-SYSOUTTT in COPY/MW_SYSOUT.cpy' not found ***
```

```
record MW-SYSOUT reselected (all records omitted)
mapping record MW-SYSOUT
record MW-SYSOUT: logical name MW-SYSOUT
```

Explanation

The RECORD name (level 01 field) is unknown.

Error: External Variable PARAM is not set

When executing `file.sh -gmi $HOME/trf STFILEORA3` the following message appears:
Refine error...

Log

The contents of the Mapper-log-STFILEORA3 log file include:

```
*-----
==
#####
##
Control of schema STFILEORA3
External Variable PARAM is not setted !
ERROR : Check directive files for schema STFILEORA3
```

Explanation

The variable \$PARAM has not been set.

Oracle Tuxedo Application Rehosting Workbench DB2 to Oracle Converter

Overview

Purpose

This chapter describes how to install, implement, and configure the Rehosting Workbench DB2-to-Oracle converter in order to migrate files from a source DB2 database to a target Oracle database.

Skills

When migrating DB2, a good knowledge of COBOL, JCL, z/OS utilities, DB2 and Oracle databases as well as UNIX/Linux Korn Shell is required.

See also

For a comprehensive view of the migration process, see the Oracle Tuxedo Application Rehosting Workbench Reference Guide for the chapters Data Conversion and Cobol Conversion as well as the COBOL Converter chapter of this guide.

Organization

Migrating data files is described in the following sections:

- [The DB2- to-Oracle Migration Process](#)

- [Reengineering rules to implement](#)
- [Preparing the environment](#)
- [Generating the components](#)
- [Performing the migration](#)
- [Troubleshooting](#)

The DB2- to-Oracle Migration Process

File Organizations Processed

When migrating from a z/OS DB2 source platform to an Oracle UNIX target platform, the first question to ask is, which tables should be migrated?. When not all DB2 tables are to be migrated, a DB2 DDL representing the sub-set of objects to be migrated should be built.

Migration Process Steps

The principle steps in the DB2- to-Oracle migration process, explained in detail in the rest of this chapter, are:

1. Create an overall list of the tables to be migrated. When not all of the DB2 tables are to be migrated, build a DB2 DDL for those tables to be migrated; otherwise take the entire DB2 DDL used by the site.
2. Prepare the environment to be used to generate the components.
3. Using the Rehosting Workbench generate the components used in the following steps.
4. Unload the tables from the source platform.
5. Transfer the data to the target platform.
6. Transcode and reload the data.
7. Check the results.
8. Build the Oracle database.

Interaction with other Oracle Tuxedo Application Rehosting Workbench tools

The DB2-to-Oracle migration is dependent on the results of the Cataloger; the DB2-to-Oracle migration impacts the COBOL conversion and should be completed before beginning the program conversion work.

Reengineering rules to implement

This section describes the reengineering rules applied by the Rehosting Workbench when migrating data from a DB2 database to an Oracle database.

Migration rules applied

The list of DB2 objects that are included in the migration towards Oracle are described in [Creating the Oracle objects](#).

Migrated DB2 objects keep their names when migrated to Oracle except for the application of the Rehosting Workbench renaming rules (see [Preparing and implementing renaming rules](#)).

DB2-to-Oracle data type migration rules

Table 4-1 DB2 to Oracle data types

DB2 z/OS data type	Oracle format
CHAR(length)	Becomes CHAR if length <= 2000 bytes Becomes VARCHAR2 if length > 2000 bytes
VARCHAR(length)	VARCHAR2 (length)
DECIMAL(...)	NUMBER(...)
NUMERIC(...)	NUMBER(...)
DEC(...)	NUMBER(...)
SMALLINT	NUMBER(6)
INTEGER	NUMBER(11)
TIMESTAMP	TIMESTAMP
TIMESTMP	TIMESTAMP
DATE	DATE
TIME	DATE
DOUBLE	FLOAT(53)
FLOAT(prec)	FLOAT(53)
REAL	FLOAT(24)

DB2-to-Oracle column property migration rules

A column property can change the behavior of an application program.


```
table;<schema name>;<DB2 table name>;<Oracle table name>
```

- column:

```
Column;<schema name>;<DB2 table name>;<DB2 column name>;<Oracle column name>
```

Comments can be added as following: % Text.

Example:

```
% Modification applied to the AUALPH0T table  
column;AUANPROU;AUALPH0T;NUM_ALPHA;MW_NUM_ALPHA
```

Example of a migration of DB2 objects

In this example, the DB2 DDL contains a table named ODCSF0 with a primary key and a unique index named XCUSTIDEN:

Listing 4-1 DDL example before migration

```
DROP TABLE ODCSF0;  
  
COMMIT;  
  
CREATE TABLE ODCSF0  
  
    (CUSTIDENT DECIMAL(6, 0) NOT NULL,  
     CUSTLNAME CHAR(030) NOT NULL,  
     CUSTFNAME CHAR(020) NOT NULL,  
     CUSTADDRS CHAR(030) NOT NULL,  
     CUSTCITY CHAR(020) NOT NULL,  
     CUSTSTATE CHAR(002) NOT NULL,  
     CUSTBDATE DATE NOT NULL,  
     CUSTEMAIL CHAR(040) NOT NULL,  
     CUSTPHONE CHAR(010) NOT NULL,  
  
     PRIMARY KEY(CUSTIDENT))  
  
IN DBPJ01A.TSPJ01A
```

```
        CCSID EBCDIC;
COMMIT;
CREATE UNIQUE INDEX XCUSTIDEN
        ON ODCSF0
        (CUSTIDENT ASC) USING STOGROUP SGPJ01A;
COMMIT;
```

After applying the migration rules, and without implementing any renaming rules the following Oracle objects are obtained:

Listing 4-2 Oracle table example after migration

```
WHENEVER SQLERROR CONTINUE;
DROP TABLE ODCSF0 CASCADE CONSTRAINTS;
WHENEVER SQLERROR EXIT 3;
CREATE TABLE ODCSF0 (
    CUSTIDENT NUMBER(6) NOT NULL,
    CUSTLNAME CHAR(30) NOT NULL,
    CUSTFNAME CHAR(20) NOT NULL,
    CUSTADDRS CHAR(30) NOT NULL,
    CUSTCITY CHAR(20) NOT NULL,
    CUSTSTATE CHAR(2) NOT NULL,
    CUSTBDATE DATE NOT NULL,
    CUSTEMAIL CHAR(40) NOT NULL,
    CUSTPHONE CHAR(10) NOT NULL);
```

Listing 4-3 Oracle index example after migration

```
WHENEVER SQLERROR CONTINUE;

DROP INDEX XCUSTIDEN;

WHENEVER SQLERROR EXIT 3;

CREATE UNIQUE INDEX XCUSTIDEN ON ODCSF0
(
    CUSTIDENT ASC
);
```

Listing 4-4 Oracle constraint example after migration

```
WHENEVER SQLERROR CONTINUE;

ALTER TABLE ODCSF0 DROP CONSTRAINT CONSTRAINT_01;

WHENEVER SQLERROR EXIT 3;

ALTER TABLE ODCSF0 ADD
    CONSTRAINT CONSTRAINT_01 PRIMARY KEY (CUSTIDENT);
```

Preparing the environment

This section describes the tasks to perform before generating the components to be used to migrate the DB2 data to Oracle.

Implementing the cataloging of the DB2 DDL source files

The DB2 DDL source files to be migrated are located when preparing for the catalog operations. During the migration process, all valid DB2 syntaxes are accepted, although only the SQL CREATE command is handled and migrated to Oracle.

system.desc file parameters

For a DB2-To-Oracle migration, two parameters must be set in the `system.desc` [System description file](#) that is used by all of the Rehosting Workbench tools:

- `DBMS=DB2`.
Indicates the type of RDBMS to migrate.
- `DBMS-VERSION="8"`.
Indicates the version of the RDBMS to migrate.

Schemas

A schema should consist of a coherent set of objects (for example there should be no `CREATE INDEX` for a table that does not exist in the schema).

By default, if the SQL commands of the DB2 DDL are prefixed by a qualifier or an authorization ID, the prefix is used by the Rehosting Workbench as the name of the schema—for example, `CREATE TABLE <qualifier or authorization ID>.table name`.

The schema name can also be determined by the Rehosting Workbench using the `global-options` clause of the `system.desc` file.

For example:

```
system STDB2ORA root ".."  
global-options  
  catalog="..",  
  sql-schema=<schema name>.
```

Implementing the configuration files

Only one file needs to be placed in the Rehosting Workbench file structure as described by `$PARAM`:

- `db-param.cfg`,

Two other configuration files:

- `rdms-templates.txt`
- `rdms-move-assignation.pgm`

are automatically generated in the file structure during the installation of the Rehosting Workbench. If specific versions of these files are required, they will be placed in the \$PARAM/rdbms file structure.

Initializing environment variables

Before executing the Rehosting Workbench set the following environment variables:

- export TMPPROJECT=/\$home/tmp
— the location for storing temporary objects generated by the process.
- export PARAM=\$HOME/param
— the location of the configuration files.

Generation parameters

Listing 4-5 Example db-param.cfg file

```
#
# This configuration file is used by FILE & RDBMS converter
# Lines beginning by "#" are ignored
# write information in lower case
#
# common parameters for FILE and RDBMS
# source information is written into system descriptor file (OS, DBMS=,
# DBMS-VERSION=) target_rdbms_name:oracle
target_rdbms_version:11
target_os:unix
#
# specific parameters for FILE to RDBMS conversion
file:char_limit_until_varchar:29
# specific parameters for RDBMS conversion
rdbms:date_format:YYYY/MM/DD
```

```
rdbms:timestamp_format:YYYY/MM/DD HH24 MI SS
rdbms:time_format:HH24 MI SS
# rename object files
# the file param/rdbms/rename-objects-<schema>.txt is automatically loaded by
# the tool if it exists.
```

Only the parameters `target_<xxxxx>` and `rdbms:<xxxxx>` need to be adapted.

- `target_rdbms_name:oracle`
name of the target RDBMS.
- `target_rdbms_version:11`
version of the target RDBMS.
- `target_os:unix`
Name of the target operating system.

The three `rdbms` parameters indicate the date, timestamp and time formats used by z/OS DB2 and stored in DSNZPARM. These parameters impact the reloading operations and the COBOL date and time manipulations.

WARNING: A correct setting of these parameters is essential.

- `rdbms:date_format:YYYY/MM/DD`
- `rdbms:timestamp_format:YYYY/MM/DD HH24 MI SS`
- `rdbms:time_format:HH24 MI SS`

Generating the components

To generate the components used to migrate data from DB2 databases to Oracle databases, the Rehosting Workbench uses the `rdbms.sh` command. This section describes the command.

`rdbms.sh`

Name

`rdbms.sh` — Generate DB2 to Oracle database migration components.

Synopsis

```
rdbms.sh [ [-c|-C] [-g] [-m] [-r] [-i <installation directory>] <schema name> ] -s <installation directory> (<schema name>,...) ]
```

Description

`rdbms.sh` generates the Rehosting Workbench components used to migrate z/OS DB2 databases to UNIX Oracle databases.

Options

Generation options

-g <schema name>

The unloading and loading components are generated in \$TMPPROJECT using the information provided by the configuration files.

-C <schema name>

The following components are generated in \$TMPPROJECT: DDL Oracle, CTL files of the SQL*LOADER, XML file used by the COBOL converter, configuration files (`mapper.re` and `Datamap.re`). If an error or warning is encountered, the process will not abort.

See [Executing the transcoding and reloading scripts](#) for information about the SQL scripts created during the generation operation.

-c <schema name>

This option has the same result as the `-C` option except the process will abort if an error or warning is generated.

Modification options

-m <schema name>

Makes the generated shell scripts executable. The COBOL programs are adapted to MICROFOCUS COBOL fixed format. When present, the shell script that modifies the generated source is executed.

-r <schema name>

Removes the schema name from the generated objects (create table, table name, CTL file for SQL*LOADER, KSH). When this option is used, the name of the schema can also be removed from the COBOL components by using the option:

`sql-remove-schema-qualifier` located in the `config-cobol` file (COBOL conversion configuration file) used when converting the COBOL components.

Installation Option

-i <installation directory> <schema name>

Places the components in the installation directory. This operation uses the information located in the `rdbms-move-assignation.pgm` file.

Generate configuration files for COBOL conversion

-s <installation directory> <schema name>,...

Enables the generation of the COBOL convertor configuration file. This file takes all of the unitary XML files of the project. All these files are created in `$PARAM/dynamic-config`.

Example: `rdbms-conv.txt` `rdbms-conv-PJ01DB2.xml`

Example

```
rdbms.sh -Cgrmi $HOME/trf PJ01DB2
```

Using the make utility

Make is a UNIX utility intended to automate and optimize the construction of targets (files or actions).

You should have a descriptor file named `makefile` in the source directory in which all operations are implemented (a `makefile` is prepared in the source directory during the initialization of a project).

The next two sections describe configuring a make file and how to use the Rehosting Workbench DB2-To-Oracle Converter functions with a make file.

Configuring a make file

Version.mk

The `version.mk` configuration file in `$PARAM` is used to set the variables and parameters required by the make utility.

In `version.mk` specify where each type of component is installed and their extensions, as well as the versions of the different tools to be used. This file also describes how the log files are organized.

The following general variables should be set at the beginning of migration process in the `version.mk` file:

- `ROOT_PROJECT`
- `DIR_TRADDATA`
- `TOOLS_DATA`
- `LOGDIR`

In addition, the `RDBMS_SCHEMAS` variable is specific to DB2 migration, it indicates the different schemas to process.

This configuration should be complete before using the make file.

make file contents

The contents of the makefile summarize the tasks to be performed:

- Including `vesion.mk`
- Controlling if the main variables are set
- Collecting the value of variables and building the required lists to process.
- Implementing the targets.

A `makefile` and a `version.mk` file are provided with the Rehosting Workbench Simple Application.

Using a makefile with the Rehosting Workbench DB2-To-Oracle Converter

The `make RdbmsConvert` command can be used to launch the Rehosting Workbench DB2-To-Oracle Converter. It enables the generation of the components required to migrate a DB2 database to Oracle.

The make file launches the `rdbms.sh` tool with the `-C`, `-g`, `-r`, `-m` and `-i` options, for all schemas contained in the `RDBMS_SCHEMAS` variable.

Locations of generated files

The unloading and loading components generated with the `-i $HOME/trf` option are placed in the following locations:

Table 4-3 Location of components

Location	Contents
<code>\$HOME/trf/unload/rdbms/<schema name></code>	The JCL used for each unloading are generated for each <code><schema name></code> .
<code>\$HOME/trf/SQL/rdbms/<schema name></code>	Location by <code><schema name></code> of the SQL scripts used to create the Oracle objects. For the example used in this chapter there are three scripts: <ul style="list-style-type: none">• <code>INDEX-ODCSF0.sql</code>• <code>TABLE-ODCSF0.sql</code>• <code>CONSTRAINT-ODCSF0.sql</code>
<code>\$HOME/trf/reload/rdbms/<schema name>/src</code>	Location by <code><schema name></code> of the COBOL transcoding programs. For the example in this chapter, there is one program: <ul style="list-style-type: none">• <code>MOD_ODCSF0.cb1</code>
<code>\$HOME/trf/reload/rdbms/<schema name>/ctl</code>	Location by <code><schema name></code> of the CTL files used by SQL*LOADER. For the example in this chapter, there is one CTL: <ul style="list-style-type: none">• <code>ODCSF0.ctl</code>
<code>\$HOME/trf/reload/rdbms/<schema name>/ksh</code>	Location by <code><schema name></code> of the reloading Korn shell scripts. For the example in this chapter, there is one script: <ul style="list-style-type: none">• <code>loadrdbms-ODCSF0.ksh</code>
<code>\$TMPPROJECT/outputs/<schema name></code>	The generation log files produced when using the <code>-c</code> or <code>-C</code> options. <code>rdbms-converter-<schema name></code> can be used to resolve problems.

Modifying generated components

The generated components may be modified using a project's own scripts. These scripts (sed, awk, perl,...) should be placed in:

```
$PARAM/rdbms/rdbms-modif-source.sh
```

When present, this file will be automatically executed at the end of the generation process. It will be called using the `<schema name>` as an argument.

Performing the migration

This section describes the tasks of unloading, transfer and reloading using the components generated using the Rehosting Workbench (see [Generating the components](#)).

Preparation

Configuring the environments and installing the components

Installing the unloading components under z/OS

The components used for the unloading (generated in `$HOME/trf/unload/rdbms`) should be installed on the source z/OS platform. The generated JCL may need adapting to specific site constraints including JOB cards, library access paths and access paths to input and output files (Data Set Name – DSN).

Installing the reloading components on the target platform

The components used for the reloading (generated in `$HOME/trf/reload/rdbms`) should be installed on the target platform (runtime).

The following environment variables should be set on the target platform:

Table 4-4

Variable	Value
DATA_SOURCE	The name of the directory containing the unloaded DB2 tables transferred from z/OS to be reloaded into Oracle tables.
BIN	The location of the generic reload and control scripts (<code>\$HOME/trf/reload/bin</code>)

Table 4-4

Variable	Value
TMPPROJECT	The temporary directory.
MT_LOG	Directory to contain execution logs.
CTL	Directory containing the <code><table name>.ctl</code> files used by the SQL*LOADER (<code>\$HOME/trf/reload/rdbms/<schema name>/ctl</code>).
DATA_TRANSCODE	Temporary directory used by the DB2 binary data transcoding script (contains temporary files in ASCII format).
NLS_LANG	Set according to the instructions in the Oracle documentation.

The following variable should be set according to the information in the Oracle Tuxedo Application Rehosting Workbench Installation Guide:

- MT_DB_LOGIN.

The reloading script `loadrdbms-<table name>.ksh` uses the SQL*LDR Oracle utility. Because this utility can access to ORACLE servers only, this script should be used in ORACLE servers and not with client connection. This variable should not contain an `@<oracle_sid>` string, especially for this reloading step.

Unloading JCL

To unload each DB2 table, a JCL using the IBM DSNTIAUL utility is executed. The DSNTIAUL utility creates three files for each table:

- a data file,
- a log file,
- a SYSPUNCH file.

These unloading JCLs are named `<table name>.jclunload`

If the table name is shorter or longer than eight characters, the Rehosting Workbench attributes an eight-character name to the z/OS JCL as close as possible to the original. The renaming process maintains the uniqueness of each table name.

- Example: `ODCSF0X1.jclunload`

In the example used in this chapter, the table named ODCSF0 is lengthened to ODCSF0X1 when naming the z/OS JCL.

Transferring the files

The unloaded data files should be transferred between the source z/OS platform and the target UNIX platform in binary format using the file transfer tools available at the site (CFT, FTP, ...).

The LOG and SYSPUNCH files should be transferred in text mode.

The files transferred to the target UNIX platform should be stored in the \$DATA_SOURCE directory.

Creating the Oracle objects

The scripts creating Oracle objects (tables, index, constraints, ...) are created in the \$HOME/trf/SQL/rdbms/<schema name> directory. They should be executed in the target Oracle instance.

The <schema name>.lst file contains the names of all of the tables in hierarchical sequence (parent table then child tables).

The following table lists the DB2 objects managed by the Rehosting Workbench and the name of the script used to create them:

Table 4-5

Object Type	File name	Notes
TABLE	TABLE- <target_table_name>.sql	One file per Table. The file contains the table construction, with column names, data type and attribute(s). Constraints, except NULL/NOT NULL attributes, are not written in this file
INDEX	INDEX- <target_table_name>.sql	This file contains all the CREATE INDEXes associated with the table <target_table_name>. This file will not be generated if there are no indexes defined on the table <target_table_name> Indexes are: unique or not Unique constraint

Table 4-5

Object Type	File name	Notes
CONSTRAINT	CONSTRAINT- <target_table_name>.sql	This file contains all constraints associated with the table <target_table_name>. This file will not be generated if there are no constraints defined on the table <target_table_name> Constraints are: Primary Key, Unique, Check and Foreign key
COMMENT	COMMENT- <target_table_name>.sql	Contains all comments for table and columns. One file per table
VIEW	VIEW-<schema_name>.sql	This file contains all the Views created in the source database/schema. In this release, the Select statements are not automatically converted into the target database language.
SEQUENCE	SEQUENCE-<schema_name>.sql	This file contains all the CREATE SEQUENCES already created on the source Database.
SYNONYM	SYNONYMS-<schema_name>.sql	This file contains all the CREATE SYNONYMS already created on the source Database.
IDENTITY	IDENTITY- <target_table_name>.sql	In case of IDENTITY when migrating from DB2 to ORACLE, the Rehosting Workbench creates a Sequence and a Trigger object.

Compiling the transcoding programs

The generated COBOL programs used for transcoding are named:

MOD_<table name>.cbl

For the example used in this chapter the generated program is:

- MOD_ODCSF0.cbl

The programs should be compiled using Microfocus COBOL and the options documented in the Oracle Tuxedo Application Rehosting Workbench Reference Guide.

The programs produce RECORD SEQUENTIAL files on output that will then be read by the SQL*LOADER utility.

Listing 4-6 FILE CONTROL example – extract from program: MOD_ODCSF0.cbl

```
SELECT MW-SORTIE

        ASSIGN TO "SORTIE"

        ORGANIZATION IS RECORD SEQUENTIAL

        ACCESS IS SEQUENTIAL

        FILE STATUS IS IO-STATUS.
```

Executing the transcoding and reloading scripts

The scripts enabling the transcoding and reloading of data are generated in the directory:

```
$HOME/trf/reload/rdbms/<schema name>/ksh
```

The format of the script names is:

```
loadrdbms-<table name>.ksh
```

In the example used in this chapter, the script is named:

```
loadrdbms-ODCSF0.ksh
```

Each script launches the COBOL program that performs the transcoding and then the SQL*LOADER utility. The CTL files used by SQL*LOADER are named:

```
<table name>.ctl.
```

The CTL file used for the example in this chapter is named:

```
ODCSF0.ctl
```

Transcoding and reloading command

The transcoding and reloading scripts have the following parameters:

Synopsis

```
loadrdbms-<table name>.ksh [-t] [-l] [-c: <method>]
```

Options

- t**
Transcode the file.
- l**
Reload the data into Oracle table.
- c dsntiaul:**
Implement the verification of the transfer (see [Checking the transfers](#)).

Examples

For the example provided in [Example of a migration of DB2 objects](#), the generated script is:

- loadrbms-ODCSF0.ksh

Checking the transfers

This check uses the following option of the `loadrbms-<table name>.ksh`

```
-c dsntiaul
```

Note: This option verifies after the reloading that the reloaded Oracle table contains the same number of records as the equivalent table unloaded from z/OS by the DSNTIAUL utility. If the number of records is different, an error message is produced.

Troubleshooting

This section describes problems resulting from usage errors that have been encountered when migrating data from a source DB2 database to a target Oracle database.

Overview

When executing any of the Rehosting Workbench tools, users should check:

- If any error messages are displayed on the screen.
- If the `rbms-converter-<schema name>.log` file contains any errors (see [Common problems and solutions](#)).

Error messages and associated explanations are listed in the appendix of the Oracle Tuxedo Application Rehosting Workbench Reference Guide.

Common problems and solutions

Error: RDBMS-0105

When executing `$REFINEDIR/$VERS/rdbms.sh -Cgrmi $HOME/trf PJ01DB2 STFLEORA` the following message appears:

```
Fatal RDBMS error.
```

```
Error: RDBMS-0105: Catalog for /home2/wkb9/param/system.desc is out of date and needs to be updated externally.
```

```
Refine error...
```

Explanation

Changes have been made to the DDL, re-perform the cataloging operation.

Error: conversion aborted. Can not read

When executing `$REFINEDIR/$VERS/rdbms.sh -Cgrmi $HOME/trf SCHEMA` the following message appears:

```
Refine error...
```

```
/tmp/refine-exit-status.MOaZwgTphIN14075
```

```
ERROR : conversion aborted . Can not read
```

```
/home2/wkb9/tmp/outputs/SCHEMA/rdbms-converter-SCHEMA.log log file
```

```
abort
```

Explanation

The schema name is not known.

Error: Configuration file /db-param.cfg is missing!

When executing `$REFINEDIR/$VERS/rdbms.sh -Cgrmi $HOME/trf PJ01DB2` the following message appears:

```
*-----  
#####  
CONVERSION OF DDLs and CTL files and GENERATION of directive files  
ERROR : Configuration file /db-param.cfg is missing !
```

```
ERROR : Error in reading configuration file
```

```
Abort
```

Explanation

The external variable PARAM is not set.

Error: Target output directory... is missing

When executing `$REFINEDIR/$VERS/rdbms.sh -Cgrmi $HOME/bad-directory PJ01DB2` the following message appears:

```
*=====
Target output directory /home2/wkb9/bad-directory is missing
Check parameters: -i <output_directory> <schema>
ERROR : usage : rdbms.sh [ [-c|-C] [-g] [-m] [-r] [-i <output_directory>]
<schema_name> ] -s <output_directory> (<schema>,...) ]
abort
```

Explanation

The target directory does not exist.

Oracle Tuxedo Application Rehosting Workbench File-to-Oracle Converter

Overview

Purpose

This chapter describes how to install, implement, and configure the Rehosting Workbench File-to-Oracle converter in order to migrate VSAM files from the source z/OS environment to the target environment.

Skills

When migrating files, a good knowledge of COBOL, JCL, z/OS utilities and UNIX/Linux Korn Shell is required.

See also

For a comprehensive view of the migration process, see the Oracle Tuxedo Application Rehosting Workbench Reference Guide for the chapters Data Conversion and Cobol Conversion as well as the [the Rehosting Workbench Cobol Converter](#) chapter of this guide.

Organization

Migrating data files is described in the following sections:

- [The File-to-Oracle migration process](#)

- [Initializing the process](#)
- [Reengineering rules to implement](#)
- [Preparing the environment](#)
- [Generating the components](#)
- [Performing the migration](#)
- [Troubleshooting](#)

The File-to-Oracle migration process

File organizations processed

When migrating VSAM files from a source platform to an Oracle UNIX target platform, the first question to ask, when VSAM is concerned, is whether to keep a file or migrate the data to an Oracle table.

The following file organizations handled by z/OS can be migrated using the Rehosting Workbench to Oracle databases: VSAM RRDS, ESDS and KSDS.

The Oracle Tuxedo Application Rehosting Workbench File-to-Oracle converter is used for those files that are to be converted to Oracle tables. For files that remain in file format, see [Oracle Tuxedo Application Rehosting Workbench File-to-File Converter](#).

Migration Process Steps

The principle steps in the File-to-Oracle migration process, explained in detail in the rest of this chapter, are:

1. Create an overall list of the files to be migrated.
2. List those files that will be converted to Oracle tables.
3. For each of these files, when necessary, create a list of discrimination rules.
4. Prepare the environment to be used to generate the components.
5. Using the Rehosting Workbench generate the components used in the following steps.
6. Unload the data from the source platform.

7. Transfer the data to the target platform.
8. Transcode and reload the data.
9. Check the results.
10. Compile the access routines and the generated utilities.
11. Create the Oracle database.

Interaction with other Oracle Tuxedo Application Rehosting Workbench tools

The migration of data in VSAM files to Oracle tables is dependant on the results of the [Oracle Tuxedo Application Rehosting Workbench Cataloger](#). The File-to-Oracle migration impacts the COBOL and JCL conversion and should be completed before beginning the COBOL program conversion work.

Initializing the process

This section describes the steps to be performed before starting the migration of VSAM files to Oracle tables.

Listing the files to be migrated

The first task is to list all of the VSAM files to be migrated (in conjunction with the use of the File -to-File converter), and then identify those files that should be converted to Oracle tables. For example, permanent files to be later used via Oracle or files that need locking at the record level.

File descriptions and managing files with the same structure

For each candidate file for migration, its structure should be described in Cobol format. This description is used in a Cobol copy by the Rehosting Workbench Cobol converter, subject to the limitations described in [COBOL description](#).

Once built, the list of files to migrate can be purged of files with the same structure in order to save work when migrating the files by limiting the number of programs required to transcode and reload data.

From the purged list of files, a last task consists of building the files:

- `Datamap-<datamap name>.re`
- `mapper-<mapper name>.re`

COBOL description

A COBOL description is related to each file and considered as the representative COBOL description used within the application programs. This description can be a complex COBOL structure using all COBOL data types, including the OCCURS and REDEFINES notions.

This COBOL description will often be more developed than the COBOL file description (FD). For example, an FD field can be described as a PIC X(364) but really contain a three times defined area including, in one case a COMP-3 based numerals table, and in another case a complex description of several characters/digits fields etc.

It is this developed COBOL description which describes the application reality and therefore is used as a base to migrate a specific physical file.

The quality of the file processing execution depends on the quality of this COBOL description. From this point, the COBOL description is not separable from the file and when referring to the file concerned, we mean both the file and its representative COBOL description. The description must be provided in COBOL format, in a file with the following name:

`<COPY name>.cpy`

Note: If a copy book on the source platform provides a detailed description of the file, the file can be directly used and declared in the Rehosting Workbench.

COBOL description format

The format of the COBOL description must conform to the following rules:

- Only one level 01.
- The word FILLER is not allowed.
- Fields names must be unique.
- Some words are reserved, a list is supplied in the Appendix of the Oracle Tuxedo Application Rehosting Workbench Reference Guide.
- The description should begin in column 1 without any preceding sequence numbers.
- Comments may be inserted by placing an * in column 1.

- Field level numbers can start from column 2.

Example

Column																														
1	2	3	4	5	6	7	8	9	1	1	1	1	1	1	1	1	1	1	2	2	2	2	2	2	.					
									0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5						
*	D	E	S	C	R	I	P	T	I	O	N		O	F		F	I	L	E		X	X	X	X						
	0	1		F	V	1	4	.																						
				0	5		F	V	1	4	x	1		P	I	C		X	.											

COBOL description and related discrimination rules

Within a COBOL description there are several different ways to describe the same memory field, which means to store objects with different structures and descriptions at the same place.

As the same memory field can contain objects with different descriptions, to be able to read the file, we need a mechanism to determine the description to use in order to interpret correctly this data area.

We need a rule which, according to some criteria, generally the content of one or more fields of the record, will enable us to determine (discriminate) the description to use for reading the re-defined area.

In the Rehosting Workbench this rule is called a discrimination rule.

Any redefinition inside a COBOL description lacking discrimination rules presents a major risk during the file transcoding. Therefore, any non-equivalent redefined field requests a discrimination rule. On the other hand, any equivalent redefinition (called technical redefinition) must be subject to a cleansing within the COBOL description (see the example below).

The discrimination rules must be presented per file and highlight the differences and discriminated areas. Regarding the files, it is impossible to reference a field external to the file description.

The following description is a sample of a COPY as expected by the Rehosting Workbench:

Listing 5-1 Cobol COPY sample

```
01 FV14.
    05  FV14-X1      PIC X.
    05  FV14-X2      PIC XXX.
    05  FV14-X3.
        10 FV14-MTMGFA      PIC 9(2).
        10 FV14-NMASMG      PIC X(2).
        10 FV14-FILLER      PIC X(12).
        10 FV14-COINFA      PIC 9(6)V99.
    05 FV14-X4 REDEFINES FV14-X3.
        10 FV14-MTMGFA      PIC 9(6)V99.
        10 FV14-FILLER      PIC X(4).
        10 FV14-IRETCA      PIC X(01).
        10 FV14-FILLER      PIC X(2).
        10 FV14-ZNCERT.
            15 FV14-ZNALEA      COMP-2.
            15 FV14-NOSCP1      COMP-2.
            15 FV14-NOSEC2      COMP-2.
            15 FV14-NOCERT      PIC 9(4)COMP-3.
            15 FV14-FILLER      PIC X(16).
    05  FV14-X5 REDEFINES FV14-X3.
        10 FV14-FIL1      PIC X(16).
        10 FV14-MNT1      PIC S9(6)V99.
    05  FV14-X6 REDEFINES FV14-X3.
        10 FV14-FIL3      PIC X(16).
        10 FV14-MNT3      PIC S9(6).
        10 FV14-FIL4      PIC X(2).
```

The discrimination rules are written in the following format:

Listing 5-2 Cobol COPY discrimination rules

```
Field FV14-X3
Rule if FV14-X1 = "A"           then  FV14-X3
elseif FV14-X1 = "B"           then  FV14-X4
elseif FV14-X1 = "C"           then  FV14-X5
else                             FV14-X6
```

Note: In the COBOL description, the field FV14-X3 must be the first field to be redefined. The order of subsequent fields: FV14-X4, FV14-X5 and FV14-X6 is not important.

The copy name of the COBOL description is: <COPY name>.cpy

Redefinition examples

Non-equivalent redefinition

Listing 5-3 Non-equivalent redefinition example

```
01 FV15.
    05 FV15-MTMGFA           PIC 9(2).
    05 FV15-ZNPCP3.
        10 FV15-NMASMG       PIC X(2).
        10 FV15-FILLER       PIC X(12).
        10 FV15-COINFA       PIC 9(6)V99.
    05 FV15-ZNB2T REDEFINES FV15-ZNPCP3.
        10 FV15-MTMGFA       PIC 9(4)V99.
```

```

10 FV15-FILLER          PIC X(4) .
10 FV15-IRETCA          PIC X(01) .
10 FV15-FILLER          PIC X(2) .
10 FV15-ZNCERT
    15 FV15-ZNALEA      COMP-2 .
    15 FV15-NOSCP1     COMP-2 .
    15 FV15-NOSEC2     COMP-2 .
    15 FV15-NOCERT     PIC 9(4)    COMP-3 .
    15 FV15-FILLER     PIC X(16) .

```

In the above example, two fields (FV15-ZNPCP3 and FV15-ZNB2T) have different structures: an EBCDIC alphanumeric field in one case and a field composed of EBCDIC data and COMP2, COMP3 data in a second case.

The implementation of a discrimination rule will be necessary to migrate the data to a UNIX platform.

Listing 5-4 Related discrimination rules

```

Field FV15-ZNPCP3
Rule if FV15-MTMGFA = 12          then FV15-ZNPCP3
    elseif FV15-MTMGFA = 08 and FV15-NMASMG = "KC" then FV15-ZNB2T

```

Equivalent redefinition called technical redefinition

Listing 5-5 Technical redefinition initial situation

```

01 FV1 .
    05 FV1-ZNPCP3 .

```

```
10 FV1-MTMGFA          PIC 9(6)V99.
10 FV1-NMASMG          PIC X(25).
10 FV1-FILLER          PIC X(12).
10 FV1-COINFA          PIC 9(10).
10 FV2-COINFA REDEFINES FV1-COINFA.
    15 FV2-ZNALEA      PIC 9(2).
    15 FV2-NOSCP1      PIC 9(4).
    15 FV2- FILLER     PIC 9(4).
10 FV15-MTMGFA         PIC 9(6)V99.
10 FV15-FILLER         PIC X(4).
10 FV15-IRETCA         PIC X(01).
10 FV15-FILLER         PIC X(2).
```

Listing 5-6 Technical redefinition potential expected results

01 FV1.	01 FV1.
05 FV1-ZNPCP3.	05 FV1-ZNPCP3.
10 FV1-MTMGFA PIC 9(6)V99.	10 FV1-MTMGFA PIC 9(6)V99.
10 FV1-NMASMG PIC X(25).	10 FV1-NMASMG PIC X(25).
10 FV1-FILLER PIC X(12).	10 FV1-FILLER PIC X(12).
10 FV1-COINFA PIC 9(10).	10 FV2-COINFA.
10 FV15-MTMGFA PIC 9(6)V99.	15 FV2-ZNALEA PIC 9(2).
10 FV15-FILLER PIC X(4).	15 FV2-NOSCP1 PIC 9(4).
10 FV15-IRETCA PIC X(01).	15 FV2- FILLER PIC X(4).
10 FV15-FILLER PIC X(2).	10 FV15-MTMGFA PIC 9(6)V99.
	10 FV15-FILLER PIC X(4).
	10 FV15-IRETCA PIC X(01).
	10 FV15-FILLER PIC X(2).

In the above example, the two descriptions correspond to a simple EBCDIC alphanumeric character string (without binary, packed or signed numeric fields). this type of structure does not require the implementation of a discrimination rule.

Reengineering rules to implement

This section describes the reengineering rules applied by the Rehosting Workbench when migrating data from VSAM files to an Oracle database.

Migration rules applied

- Each VSAM file becomes an Oracle table.

- Each table name is stipulated in the `mapper-<configuration name>.re` file using the `table name` clause.
- Each elementary field name contained in the copy description of the file becomes a column name in an Oracle table. Hyphens (-) are replaced by underscore (_) characters.
- For sequential VSAM files (VSAM ESDS):

the Rehosting Workbench adds a technical column: `*_SEQ_NUM NUMBER(8)`.

This column is incremented each time a new line is added to the table; the column becomes the primary key of the table.
- For relative VSAM files (VSAM RRDS):

the Rehosting Workbench adds a technical column `*_RELATIVE_NUM`.

The size of the column is deduced from the information supplied in the Datamap parameter file; the column becomes the primary key of the table.

The column:

 - is incremented when a sequential write is made to the table, and the relative key is zero.
 - contains the relative key when the relative key is not zero.
- For indexed VSAM files (VSAM KSDS):

the Rehosting Workbench does not add a technical column unless duplicate keys are accepted; the primary key of the VSAM file becomes the primary key of the table.

Rules applied to Picture clauses

The following rules are applied to COBOL Picture clauses when migrating data from VSAM files to Oracle tables:

Table 5-1 Picture clause reengineering

COBOL Picture	Oracle format
PIC 9(...)	NUMBER(...)

Table 5-1 Picture clause reengineering

COBOL Picture	Oracle format
COMP-1 or COMP-2	NUMBER(...)
PIC X(...)	Becomes CHAR if length <= 2000 Becomes VARCHAR2 if length > 2000 Note: If the parameter: file:char_limit_until_varc har is set in the db-param.cfg file, it takes precedence over the above rule.

Rules applied to Occurs and Redefines clauses

For OCCURS and REDEFINES clauses with discrimination rules, three reengineering possibilities are proposed:

- Creation of a sub-table:
 - Redefinitions: each description is associated with a sub-table (one sub-table for each description).
 - Occurs: one sub-table is created containing a technical column that references the element of the array to which the data corresponds.
- Creation of an opaque field:
 - Redefinitions: all the descriptions are stored in an opaque field type CHAR or VARCHAR2.
 - Occurs: all the occurrences are stored in an opaque field type CHAR or VARCHAR2.
- Extended description
 - Redefinitions: all the fields described in the copy file are created as columns in the Oracle table.
 - Occurs: each occurrence of a field in a redefined area is created as a column in the Oracle table, one column for each occurrence in the OCCURS clause.

Example VSAM file migration to Oracle table

In the following example, the indexed VSAM file described in ODCSF0B uses as a primary key the VS-CUSTIDENT field.

Listing 5-7 Example VSAM copy description

```
* -----
* Customer record description
* -Record length : 266
* -----

01 VS-ODCSF0-RECORD.
    05 VS-CUSTIDENT          PIC 9(006) .
    05 VS-CUSTLNAME          PIC X(030) .
    05 VS-CUSTFNAME          PIC X(020) .
    05 VS-CUSTADDRS          PIC X(030) .
    05 VS-CUSTCITY           PIC X(020) .
    05 VS-CUSTSTATE          PIC X(002) .
    05 VS-CUSTBDATE          PIC 9(008) .
    05 VS-CUSTBDATE-G        REDEFINES VS-CUSTBDATE.
    10 VS-CUSTBDATE-CC PIC 9(002) .
    10 VS-CUSTBDATE-YY PIC 9(002) .
    10 VS-CUSTBDATE-MM PIC 9(002) .
    10 VS-CUSTBDATE-DD PIC 9(002) .
    05 VS-CUSTEMAIL          PIC X(040) .
    05 VS-CUSTPHONE          PIC 9(010) .
    05 VS-FILLER             PIC X(100) .
* -----
```

Listing 5-8 Oracle table generated from VSAM file

```
WHENEVER SQLERROR CONTINUE;
DROP TABLE CUSTOMER CASCADE CONSTRAINTS;
WHENEVER SQLERROR EXIT 3;
CREATE TABLE CUSTOMER (
    VS_CUSTIDENT          NUMBER(6) NOT NULL,
    VS_CUSTLNAME          VARCHAR2(30),
    VS_CUSTFNAME          CHAR(20),
    VS_CUSTADDRS          VARCHAR2(30),
    VS_CUSTCITY           CHAR(20),
    VS_CUSTSTATE          CHAR(2),
    VS_CUSTBDATE          NUMBER(8),
    VS_CUSTEMAIL          VARCHAR2(40),
    VS_CUSTPHONE          NUMBER(10),
    VS_FILLER             VARCHAR2(100),
    CONSTRAINT PK_CUSTOMER PRIMARY KEY (
        VS_CUSTIDENT)
);
```

Note: The copy book ODCSFOB contains a field redefinition: VS-CUSTBDATE-G PIC 9(008), as this is a technical field, no discrimination rule is implemented. In this case, only the redefined field is created in the generated table VS_CUSTBDATE NUMBER(8).

Preparing the environment

This section describes the tasks to perform before generating the components to be used to migrate data from VSAM files to Oracle tables.

Initializing environment variables

Before executing the Rehosting Workbench set the following environment variables:

- `export TMPPROJECT=$Home/tmp`
 - the location for storing temporary objects generated by the process.
- `export $PARAM=$HOME/param`
 - the location of the configuration files.

Implementing the configuration files

Three files need to be placed in the Rehosting Workbench file structure as described by:

- `$PARAM` for:
 - `db-param.cfg`,
- `$PARAM/file` for:
 - `Datamap-<configuration name>.re`,
 - `mapper-<configuration name>.re`.

For a File-to-Oracle conversion you must create the `Datamap-<configuration name>.re` and `mapper-<configuration name>.re` files yourself.

Two other configuration files:

- `file-templates.txt`
- `file-move-assignation.pgm`

are automatically generated in the file structure during the installation of the Rehosting Workbench. If specific versions of these files are required for particular z/OS files they will be placed in the `$PARAM/file` file structure.

Configuring the files

Database Parameter File (`db-param.cfg`)

For the `db-param.cfg` file, only the target and file parameters need to be adapted.

Listing 5-9 db-param.cfg example

```
# This configuration file is used by FILE & RDBMS converter
# Lines beginning with "#" are ignored
# write information in lower case
# common parameters for FILE and RDBMS
# source information is written into system descriptor file (OS, DBMS=,
# DBMS-VERSION=)
target_rdbms_name:oracle
target_rdbms_version:11
target_os:unix
#
# specific parameters for FILE to RDBMS conversion
file:char_limit_until_varchar:29
# specific parameters for RDBMS conversion
rdbms:date_format:YYYY/MM/DD
rdbms:timestamp_format:YYYY/MM/DD HH24 MI SS
rdbms:time_format:HH24 MI SS
# rename object files
# the file param/rdbms/rename-objects-<schema>.txt is automatically loaded
# by the tool if it exists.
```

- `target_rdbms_name:oracle`
name of the target RDBMS.
- `target_rdbms_version:11`
version of the target RDBMS.
- `target_os:unix`

Name of the target operating system.

- `file:char_limit_until_varchar:29`

Indicates the maximum field length of a COBOL alphanumeric (PIC X) field before the field will be transformed into an ORACLE VARCHAR data type.

In this example, fields longer than 29 characters will become VARCHAR, fields shorter than 30 characters will become CHAR fields.

Datamap parameter file (Datamap-<configuration name>.re)

Each VSAM file to be migrated must be listed.

The following parameters must be set:

Table 5-2 Datamap parameters

Parameter	Value in example	Set by
configuration name	STFILEORA (data map STFILEORA-map)	Freely chosen by user.
system name	STFILEORA (system cat::STFILEORA)	Determined during cataloging in the System description file .
file name	PJ01AAA.SS.VSAM.CUSTOMER	z/OS physical file name.
Organization	Indexed	Source organization

Note: The description of the different parameters used is provided in the Oracle Tuxedo Application Rehosting Workbench Reference Guide - File To File Convertor.

The PJ01AAA.SS.VSAM.CUSTOMER file is a VSAM KSDS file and the organization is therefore indexed. The parameters, `keys offset 1 bytes length 6 bytes primary`, describe the key. In this example, the key is six bytes long starting in position 1.

Listing 5-10 Example datamap file: Datamap-STFILEORA.re

```
%% Lines beginning with "%%" are ignored
data map STFILEORA-map system cat::STFILEORA
%%
```

```

%% Datamap File PJ01AAA.SS.VSAM.CUSTOMER
%%
file PJ01AAA.SS.VSAM.CUSTOMER
    organization Indexed
    keys offset 1 bytes length 6 bytes primary

```

Mapping parameter file (mapper-<configuration name>.re)

Each z/OS file to be migrated, that is included in the Datamap configuration file, must be listed. A file parameter and its options must be included for every VSAM file to convert to an Oracle table. The following parameters must be set:

Table 5-3 Mapping parameters

Parameter	Value in example	Set by
configuration name	STFILEORA (ufas mapper STFILEORA)	Coherent with the name used in the datamap configuration file.
file name	PJ01AAA.SS.VSAM.CUSTOMER	Name used in the Datamap file.
include	include "COPY/ODCSF0B.cpy"	The name and path of the copy file BCOAC01E.cpy is freely chosen by the user when creating the file.
table name	table name CUSTOMER	Provide a name for the Oracle table to be created.
map record	map record VS-ODCSF0-RECORD defined in "COPY/ODCSF0B.cpy"	VS-ODCSF0-RECORD corresponds to the level 01 field name in the copy file.
logical name	logical name ODCSF0B	Chosen by the user, maximum eight characters. This name is used for naming the objects (Cobol, JCL) created by the different tools in the Rehosting Workbench.
converter name	converter name ODCSF0B	Same name and use as logical name.

Note: The description of the different parameters used is provided in the Oracle Tuxedo Application Rehosting Workbench Reference Guide - File To File Converter.

Listing 5-11 Example mapper file: mapper-STFILEORA.re

```
%% Lines beginning with "%%" are ignored
ufas mapper STFILEORA
%%
%% Desc file PJ01AAA.SS.VSAM.CUSTOMER
%%
file PJ01AAA.SS.VSAM.CUSTOMER transferred converted
    table name CUSTOMER
    include "COPY/ODCSF0B.cpy"
    map record VS-ODCSF0-RECORD defined in "COPY/ODCSF0B.cpy"
    source record VS-ODCSF0-RECORD in "COPY/ODCSF0B.cpy"
    logical name ODCSF0B
    converter name ODCSF0B
    attributes LOGICAL_MODULE_IN_ADDITION
```

Installing the copy files

Once the [COBOL description](#) files have been prepared, the copy files described in the `mapper-<configuration name>.re` file should be placed in the `$PARAM/file/recs-source` directory.

If you use a COBOL copy book from the source platform to describe a file (see note in [COBOL description](#)), then it is the location of the copy book that is directly used in the mapping parameter file as in the `"COPY/ODCSF0B.cpy"` example above.

Generating the components

To generate the components used to migrate data from VSAM file to Oracle tables the Rehosting Workbench uses the `file.sh` command. This section describes the command.

file.sh

Name

`file.sh` — Generate z/OS migration components.

Synopsis

```
file.sh [ [-g] [-m] [-i <installation directory>] <configuration name> | -s  
<installation directory> (<configuration name>,...) ]
```

Description

`file.sh` generates the components used to migrate VSAM files by the Rehosting Workbench.

Options

-g <configuration name>

Generation option. The unloading and loading components are generated in \$TMPPROJECT using the information provided by the configuration files.

-m <configuration name>

Modification option. Makes the generated shell scripts executable. The COBOL programs are adapted to Micro Focus COBOL fixed format. When present, the shell script that modifies the generated source files is executed.

-i <installation directory> <configuration name>

Installation option. Places the components in the installation directory. This operation uses the information located in the `file-move-assignment.pgm` file.

-s <installation directory> <schema name>,...

Enables the generation of the configuration files used by the Cobol converter. All these files are created in \$PARAM/dynamic-config.

Example

```
file.sh -gmi $HOME/trf FTFIL001
```

Using the make utility

Make is a UNIX utility intended to automate and optimize the construction of targets (files or actions).

You should have a descriptor file named `makefile` in the source directory in which all operations are implemented (a makefile is prepared in the source directory during the initialization of a project).

The next two sections describe configuring a make file and how to use the Rehosting Workbench File-To-Oracle Converter functions with a make file.

Configuring a make file

Version.mk

The `version.mk` configuration file in `$PARAM` is used to set the variables and parameters required by the make utility.

In `version.mk` specify where each type of component is installed and their extensions, as well as the versions of the different tools to be used. This file also describes how the log files are organized.

The following general variables should be set at the beginning of migration process in the `version.mk` file:

- `ROOT_PROJECT`
- `DIR_TRADDATA`
- `TOOLS_DATA`
- `LOGDIR`

In addition, the `FILE_SCHEMAS` variable is specific to file migration, it indicates the different configurations to process.

This configuration should be complete before using the make file.

make file contents

The contents of the makefile summarize the tasks to be performed:

- Including `version.mk`
- Controlling if the main variables are set

- Collecting the value of variables and building the required lists to process.
- Implementing the targets.

A `makefile` and a `version.mk` file are provided with the Rehosting Workbench Simple Application.

Using a makefile with the Rehosting Workbench File-To-Oracle Converter

The `make FileConvert` command can be used to launch the Rehosting Workbench File-To-Oracle Converter. It enables the generation of the components required to migrate z/OS files to a UNIX/Linux target platform.

The `make` file launches the `file.sh` tool with the `-g`, `-m` and `-i` options, for all configurations contained in the `FILE_SCHEMAS` variable.

Locations of generated files

The unloading and loading components generated with the `-i $HOME/trf` option are placed in the following locations:

Table 5-4 Location of components

Location	Contents
<code>\$HOME/trf/unload/file/<configuration name></code>	The programs and JCL used for each unloading are generated for each <configuration name>.
<code>\$HOME/trf/reload/file/<configuration name></code>	The programs and KSH used for each loading are generated for each <configuration name>. Example: <code>loadfile-ODCSF0.ksh</code> <code>RELFILE-ODCSF0.cbl</code>
<code>\$TMPPROJECT/outputs</code>	The generation log files <code>Mapper-log-<configuration name></code> can be used to resolve problems.

Table 5-4 Location of components

Location	Contents
<code>\$HOME/trf/SQL/file/<schema name></code>	Location of the SQL scripts for creating tables and the Korn shell scripts used to perform various technical operations for the Oracle tables generated.
<code>\$HOME/trf/DML</code>	Location of the COBOL and PRO*COBOL access routines. These routines are executed by the COBOL programs instead of the access verbs used for the VSAM file migrated to Oracle tables.

Examples of generated components

For the example used in this chapter, the following scripts are generated.

The SQL script used to create the CUSTOMER table is named:

- ODCSF0B.sql

The scripts used for the different technical operations are named:

- cleantable-ODCSF0B.ksh
- createtable-ODCSF0B.ksh
- droptable-ODCSF0B.ksh
- ifemptytable-ODCSF0B.ksh
- ifexisttable-ODCSF0B.ksh

Nine COBOL programs are generated, their usage is described in the Oracle Tuxedo Application Rehosting Workbench Reference Guide.

- ASG_ODCSF0B.cbl
- close_all_files_STFILEORA.cbl
- dml_locking.cbl
- init_all_files.cbl

- UL_ODCSF0B.cbl
- close_all_files.cbl
- DL_ODCSF0B.cbl
- getfileinfo.cbl
- init_all_files_STFILEORA.cbl

One PRO*COBOL program for accessing the ORACLE CUSTOMER table is generated:

- RM_ODCSF0B.pco

Modifying generated components

The generated components may be modified using a project's own scripts. these scripts (sed, awk, perl,...) should be placed in:

```
$PARAM/file/file-modif-source.sh
```

When present, this file will be automatically executed at the end of the generation process. It will be called using the <configuration name> as an argument.

Performing the migration

This section describes the tasks of unloading, transfer and reloading using the components generated using the Rehosting Workbench (see [Generating the components](#)).

Preparation

Configuring the environments and installing the components

Installing the unloading components under z/OS

The components used for the unloading (generated in `$HOME/trf/unload/file`) should be installed on the source z/OS platform. The generated JCL may need adapting to specific site constraints including JOB cards, library access paths and access paths to input and out put files (Data Set Name – DSN).

Installing the reloading components under UNIX

The components used for the reloading (generated in `$HOME/trf/reload/file`) should be installed on the target platform (runtime).

Installing the Oracle object creation components

The components used for creating the Oracle objects (generated in `$HOME/SQL/rdbms/<schema name>`) should be installed on the target platform (runtime).

Setting the target platform environment variables

The following environment variables should be set on the target platform:

Table 5-5

Variable	Value
DATA_SOURCE	The name of the directory containing the files transferred from z/OS to be reloaded.
DATA	The name of the directory containing the physical files converted to ASCII format and ready to be loaded to Oracle tables.
BIN	The location of the generic reload and control scripts (<code>\$HOME/trf/reload/bin</code>)
TMPPROJECT	The temporary directory.
MT_LOG	Directory to contain execution logs.
NLS_LANG	Set according to the instructions in the Oracle documentation.
DDL	The location of the SQL scripts used to create Oracle objects (<code>\$HOME/trf/SQL/file/<schema name></code>).

The following variable should be set according to the information in the Oracle Tuxedo Application Rehosting Workbench Installation Guide:

- MT_DB_LOGIN.

Unloading the JCL

An unloading JCL is generated for each z/OS file listed in the [Datamap parameter file](#) (`Datamap-<configuration name>.re`). These unloading JCLs are named `<logical file`

name>.jclunload. These JCL use the REPRO function of the IDCAMS utility to unload the files.

Note: The .jclunload extension should be deleted for execution under z/OS.

Transferring the files

Files should be transferred between the source z/OS platform and the target UNIX platform in binary format using the file transfer tools available at the site (CFT, FTP, ...).

Compiling the transcoding programs

The generated COBOL programs used for transcoding and reloading are named:

```
RELTABLE-<logical file name>
```

Example:

```
RELTABLE-ODCSF0.cbl
```

These COBOL programs should be compiled with Micro Focus COBOL using the options described in the Cobol converter section of the Rehosting Workbench Reference Guide.

Each program produces an output file that is then read by the SQL*LOADER utility.

Executing the Oracle object creation scripts

The option **-d** of the loadtable-<...>.ksh scripts enables the creation of the Oracle objects.

Executing the transcoding and reloading scripts

The transcoding and reloading scripts have the follow parameters:

Synopsis

```
loadtable-<logical file name>.ksh [-d] [-t/-l] [-c <method>]
```

Options

-d

Create the Oracle objects.

-t

Transcode and reload the file.

-l

Transcode and reload the file (same action as -t).

-c ftp:<...>:<...>

Implement the verification of the transfer (see [Checking the transfers](#)).

Examples

For the example provided in [Mapping parameter file \(mapper-<configuration name>.re\)](#), the generated script is:

- loadtable-ODCSF0B.ksh

Files

By default, the input file is located in the directory indicated by `$DATA_SOURCE`, and the output file is placed in the directory indicated by `$DATA`.

These files are named with the logical file name used in the [Mapping parameter file \(mapper-<configuration name>.re\)](#) configuration file.

An execution log is created in the directory indicated by `$MT_LOG`.

A return code different from zero is produced when a problem is encountered.

Checking the transfers

This check uses the following option of the `loadtable-<logical file name>.ksh`

```
-c ftp:<name of transfered physical file>:<name of FTP log under UNIX>
```

Note: This option enables the verification after the reloading that the physical file transferred from z/OS and the file reloaded on the target platform contain the same number of records. This check is performed using the FTP log and execution report of the reloading program. If the number of records is different, an error message is produced.

Compiling the access routines and utilities

The COBOL and PRO*COBOL access routines should be compiled using the Microfocus compilation options described in the Oracle Tuxedo Application Rehosting Workbench Reference Guide.

Troubleshooting

This section describes problems resulting from usage errors that have been encountered when migrating files from the source to target platform.

Overview

When executing any of the Rehosting Workbench tool users should check:

- If any error messages are displayed on the screen.
- If the `Mapper-log-<configuration name>` file contains any errors (see [Common problems and solutions](#)).

Error messages and associated explanations are listed in the appendix of the Oracle Tuxedo Application Rehosting Workbench Reference Guide.

Common problems and solutions

Error: External Variable PARAM is not set

When executing `file.sh -gmi $HOME/trf STFILEORA` the following message appears:

```
*=====
#####
Control of configuration STFILEORA
External Variable PARAM is not set!
ERROR: Check directive files for configuration STFILEORA
```

Abort

Explanation

The variable \$PARAM has not been set.

Error: Target directory does not exist

When executing `file.sh -gmi $HOME/trf STFILEORA1` the following message appears:

```
*-----
Target output directory /home2/wkb9/trf is missing
```

Check parameters: -i <output_directory> <schema>

```
ERROR : usage : file.sh [ [-g] [-m] [-i <output_directory>] <schema_name> |  
-s <output_directory> (<schema>,...) ]
```

abort

Error: Unknown file organization

When executing `file.sh -gmi $HOME/trf STFILEORA2` the following message appears:

Refine error...

Log

The contents of the Mapper-log-STFILEORA2 log file include:

```
file PJ01AAA.SS.VSAM.CUSTOMER Oracle  
    file logical name ODCSF0B  
*** Unknown file organization : INDEXD  
    mapping record VS-ODCSF0-RECORD  
    record VS-ODCSF0-RECORD: logical name VS  
    record VS-ODCSF0-RECORD: logical name VS  
    record VS-ODCSF0-RECORD REDEFINES:  
        field VS-CUSTBDATE as opaque (default strategy)  
    record VS-ODCSF0-RECORD REDEFINES:  
        field VS-CUSTBDATE as opaque (default strategy)
```

Explanation

The file is configured with a file organization of INDEXD instead of INDEXED.

Error: The illegal text is: "converted"

When executing: `file.sh -gmi $HOME/trf STFILEORA` the following message appears:

```
*-----  
#####  
Control of configuration STFILEORA
```

```
#####
Control of templates

Project Templates list file is missing
/home2/wkb9/param/file/file-templates.txt

OK: Use Default Templates list file

File name is
/Qarefine/release/M2_L4_1/convert-data/default/file/file-templates.txt
#####
```

```
Control of Mapper
#####
COMPONENTS GENERATION
```

...

```
Parsing mapper file /home2/wkb9/tmp/mapper-STFILEORA.re.tmp ...
Parse error at character position 1346 in file:
/home2/wkb9/tmp/mapper-STFILEORA.re.tmp.
```

The illegal text is: "converrted

```
table name CUSTOMER
include \"COPY/ODCSF0B.cpy\"
map record VS-ODCSF0-RECORD defin"
```

While parsing in grammar UFAS-CONVERTER::MAPPER-INPUT-GRAMMAR,
 CONVERTED is a symbol, which is not a legal token
 at this point in the input. The legal tokens at this point are:

```
:END "templates" "filler" "field" "file" "one-for-n" "multi-record"
"table" "transferred" "mode-tp" ... [14 others]
```

ERROR: parse error is found in /home2/wkb9/tmp/mapper-STFILEORA.re.tmp.

```
Refine error...
/tmp/refine-exit-status.snICMmEINYF25625
ERROR: generation aborted
abort
```

Explanation

A typing error in the word 'CONVERTED' was made in the mapping file.

Error: Can't find file in file table named PJ01AAA.SS.VSAM.CUSTOMERS

When executing: `file.sh -gmi $HOME/trf STFILEORA` the following message appears:

```
*=====
#####
Control of configuration STFILEORA
#####
Control of templates
Project Templates list file is missing
/home2/wkb9/param/file/file-templates.txt
OK: Use Default Templates list file
File name is
/Qarefine/release/M2_L4_1/convert-data/default/file/file-templates.txt
#####
Control of Mapper
#####
COMPONENTS GENERATION

...

Point 1 !!
Warning: Can't find file in file table named PJ01AAA.SS.VSAM.CUSTOMERS
```

Point 2 !!

...

Refine error...

/tmp/refine-exit-status.IvmFDYsYDdr31956

ERROR : generation aborted

Abort

Explanation

The name of the file to convert to an Oracle table does not correspond to the name present in the Datamap file.

the Rehosting Workbench Cobol Converter

Overview

Purpose

the Rehosting Workbench Cobol Converter aims to generate for each Cobol program on the z/OS source platform the same Cobol program for the target UNIX platform by applying accommodated transformations and modernization and restructuring rules.

The Cobol converter uses a set of configuration files to state which transformation rules to apply and provide specific hints to obtain the desired results.

This chapter describes how to convert Cobol programs from z/OS Cobol to Micro Focus Cobol.

Skills

This guide is intended for anyone using the the Rehosting Workbench to convert programs from z/OS Cobol to Micro Focus Cobol.

When converting Cobol, a good knowledge of COBOL, JCL, MVS utilities and UNIX Korn Shell is required.

See also

For more information, see:

- Oracle Tuxedo Application Rehosting Workbench Reference Guide

Organization

Migrating data files is described in the following sections:

- [Requirements & Prerequisites](#)
- [Overview of the Cobol Converter in the replatforming process](#)
- [Conversion steps](#)
- [Using a make file](#)
- [Troubleshooting the Cobol Converter](#)

Requirements & Prerequisites

Cataloguing requirements

Cataloguing is mandatory before running the Cobol converter in order to check the consistency of the assets to be migrated and fix any errors related either to syntax or to the coherency (missing or unused components) of the asset.

Data Conversion

The data migration process must have been performed before the Cobol conversion is started. This dependency is because the data migration tools generate some of the configuration files read by the Cobol converter. The configuration files from data conversion are documented in the Oracle Tuxedo Application Rehosting Workbench Reference Guide.

Overview of the Cobol Converter in the replatforming process

The inputs to the the Rehosting Workbench Cobol converter process are:

- The abstract syntax trees of the Cobol programs stored in the pob files produced by the the Rehosting Workbench Cataloger.
- A set of configuration files, some of them are produced by the data conversion tools that specifies information about file-to-RDBMS conversion and relational DBMS conversion (from DB2to Oracle).

Conversion steps

Building and setting the configuration files

The main configuration file for translation is `config-cobol`. It references other additional configuration files including:

- `keywords-file`
- `tr-hexa.map`
- `rename-call-map-file`

Samples of all the needed configuration files are given in the Simple Application. You only need to check and adapt the values if necessary.

All of the configuration files that the Cobol Converter uses are described in the Cobol Converter chapter of the Oracle Tuxedo Application Rehosting Workbench Reference Guide.

Configuring config-cobol

After preparing the prerequisites for the cobol conversion, prepare the main configuration file using the following example as a model.

Note: The following files can be used:

- `post-translation-file` used when there is a need to perform some specific transformations. This file is to write manually.
- `rdbms-conversion-file` used when migrating DB2 to an Oracle database. This file is generated by the [Oracle Tuxedo Application Rehosting Workbench DB2 to Oracle Converter](#).
- `conv-ctrl-list-file` used when there are files to be converted to Oracle tables. This file should be generated by the [Oracle Tuxedo Application Rehosting Workbench File-to-Oracle Converter](#).

Listing 6-1 config-cobol file

Config:

```
"Config version 1.0"  
  
/* GENERAL */
```

```
target-dir:    "../trf/".
keywords-file: "../param/keywords-file".
rename-call-map-file: "../param/rename-call-map-file".
accept-date   : MW-DATE.
accept-day    : MW-DAY.
# post-translation-file: "../param/renov.desc".
hexa-map-file: "../param/tr-hexa.map".
# rdbms-conversion-file : "dynamic-config/rdbms-conv.txt".
conv-ctrl-list-file : "dynamic-config/Conv-ctrl.txt".
on-size-error-call : "ABORT".

dcrp.                /* Without reconciliation of copies files */
end
```

keywords-file

The keywords-file is a hint file for the Cobol Converter to help rename specific variables including reserved keywords probably not renamed systematically by the the Rehosting Workbench Cobol Converter.

This is a reengineering mechanism offered by the WorkBench to perform a (mass-change) renaming operation for the customer's own purposes, even when the variables are not MicroFocus reserved keywords.

Place the following entry in the main configuration file config-cobol:

```
keywords-file: "../param/keywords-file"
```

Listing 6-2 Sample keywords file

```
( TAB . MW-TAB )
( DOUBLE . MW-DOUBLE )
```

```
( POS . MW-POS )  
)
```

In this example each occurrence of the item TAB will be replaced by MW-TAB.

tr-hexa.map

The `tr-hexa.map` file is a mapping table between EBCDIC (z/OS code set) and ASCII (Linux/UNIX code set) hexadecimal values.

Place the following entry in the main configuration file `config-cobol`:

```
hexa-map-file: "../param/tr-hexa.map"
```

This file is used by some translation rules like Tr-Hexa-Map and may help the user to solve problems related to differences between EBCDIC and ASCII codes in values and strings that could lead to different behavior in sorting and in string comparisons.

Hexadecimal code for space example

The hexadecimal code for space in z/OS is 40 and the hexadecimal code for space in UNIX is 20.

The statement in the source platform code is:

```
01 VarName          pic X          value X'40'.
```

This is translated as:

Listing 6-3 Hexadecimal code translation

```
*{ Tr-Hexa-Map 1.4.2.1  
*01 VarName          pic X          value X'40'.  
*--  
01 VarName          pic X          value X'20'.  
*}
```

Listing 6-4 Sample of tr-hexa.map

```
00;00
01;01
02;02
03;03
37;04
2d;05
2e;06
2f;07
...
40;20
...
```

rename-call-map-file

The `rename-call-map-file` is a mapping file between the old call name and the new one.

It is used by some translation rules like `Tr-Rename-External-Call` and allows the user to make specific changes if needed.

Place the following entry in the main configuration file `config-cobol`:

```
rename-call-map-file: "../param/rename-call-map-file"
```

Listing 6-5 Example rename-call-map-file

```
(
  ("MQGET" . "MWMQGET")
  ("KIX-ABEND" . "KIX_ABEND")
  ("KIX-ASKTIME" . "KIX_ASKTIME")
)
```

In this example all calls to MQGET are changed to MWMQGET.

Conversion

There are many possibilities allowed by the conversion command options (see the Oracle Tuxedo Application Rehosting Workbench Reference Guide). In this section the following examples are described:

- Translate Batch programs and CICS programs.
- Translate a list of programs.
- Translate one program.

The distinction between programs (Batch and CICS) and sub-programs is mandatory; the option `-cobol-type` takes the following values:

- Batch programs: `batch`
- CICS programs: `tpr`
- Sub-programs: `sub`

In the following command lines, the following working variables are set:

Table 6-1 Conversion variables

Variable	Value
REFINEDIR	/product/art_wb11gR1/refine
PROJECT	\$(HOME)/ simpleapp
PARAM	\$PROJECT/param
SOURCE	\$PROJECT/source
LOGS	\$PROJECT/Logs

Translation of Batch, CICS programs and sub-programs

The command lines are:

- To translate all batch programs:

```
$REFINEDIR/refine cobol-convert -v version -loop -limit=50 -s
$PARAM/system.desc -c $PARAM/config-cobol -cobol-type batch
```

The Cobol Converter knows which are the batch programs to be translated from the system description file which describes all components. Where `version` is the release version, for example `M2_L5_7`

- To translate all CICS programs invoke the command:

```
$REFINEDIR/refine cobol-convert -v version -loop -limit=50 -s
$PARAM/system.desc -c $PARAM/config-cobol -cobol-type tpr
```

- To translate one program, for example the CICS program `PGMM002.cbl`, invoke the command:

```
$REFINEDIR/refine cobol-convert -v version -loop -limit=50 -s
$PARAM/system.desc -c $PARAM/config-cobol -cobol-type tpr
CICS/PGMM002.cbl
```

- To translate all sub-programs files invoke the following command:

```
$REFINEDIR/refine cobol-convert -v version -loop -limit=50 -s
$PARAM/system.desc -c $PARAM/config-cobol -cobol-type sub
```

- To translate all CICS programs invoke the command:

```
$REFINEDIR/refine cobol-convert -v version -loop -limit=50 -s
$PARAM/system.desc -c $PARAM/config-cobol -cobol-type tpr -cics
```

The log file is generated in the directory from where the command line is executed. If you want to have logs in a specific directory or file use `-log-file-base` followed by the path and name of the file to store the execution logs.

```
$REFINEDIR/refine cobol-convert -v version -loop -limit=50 -s
$PARAM/system.desc -log-file-base
$LOGS/trans-cbl/translate-cobol-datetime -c $PARAM/config-cobol
-cobol-type sub
```

In this example, the log file will be generated in

`$LOGS/trans-cbl/translate-cobol-datetime`. The logs directory should be previously created.

Reconciling copybooks

Reconciliation of copybooks can be executed implicitly by the Cobol Converter or can be performed separately (through usage of the `dcrp` option, see [Configuring config-cobol](#)) after all

programs have been generated..

To apply reconciliation separately, you should execute the following script from the directory where converted programs are located, (in the case of the Simple Application, from the \$PROJECT/trf directory:

Listing 6-6 Copy reconciliation

```
for file in `find * -name '*-copies'`  
do  
    $REFINEDIR/scripts/reconcil-copy-opt-imbr $PROJECT/trf $file .cbl  
done
```

Checking results

To check conversion results, verify that:

- All expected programs are generated.
- There are no empty programs.
- There are no truncated programs.
- There are no multi-versions for copybooks (reconciliation completed properly).

For empty or truncated programs, the user can refer to execution logs generated in \$Logs to analyze errors encountered during the conversion.

Compiling

Compilation is a validation step for conversion. A program cannot be considered fully converted if it has not compiled successfully.

Compilation options and settings

You need to check that the compilation environment is configured correctly for Cobol MicroFocus compilation according to the following variables:

Table 6-2 Compilation variables

Variable	Value	Usage
COBDIR	/Product/microfocus/cobol	Cobol product access
PATH	\$COBDIR/bin: \${PATH}	Add Cobdir to PATH variable
LD_LIBRARY_PATH	/usr/lib:\$COBDIR/lib:\${ORACLE_HOME}/lib:\${J AVA_HOME}/lib	Defined LD_LIBRARY_PATH
COBCPY	../DML:../Master-copy/COPY:../fixed-copy:.	Indicates where programs can find includes
PCCINCLUDE	"include=../Master-copy/COPY, include=../fixed-copies, include=.	Indicates where programs can find includes

A compilation options file must be prepared. The compilation options used for the Simple Application are:

Listing 6-7 Compilation options example

```
SOURCEFORMAT "FREE "  
DEFAULTBYTE "00 "  
ADDRSV "COMP-6 "  
COMP-6 "2 "  
ALIGN "8 "  
NOTRUNCCALLNAME  
NOTRUNCCOPY  
NOCOPYLBR  
COPYEXT "cpy, cbl "
```

```
RWHARDPAGE
PERFORM-TYPE "OSVS "
NOOUTDD
INDD
NOTRUNC
HOSTARITHMETIC
NOSPZERO
INTLEVEL " 4 "
SIGN " EBCDIC "
ASSIGN " EXTERNAL "
NOBOUND
SETTINGS
REPORT-LINE " 256 "
WARNING " 2 "
TRACE
LIST ( )
```

For more details on compilation options, see the Oracle Tuxedo Application Rehosting Workbench Reference Guide.

Compilation command

The command to compile the programs BATCH/PGMMB00.cbl:

Listing 6-8 Compilation example

```
# From $PROJECT/trf/BATCH
cd $PROJECT/trf/BATCH
```

```
export COBCPY=../DML:../Master-copy/COPY:../fixed-copy:.  
export PCCINCLUDE="include=../Master-copy/COPY, include=../fixed-copies,  
include=."  
cob -ug PGMMB00.cbl -C "use(../..../compil_tools/opt.dir)" -C  
"list(PGMMB00.lst)" -C XREF -C SETTINGS 2> PGMMB00.err
```

Where:

- opt.dir is a file where the compilation options are specified.
- Compilation errors are generated in PGMMB00.err.
- Compilation is listed in PGMMB00.lst.
- Compilation is successful if GMMB00.gnt is generated

Using a make file

Configuration

The makefile delivered with the Simple Application example implements all conversion operations and can be used in any other project with the following adaptations:

- Set the required working variables.
- Update the `version.mk` configuration file according to your project properties and organization.

This configuration is supplementary to the preceding configuration started in the cataloging step.

Before using the make commands, the user should check the values in the `version.mk` supplied with the Simple Application:

Listing 6-9 `version.mk` example

```
#  
# Defined extensions converted files  
#
```

```

ext_trad = cbl
ext_trad_copy = cpy
ext_trad_ksh = ksh
ext_trad_map = bms

#
# Define Version variables
# Information
# with GLOBAL_VERSION=CURRENT all -v option in the makefile are ignored
#
GLOBAL_VERSION = M2_L3_5
CATALOG = $(GLOBAL_VERSION)
TRAD = $(GLOBAL_VERSION)
TRAD_JCLZ = $(GLOBAL_VERSION)
DATA_TOOLS = $(GLOBAL_VERSION)
RECONCIL_COPY = $(GLOBAL_VERSION)
TIMEOUT = 900
TIMEOUT_PARSE = 300

#
# Define Config and opt files
#
FILE_TRAD_JCL = "$(PARAM)/config-trad-JCL.desc"
FILE_TRAD_COBOL = "$(PARAM)/config-cobol"
COMM_TRADJCL = "-c $(FILE_TRAD_JCL)"

COMM_RECONCIL_COPY = "reconcil-copy-opt-imbr"

```

Cobol conversion

To perform the Cobol conversion the following commands are run from `$$SOURCE`:

- To translate all programs in the source directory

```
make trad
```

- To translate batch programs only:

```
make trad_batch
```

- To translate CICS programs only:

```
make trad_cics
```

- To translate sub-programs:

```
make trad_sub
```

Reconciling copybooks

To reconcile copybooks invoke the following command from `$$SOURCE`:

```
make reconcil_copy
```

Troubleshooting the Cobol Converter

During translation you may encounter error messages or the Cobol Converter may abort. Certain errors are shown below as examples of how to proceed when errors occur.

Configuration file does not exist

Particularly at the beginning of the conversion process, some configuration files could be missing, the Cobol Converter then aborts showing the following error:

Message

```
Parsing config /home2/wkb7/simpleapp/param/config-cobol...
*FATAL*: Hexa-map-file: this file
'/home2/wkb7/simpleapp/param/tr-hexa.map' does not exist
```

```
Error: Uncaught throw of :MESSAGE-ERROR to :MESSAGE-ERROR.
```

```
1 (abort) Quit process.
```

```
Type :b for backtrace, :c <option number> to proceed, or :? for other options
```

Solution

Add the missing configuration file or disable the line in the main configuration file if the file requested (in this example, `tr-hexa.map`) is unnecessary.

Missing POB file

Message

```
Parsing config /home2/wkb7/simpleapp/param/config-cobol...
Creating target file /home2/wkb7/simpleapp/trf/CICS/PGMM002.cbl ...
*FATAL INTERNAL ERROR*: Can't find POB file
/home2/wkb7/simpleapp/source/CICS/pob/PGMM002.cbl.pob; please
re-catalog the system.

FIN

Rest in peace, Refine...
```

Solution

Either the programs is not catalogued yet or the `.pob` file was wrongly deleted. Recatalog to generate the requested file.

POB file too old

Message

```
Creating target file /home2/wkb7/simpleapp/trf/CICS/PGMM002.cbl ...
*FATAL INTERNAL ERROR*: POB file
/home2/wkb7/simpleapp/source/CICS/pob/PGMM002.cbl.pob is less recent
than source file /home2/wkb7/simpleapp/source/CICS/PGMM002.cbl; please
re-catalog the system.
```

Solution

This error occurs if the modification date of the source program is more recent than its corresponding POB file. Sometimes the POB file is generated but the source program is updated later, recatalog the program to ensure you have the latest changes.

Parsing error encountered

A program containing parsing errors is not translated, especially programs with fatal errors during cataloging. Generally, Cobol Converter can translate programs containing errors with severities less than FATAL.

Message

```
Program name is PGMM002
Warning:-- Parse-Error at line 163
*FATAL*: file CICS/PGMM002.cbl contains true parse errors, ABORTING!

FIN
Rest in peace, Refine...
```

Solution

Check the source code of the program and fix any errors, catalog the program and convert again. More details about the parsing errors can be found in the cataloging reports and logs. In principle, you should not start conversion work before fixing errors identified during the cataloging step.

Oracle Tuxedo Application Rehosting Workbench JCL Converter

Overview

Purpose

Oracle Tuxedo Application Rehosting Workbench JCL Translator translates complete z/OS JCL files (main jobs, PROCs, INCLUDEs, etc.) into complete Unix/Linux shell scripts using the Korn shell (ksh) syntax and the Oracle Tuxedo Application Runtime for Batch components.

Skills

This guide is intended for anyone using the Rehosting Workbench to convert from z/OS JCL to UNIX/ Linux Korn Shell script.

When translating JCL, a good knowledge of JCL, Z/OS utilities and UNIX/Linux Korn Shell is required.

In particular, a knowledge of how Job Control Language (JCL) manages programs, jobs and utilities and allocates necessary resources to fulfill required functions.

In addition a good knowledge of how target shell scripts are structured and how Batch Runtime components are used will help the user to better understand JCL translation.

See also

For more information, see:

- [Oracle Tuxedo Application Rehosting Workbench Reference Guide.](#)
- [Oracle Tuxedo Application Runtime for Batch User Guide](#)

Organization

Translating JCL is described in the following sections:

- [Requirements & Prerequisites](#)
- [Overview of the JCL Translator in the replatforming process](#)
- [Translation steps](#)
- [Using a make file](#)

Requirements & Prerequisites

Cataloguing requirements

Cataloguing is mandatory before running the JCL translator in order to check the consistency of the assets to be migrated and fix any errors related either to syntax or to the coherency (missing or unused components) of the asset.

Data Conversion

The data migration process must have been performed before the JCL translation is started. This dependency is because the file migration tools generate some of the configuration files read by the JCL Translator. The configuration files from data conversion are documented in the [Oracle Tuxedo Application Rehosting Workbench Reference Guide](#).

Overview of the JCL Translator in the replatforming process

The inputs to the Rehosting Workbench JCL Translator process are:

- The abstract syntax trees of the JCL stored in the pob files produced by the [Oracle Tuxedo Application Rehosting Workbench Cataloger](#).
- A set of configuration files produced by the data conversion tools that specifies information about the [Oracle Tuxedo Application Rehosting Workbench DB2 to Oracle](#)

[Converter](#) conversion and the [Oracle Tuxedo Application Rehosting Workbench DB2 to Oracle Converter](#) conversion.

- A set of configuration files which are either:
 - Generated by other Oracle Tuxedo Application Rehosting Workbench tools, the [Oracle Tuxedo Application Rehosting Workbench DB2 to Oracle Converter](#) provides the JCL translator with the list of files to be converted to Oracle tables
 - Supplied together with the Rehosting Workbench and customizable.

Translation steps

This section provides information and procedures for:

[Building and setting the configuration files](#)

[Translation steps](#)

[Checking results](#)

Building and setting the configuration files

In addition to the AST of the JCL(s) to convert produced by the [Oracle Tuxedo Application Rehosting Workbench Cataloger](#), the Rehosting Workbench JCL Translator takes as input a main configuration file that specifies various aspects of the translation, such as:

- Customizable top skeleton and bottom skeleton;
- Root pathname for migrated data files, temporary files and spool files;
- The list of data files converted to Oracle tables;
- The sort utility to use on the target platform;
- Component and data-file renaming information;
- Program-launcher configuration;

This file is very easy to write using any standard text editor, or to generate from other sources of information. below, we comment a sample of configuration file written to translate JCL files in the simple application STFILEORA provided with the Rehosting Workbench tools.

The main configuration file also refers to separate sub-files which are either:

- Generated by other Oracle Tuxedo Application Rehosting Workbench tools,

- Supplied together with the Rehosting Workbench (and customizable)
- Wholly written by the user.

Oracle Tuxedo Application Rehosting Workbench comes with examples of configuration files.

The main configuration file for the JCL translation is called `config-trad-JCL.desc` in this guide for use with the Simple Application application.

The following sections describe the different configuration parameters and configuration sub-files and gives a full sample at the end of this section.

Top skeleton and Bottom skeleton

The sub-files specified by the two options `top-skeleton` and `bottom-skeleton` represent respectively a header file and a footer file for the generated script. You can customize these files.

Listing 7-1 Top skeleton and bottom skeleton entries in the main configuration file

```
top-skeleton = "../param/top-ksh.txt"
bottom-skeleton = "../param/bottom-ksh.txt"
```

Listing 7-2 Example of top-ksh.txt prolog code

```
#@ (#) -----
#@ (#) - SCRIPT NAME      ==  ${JOBNAME}.ksh          --- VERSION OF ${DATE]
#@ (#) - AUTHOR           ==
#@ (#) - TREATMENT        ==
#@ (#) - OBSERVATIONS     ==  MAINFRAME MIGRATION
#@ (#) -      ... .
```

Where:

`JOBNAME`

Will take the name of the current JOB after translation.

DATE

Is the generation date of the KSH script.

List of data files converted to Oracle tables

When files are converted to Oracle tables, the main configuration file must reference a sub-configuration file such as:

```
file-list-in-table = "../param/dynamic-config/File-in-table.txt"
```

This sub-file is generated by the [Oracle Tuxedo Application Rehosting Workbench File-to-Oracle Converter](#). This file indicates to the JCL Translator, which are the files that will be converted to Oracle tables in order to correctly translate the steps involving these files. In our example PJ01AAA.SS.VSAM.CUSTOMER is the file to be converted.

For example, when the JCL source involves files that will be converted to Oracle tables, the corresponding shell script uses the Batch Runtime function `m_ProgramExec` with the `-b` option to execute a Cobol program. The `-b` option indicates a connection to the database must be opened before executing the program. For example:

```
m_ProgramExec -b RSSABB01
```

Full example configuration

The general options `root-skeleton`, `target-proc`, `label-end`, management of FSN, etc. are described in the JCL Translator section of the Oracle Tuxedo Application Runtime Workbench Reference Guide.

Listing 7-3 JCL translator configuration file for STFILEORA simple application (config-trad-JCL.desc):

```
% config.desc :  
  
%  
  
root-skeleton =                "../trf-jcl/"  
target-proc    =                "../trf-jcl/Master-Proc"  
use-sort=mf-sort  
  
%  
  
var-dataroot = "${DATA}/"  
var-tmp      = "${TMP}/"
```

```

var-spool = "${SPOOL}/"

% Ksh heading
%
top-skeleton = "../param/top-ksh.txt"

% KSH footer
%
bottom-skeleton = "../param/bottom-ksh.txt"

% Files passed in tables
file-list-in-table = "../param/dynamic-config/File-in-table.txt"

% Suffix of translated ksh du ksh traduit
suffix-skeleton = "ksh"

% Management of FSN to keep
set-no-delete-fsn = SYSIN ( DSNUTILB ),
                    SORTIE ( ZIP390 ),
                    ENTREE ( ZIP390 ),
                    * ( DB2CMD,CSQUTIL ),
                    CFTIN ( CFTUTIL ),
                    SYSUT1 ( DUMMY ),
                    * ( ADRDSSU ).

% Management of FSN to delete
set-delete-fsn = SYSOUT ( IDCAMS ),

```

```
SYSEXEC ( CSCOLMVS ),
SYSTSIN ( * ),
SYSIN ( IDCAMS, XMFSORT, ICEMAN, SPOOL, SORT ),
SYSPUNCH ( * ),
TOOLIN ( * ),
SYSUDUMP ( * ),
SYSDBOUT ( * ),
SYSABOUT ( * ),
SYSTSPRT ( * ),
SORTLIB ( * ),
OPLIB ( * ),
STEPLIB ( * ),
JOBLIB ( * ).
```

Full example configuration

The general options `root-skeleton`, `target-proc`, `label-end`, etc. are described in the JCL Translator section of the Oracle Tuxedo Application Rehosting Workbench Reference Guide.

Listing 7-4 JCL converter configuration file for Simple Application (config-trad-JCL.desc):

```
%
% config.desc :
%
%
root-skeleton =          "../trf-jcl/"
target-proc    =          "../trf-jcl/Master-Proc"
use-sort=mf-sort
%
```

```

%
var-dataroot = "${DATA}/"
var-tmp = "${TMP}/"
var-spool = "${SPOOL}/"

%
% Ksh heading
-----

%

top-skeleton = "../param/top-ksh.txt"

% Ksh heading
-----

%
% KSH footer
-----

%
bottom-skeleton = "../param/bottom-ksh.txt"
%
% KSH footer
-----

%
% File passed in table
%
file-list-in-table = "../param/dynamic-config/File-in-table.txt"

```

```
% Suffix of translated ksh
```

```
suffix-skeleton = "ksh"
```

```
% Management of FSN to keep
```

```
set-no-delete-fsn = SYSIN ( DSNUTILB ),  
                   OUTPUT ( ZIP390 ),  
                   INPUT ( ZIP390 ),  
                   * ( DB2CMD,CSQUTIL ),  
                   CFTIN ( CFTUTIL ),  
                   SYSUT1 ( DUMMY ),  
                   * ( ADRDSSU ).
```

```
% Management of FSN to delete
```

```
set-delete-fsn = SYSOUT ( IDCAMS ),  
                SYSEXEC ( CSCOLMVS ),  
                SYSTSIN ( * ),  
                SYSIN ( IDCAMS,XMFSORT,ICEMAN,SPOOL, SORT ),  
                SYSPUNCH ( * ),  
                TOOLIN ( * ),  
                SYSUDUMP ( * ),  
                SYSDBOUT ( * ),  
                SYSABOUT ( * ),  
                SYSTSPRT ( * ),  
                SORTLIB ( * ),  
                OPLIB ( * ),  
                STEPLIB ( * ),
```

Translation steps

The Oracle Tuxedo Application Rehosting Workbench JCL Translator executes on a migration platform (Linux) and can automatically translate a single job file, a series of jobs files or the entire system content.

Set environment variables

Before executing the translator the following variables must be set:

Variable	Value	Usage
REFINEDIR	/product/art_wb11gR1/refine	the Rehosting Workbench product file structure.
PROJECT	\$(HOME)/ simpleapp	Project location.
PARAM	\$PROJECT/param	Location of configuration files.
SOURCE	\$PROJECT/source	Location of source files.
LOGS	\$PROJECT/Logs	Location of source files.

Translation commands

The following commands can be used to execute the translation. Logs file are generated in \$LOGS/trad-jcl.

Command line to translate one JCL file

To create a Korn shell script with the same name as the input file except with a .ksh suffix.

Listing 7-5 Single JCL translation script

```
cd $LOGS/trans-jcl
$REFINEDIR/refine jclz-unix -v version -s $PARAM/system.desc -c
$PARAM/config-trad-JCL.desc JCL/defvcust.jcl
```

Command line to translate a list of JCL files

To translate a list of JCL files invoke following command:

Listing 7-6 List of JCL translation script

```
cd $LOGS/trans-jcl
$REFINEDIR/refine jclz-unix -v version -s $PARAM/system.desc -c
$PARAM/config-trad-JCL.desc -f jcl-files-list
```

Command line to translate all JCL files

To translate all JCL files invoke a command:

Listing 7-7 All JCL translation script

```
cd $LOGS/trans-jcl
$REFINEDIR/refine jclz-unix -v version -s $PARAM/system.desc -c
$PARAM/config-trad-JCL.desc
```

Checking results

To check conversion results, verify that:

- All expected files are generated.
- All corresponding ksh scripts are generated.
- All procs and includes are extracted.
- There are no empty scripts.
- There are no truncated scripts.

Check the error messages; the JCL translator prints error messages encountered during the translation in the generated script. Check if any messages are present by searching for the following keywords: " UNDEFINED ", " NIL ", " UNTRANSLATED ". The complete list of error messages along with their explanation can be found in the Oracle Tuxedo Application Rehosting Workbench Reference Guide.

- Control the procedures
- Control the Batch RunTime functions.
- Check that there are no unjustified warning messages.

Using a make file

The use of makefiles when using the Rehosting Workbench is recommended because it enables you to:

- Document all (executable) processes
- Effectively manage incremental operations and thereby save time.
- Control and check the results of processes

A makefile should be placed in the source directory in which the operations are implemented at the initialization of a project.

Configuration

See [Make configuration](#) in the Cataloger user guide

JCL translation

To translate all the JCL in the \$\$SOURCE file system

From the \$\$SOURCE directory launch the command:

```
make trad_jcl
```

The JCLs are translated sequentially one by one.