Informatica® PowerExchange
10.2

Utilities Guide

Informatica PowerExchange Utilities Guide
10.2
September 2017

Publication Date: 2018-10-30

Table of Contents

Preface

This guide describes a collection of utility programs that are designed to aid maintenance of your Informatica® PowerExchange® installation.

This guide applies to the following PowerExchange products:

- PowerExchange for Adabas
- PowerExchange for CA Datacom
- PowerExchange for CA IDMS
- PowerExchange for DB2® for i5/OS
- PowerExchange for DB2 for Linux, UNIX, and Windows
- PowerExchange for DB2 for z/OS
- PowerExchange for IMS
- PowerExchange for Oracle
- PowerExchange for SQL Server
- PowerExchange for VSAM

Informatica Resources

Informatica Network

Informatica Network hosts Informatica Global Customer Support, the Informatica Knowledge Base, and other product resources. To access Informatica Network, visit https://network.informatica.com.

As a member, you can:

- Access all of your Informatica resources in one place.
- Search the Knowledge Base for product resources, including documentation, FAQs, and best practices.
- View product availability information.
- Review your support cases.
- Find your local Informatica User Group Network and collaborate with your peers.

Informatica Knowledge Base

Use the Informatica Knowledge Base to search Informatica Network for product resources such as documentation, how-to articles, best practices, and PAMs.

To access the Knowledge Base, visit https://kb.informatica.com. If you have questions, comments, or ideas about the Knowledge Base, contact the Informatica Knowledge Base team at KB_Feedback@informatica.com.

Informatica Documentation

To get the latest documentation for your product, browse the Informatica Knowledge Base at https://kb.informatica.com/_layouts/ProductDocumentation/Page/ProductDocumentSearch.aspx.

If you have questions, comments, or ideas about this documentation, contact the Informatica Documentation team through email at infa_documentation@informatica.com.

Informatica Product Availability Matrixes

Product Availability Matrixes (PAMs) indicate the versions of operating systems, databases, and other types of data sources and targets that a product release supports. If you are an Informatica Network member, you can access PAMs at https://network.informatica.com/community/informatica-network/product-availability-matrices.

Informatica Velocity

Informatica Velocity is a collection of tips and best practices developed by Informatica Professional Services. Developed from the real-world experience of hundreds of data management projects, Informatica Velocity represents the collective knowledge of our consultants who have worked with organizations from around the world to plan, develop, deploy, and maintain successful data management solutions.

If you are an Informatica Network member, you can access Informatica Velocity resources at http://velocity.informatica.com.

If you have questions, comments, or ideas about Informatica Velocity, contact Informatica Professional Services at ips@informatica.com.

Informatica Marketplace

The Informatica Marketplace is a forum where you can find solutions that augment, extend, or enhance your Informatica implementations. By leveraging any of the hundreds of solutions from Informatica developers and partners, you can improve your productivity and speed up time to implementation on your projects. You can access Informatica Marketplace at https://marketplace.informatica.com.

Informatica Global Customer Support

You can contact a Global Support Center by telephone or through Online Support on Informatica Network.

To find your local Informatica Global Customer Support telephone number, visit the Informatica website at the following link:
http://www.informatica.com/us/services-and-training/support-services/global-support-centers.

If you are an Informatica Network member, you can use Online Support at http://network.informatica.com.

CHAPTER 1

Introduction to PowerExchange Utilities

This chapter includes the following topics:

PowerExchange Utilities Overview

This guide is intended for PowerExchange administrators who run one or more utilities to manage their PowerExchange installations.

Each chapter provides some or all of the following reference information for a specific utility:

- The tasks that you can complete with the utility
- Any prerequisites to running the utility
- The systems on which the utility can run
- The syntax of the utility commands and parameters
- Detailed descriptions of the required and optional parameters
- How to issue the utility control statements on the various systems
- Examples of utility syntax by operating system, task, or both

PowerExchange includes the following utilities to help you manage your PowerExchange installation:

- createdatamaps - Data map creation utility. Use createdatamaps to generate data maps for IMS, sequential, and VSAM data sources from the Windows command line. createdatamaps is an option of the infacmd pwx command.
- DTLCCADW - Adabas PCAT file utility. Use DTLCCADW to manipulate the contents of the PCAT file.
- DTLCUIML - IMS log marker utility. Use DTLCUIML to define a marker for the IMS log-based ECCR in the IMS system log data set (SLDS).
- DTLINFO - Build information utility. Use DTLINFO to display the version, release, and build level for PowerExchange.

- DTLREXE - Remote program utility. Use DTLREXE to run programs on remote platforms.
- DTLUAPPL - Restart token utility. Use DTLUAPPL to update the CDEP file with specified applications and capture registrations.
- DTLUCBRG - Batch registration utility. Use the DTLUCBRG utility to add or modify capture registrations and extraction maps.
- DTLUCDEP - CDEP utility. Use DTLUCDEP to modify or print out the contents of the CDEP file.
- DTLUCSR2 - IDMS SR2 and SR3 records utility. Use DTLUCSR2 to determine the position of SR3 records.
- DTLUCUDB - DB2 CDC utility. Use DTLUCUDB to create a catalog snapshot to initialize the capture catalog table and to generate diagnostic information.
- DTLULCAT and DTLULOGC - IDMS log catalog utilities. Use DTLULCAT and DTLULOGC to populate the log catalog with information about the logs to process.
- DTLURDMO - Data map utility. Use DTLURDMO to migrate data maps, capture registrations and capture extraction map definitions, from one environment or location to another.
- DTLUTSK - Task control utility. Use DTLUTSK to list active tasks and stop them if required.
- EDMLUCTR - Scan and print utility for PowerExchange logs. Use EDMLUCTR to display information about the changes that are captured in the logs of the PowerExchange Logger, or to diagnose problems related to capturing changes.
- EDMXLUTL - Event marker utility. Use EDMXLUTL to create an event marker in your PowerExchange Logger.
- HOSTENT - TCP/IP Address Reporter Utility. Use HOSTENT to display the TCP/IP host name and address for a system and to diagnose problems with PowerExchange communication and licensing.
- PWXUCDCT - PowerExchange Logger for Linux, UNIX, and Windows utility. Use PWXUCDCT to manage and regenerate the CDCT file, delete log files that are not referenced by CDCT records, and print reports on the CDCT file, checkpoint files, and log files.
- PWXUCREG - PowerExchange Capture Registration Suspend Utility. Use PWXUCREG to temporarily suspend change capture processing for registered sources. Later, you can use the utility to reactivate the suspended registrations to resume change capture.
- PWXUDMX - PowerExchange Data Maps Update Time ECSA memory utility. Use PWXUDMX to allocate, display, and delete ECSA memory, which holds time stamps of the latest updates to data maps files, and to modify the use counts of a file.
- PWXUSSL - PowerExchange SSL Reporting Utlity. Use PWXUSSL to generate reports about SSL libraries and certificates on Linux, UNIX, and Windows.

PowerExchange Utilities by Operating System

The following table lists the operating systems on which each utility can run:

Utility	Linux	UNIX	Windows	z/OS	i5/OS
createdatamaps	No	No	Yes	No	No
DTLCCADW	No	No	No	Yes, for Adabas only	No
DTLCUIML	No	No	No	Yes, for IMS only	No

Utility	Linux	UNIX	Windows	z/OS	i5/OS
DTLINFO	Yes	Yes	Yes	Yes	Yes
DTLREXE	Yes	Yes	Yes	Yes	Yes
DTLUAPPL	Yes	Yes	Yes	Yes	Yes
DTLUCBRG	Yes	Yes	Yes	Yes	Yes
DTLUCDEP	Yes	Yes	Yes	Yes	Yes
DTLUCSR2	No	No	No	Yes, for IDMS CDC only	No
DTLUCUDB	Yes, for DB2 only	Yes, for DB2 only	Yes, for DB2 only	No	No
DTLULCAT and DTLULOGC	No	No	No	Yes, for IDMS Log-Based CDC only	No
DTLURDMO	Yes	Yes	Yes	Yes	Yes
DTLUTSK	Yes	Yes	Yes	Yes	Yes
EDMLUCTR	No	No	No	Yes	No
EDMXLUTL	No	No	No	Yes	No
HOSTENT	Yes	Yes	No	Yes	Yes
PWXUCDCT	Yes	Yes	Yes	No	No
PWXUCREG	No	No	No	Yes	No
PWXUDMX	No	No	No	Yes	No
PWXUSSL	Yes	Yes	Yes	No	No

PowerExchange Utilities Syntax Conventions

This guide uses the following syntax conventions for the utility commands and parameters:

- All UPPERCASE letters are used for most command names and for most parameter names, regardless of the type of platform. However, positional parameters for which you enter a specific value are shown in lowercase and italics, for example, *instance*.
- Square brackets [ ] indicate optional parameters. You can consider any parameters without these brackets to be required.
- A vertical bar | separates alternative options of which one can be entered for a parameter.
- Single braces { } enclose alternative entries. Use only one of the entries. Do not type the braces when you enter the option.

- Underlining indicates the default option for a parameter, if available.
- Italics indicate a variable or positional parameter for which the value varies.

Environment Variable Incompatibilities Between PowerExchange and PowerCenter

When PowerCenter® and PowerExchange are installed on the same Linux, UNIX, or Windows machine, in certain cases, they have conflicting requirements for the PATH and LD_LIBRARY_PATH environment variables. To run correctly in these cases, PowerExchange and PowerCenter must run in separate environments.

This requirement applies when the PowerCenter Integration Service or PowerCenter Repository Service runs on the same machine as one of the following PowerExchange components:

- PowerExchange Listener
- PowerExchange Logger for Linux, UNIX, and Windows
- PowerExchange Navigator
- Any PowerExchange utility except the createdatamaps utility

The following table describes the restrictions that apply to the PATH and LD_LIBRARY_PATH variables in the PowerExchange and PowerCenter environments:

Environment	PATH	LD_LIBRARY_PATH
PowerExchange	$INFA_HOME must not precede $PWX_HOME. Otherwise, you cannot start the PowerExchange Listener or Logger from the command line.	LD_LIBRARY_PATH must not contain an entry for PowerCenter. This requirement ensures that PowerExchange utilities pick up their libraries from $PWX_HOME only.
PowerCenter	The $PWX_HOME entry must not precede the $INFA_HOME entry.	The $LD_LIBRARY_PATH variable definition must include both $INFA_HOME and $PWX_HOME, and $INFA_HOME must precede $PWX_HOME. For example: `$INFA_HOME/server/bin:$PWX_HOME:$LD_LIBRARY_PATH`

To set the correct environment for PowerExchange or PowerCenter instances on the same machine, use one of the following strategies:

- Always start PowerExchange and PowerCenter using separate user accounts, and set the environment variables appropriately for each account.
- Run the pwxsettask.sh or pwxsettask.bat script each time you start a PowerExchange component.

Setting the PowerExchange Environment and Starting a Utility

If you run PowerExchange and PowerCenter using the same user account and need to create separate environments for PowerExchange and PowerCenter, run the pwxsettask.sh or pwxsettask.bat script to start a PowerExchange utility. The script sets the PATH and LD_LIBRARY_PATH variables correctly for PowerExchange.

To set the PowerExchange environment and start a PowerExchange utility on Linux or UNIX, issue the following command:

```
pwxsettask.sh utility_name parameter_list
```

To set the PowerExchange environment and start a PowerExchange utility on Windows, issue the following command:

```
pwxsettask utility_name parameter_list
```

parameter_list represents a list of expressions in the form *parameter=value*. When you run the script on Windows, you must include each *parameter=value* expression in double quotation marks. On Linux and UNIX, the quotation marks are optional.

For example, to start the DTLUCBRG utility on Windows:

```
pwxsettask dtlucbrg "cs=filename"
```

PowerExchange Sample JCL

When you install PowerExchange on z/OS, you install sample JCL to the *HLQ*.RUNLIB library.

If you chose to select the **Delete Install Members** option on the **Select Additional Parameters** tab of the z/OS Installation Assistant, the installation process moves the sample JCL to the *HLQ*.DTLEXPL library.

CHAPTER 2

createdatamaps - Data Map Creation Utility

This chapter includes the following topics:

createdatamaps Utility Overview

Use the createdatamaps utility to generate data maps for IMS, sequential, and VSAM data sources from the Windows command line. To determine the structure of the data map, the utility imports metadata from COBOL copybooks and IMS DBDs. The utility provides an alternative to using the PowerExchange Navigator in certain cases and allows you to generate or regenerate data maps noninteractively.

Also, for sequential and VSAM data sources on z/OS, the utility can find record ID fields. This feature is useful if the COBOL copybook includes REDEFINE statements or multiple 01-level records and includes one or more record ID fields. The utility reads the COBOL copybook and the data files that you specify in the control file and finds likely RID fields and the data values that they contain. For each record layout that matches all of the data records that have a given RID value, the utility creates a table and a record in the data map and assigns a data value to the RID field.

To run the utility, use the infacmd pwx createdatamaps command. The Informatica services or the Informatica Client must be installed on the Windows machine on which you run the command.

You can create multiple data maps per run, but they must all be of the same data source type.

Use the createdatamaps utility to create new data maps only. Do not use the utility to modify data maps that are already in use.

Note: If the command fails with a Java memory error, increase the system memory available for infacmd. To increase the system memory, set the -Xmx value in the ICMD_JAVA_OPTS environment variable. For more information, see the *Informatica Command Reference*.

Requirements and Considerations for COBOL Copybooks and DBDs

The createdatamaps utility imports metadata from COBOL copybooks and DBDs.

The following requirements and considerations apply:

- You can import metadata from only one COBOL copybook for each sequential or VSAM data map that you create. You can import metadata from one COBOL copybook for each segment in an IMS data map.
- If the copybook for a VSAM or sequential data map includes multiple 01-level records, the utility creates one record and one table in the data map for each 01-level record in the copybook. If the copybook for an IMS data map includes multiple 01-level records, the utility creates a record and a table for the first level-01 record in the copybook only.
- You can import metadata from only one DBD for each IMS data map that you create. You can optionally overlay the DBD metadata with metadata from one COBOL copybook for each segment that the DBD defines.
- For IMS data maps, if the DBD or COBOL copybook overlay includes field redefinitions, the utility creates a record only for the first combination of redefined fields.

For additional requirements and limitations that apply to finding record IDs in COBOL copybooks, see "Requirements and Limitations for Finding RID Fields" on page 47.

createdatamaps Command Syntax

The infacmd pwx createdatamaps command uses the following syntax:

```
infacmd pwx createdatamaps

[<-pwxLocation|-loc> pwx_location]

[<-pwxUserName|-pun> pwx_user_name]

[<-pwxPassword|-ppd> pwx_password]

[<-pwxEncryptedPassword|-epwd> pwx_encrypted_password]

[<-datamapOutputDir|-dod> datamap_output_directory]

[<-replace|-r> replace_existing_datamaps

<-controlFile|-cf> file_path_for_control_file

[<-logFile|-lf> file_path_for_log_file]

[<-verbosity|-v> logging_verbosity]
```

The following table describes infacmd pwx createdatamaps options and arguments:

Option	Argument	Description
-pwxLocation -loc	pwx_location	Optional. The location of the data source as specified in a NODE statement in the PowerExchange dbmover configuration file. If pwxLocation is not specified, the createdatamaps utility accesses the copybook and DBD metadata on the local file system. If you configure the control file to find record IDs, pwxLocation is required.
-pwxUserName -pun	pwx_user_name	Optional. The user ID for connecting to the PowerExchange Listener, if pwxLocation is specified.
-pwxPassword -ppd	pwx_password	Optional. Password for connecting to the PowerExchange Listener, if pwxLocation is specified. Instead of a password, you can enter a valid PowerExchange passphrase. Passphrases for accessing a PowerExchange Listener on z/OS can be from 9 to 128 characters in length and can contain the following characters: - Uppercase and lowercase letters - The numbers 0 to 9 - Spaces - The following special characters: `' - ; # \ , . / ! % & * ( ) _ + { } : @ | < > ?` **Note:** The first character is an apostrophe. Passphrases cannot include single quotation marks (‘), double quotation marks (“), or currency symbols. If a passphrase contains spaces, you must enclose it with double-quotation marks ("), for example, `"This is an example passphrase"`. If a passphrase contains special characters, you must enclose it with triple double-quotation characters ("""), for example, `"""This passphrase contains special characters ! % & *."""`. If a passphrase contains only alphanumeric characters without spaces, you can enter it without delimiters. **Note:** On z/OS, a valid RACF passphrase can be up to 100 characters in length. PowerExchange truncates passphrases longer than 100 characters when passing them to RACF for validation. To use passphrases, ensure that the PowerExchange Listener runs with a security setting of SECURITY=(1,N) or higher in the DBMOVER member. For more information, see "SECURITY Statement" in the *PowerExchange Reference Manual*.
-pwxEncryptedPassword -epwd	pwx_encrypted_password	Optional. Encrypted password for connecting to the PowerExchange Listener, if pwxLocation is specified. If the PowerExchange Listener runs on a z/OS or i5/OS system, you can enter an encrypted PowerExchange passphrase instead of an encrypted password. Do not encrypt a passphrase that contains characters that are not valid, such as double-quotation marks, single quotation marks, or currency symbols.

Option	Argument	Description
-datamapOutputDir -dod	datamap_output_directory	Optional. The local file directory to which to write the output data maps. Default is the current working directory.
-replace -r	replace_existing_datamaps	Optional. Specifies whether to replace existing data maps. If replace=Y, replaces any data maps in datamap_output_directory that have the same name as the data map that you are creating. If replace=N, skips the creation of a data map if a data map with the same name already exists in datamap_output_directory. Default is N.
-controlFile -cf	file_path_for_control_file	Required. Path and file name of the control file that controls data map generation.
-logFile -lf	file_path_for_log_file	Optional. Path and file name of the output log file. Default is STDOUT.
-verbosity -v	logging_verbosity	Optional. Verbosity for log files. Default is INFO. Valid values: - DEBUG. Most detailed logging. Might show stack traces. - INFO. Informational messages. - WARN. Indicates a potential problem. - ERROR. Indicates a failure. Processing continues. - FATAL. Indicates a fatal condition. Process terminates.

The PowerExchange node name and credentials are optional. If you do not include the pwxLocation option, the command accesses the local file system directly to read metadata. In this case, PowerExchange does not need to be installed on the machine on which you run createdatamaps.

Creating, Editing, and Testing a Data Map

To create, edit, and test a data map, perform the following high-level tasks:

1. If PowerExchange is not installed on the local machine, where you will run the createdatamaps command, copy the copybooks and DBDs to the local machine.

   If you configure the control file to find record ID fields, the copybooks must reside on the source z/OS system, and PowerExchange must be installed on the machine on which you run the createdatamaps command.
2. Create the control file.

   You can create a new file using an XML editor. Or you can open a sample control file in an XML editor, rename the file, and edit the file as needed.
3. Run the infacmd pwx createdatamaps command from the command line on any machine that can access the Informatica domain.
4. Review the output log. If errors occur, correct them and run the command again.
5. Copy the data map to the PowerExchange Navigator machine.

6. Open the data map in the PowerExchange Navigator.

   Tip: If the PowerExchange Navigator is already open when you copy the data map, select **File** > **Refresh** to refresh the list of data maps.

7. If required, edit the data map.

   For example, you might need to delete unused records, assign record ID values, or assign other properties that you cannot assign with the createdatamaps utility.

   If you configured the control file to find record ID fields, review the record types that the utility created and the record ID values that the utility assigned. Edit these results if necessary.

8. Perform a database row test.
9. Make additional edits and perform additional database row tests, as needed.
10. Send the data map to the remote node.

Control Files for Data Map Generation

A control file is an XML file that defines the options and values that the createdatamaps utility uses to create one or more data maps. A control file provides a simple way to define parameters and helps to make the bulk import creation process easily repeatable.

When you run the createdatamaps command, provide the name of the control file as a parameter.

Each control file must conform to the schema that is installed with the Informatica Client and with Informatica services. For more information, see "Schema File for Describing Control Files" on page 27 and "Schema File Reference" on page 28.

Each control file can specify only one data source type.

After you create the control file and run the createdatamaps command, you might need to edit the data map in the PowerExchange Navigator.

Example control files are provided with the product. For more information, see "Examples" on page 52.

Rules for Control Files

Observe the following rules when you create a control file:

- All XML elements are case sensitive. For each element, follow the capitalization that is defined in the schema.
- Elements must appear in the control file in the order in which they are defined in the schema. Your XML editor should indicate the allowable elements at each point in the control file.
- Observe the properties that are defined for each element in the schema file. For each element, the schema file might define the following properties, depending on the data type:
  - Name
  - Cardinality, that is, the minimum and maximum number of occurrences
  - Type
  - Valid values
  - Minimum and maximum length

Control File Structure

The XML control files for sequential, VSAM, and IMS data maps have hierarchies that allow certain elements at specific points.

Each control file includes the following elements:

- DatamapGeneration root element. Required.
- imsGen, seqGen, or vsamGen element that is a child of the DatamapGeneration element. Required.
- Global elements that apply to all data maps that the control file defines. These global elements are children of the imsGen, seqGen, or vsamGen element. Optional.
- DatamapInstances element. This element is a child of the imsGen, seqGen, or vsamGen element. Required.
- One imsDatamapInstance, seqDatamapInstance, or vsamDatamapInstance element for each data map that the control file defines. These elements are children of the DatamapInstances element. At least one element instance is required.

For each data map, the imsDatamapInstance, seqDatamapInstance, or vsamDatamapInstance element inherits and optionally overrides the global settings.

The following control file hierarchies do not show the required syntax, whether the element is required or optional, the number of times the element can occur, or whether a choice of only one of multiple elements can appear. For this information, see "Schema File Reference" on page 28 or see the DatamapGeneration.xsd schema file.

Control File Hierarchy for Sequential Data Maps

Use the following control file hierarchy for sequential data maps:

```
DatamapGeneration
   seqGen
      globalCopybookParserConfig
         startColumn
         endColumn
         maxRedefines
         decimalPointIsComma
      cacheConfig
         cachePath
         flushDataMode
      globalGenConfig
         schemaName
         datamapName
         datamapRecordName
         findRecordIds
         excludeUnmatchedRecords
       globalSeqProperties
         skipRecordCount
         ridConfig
            readRecordLimit
            recordTypeLimit
            fieldWidth
            matchFieldWidth
            fieldOffset
         seqFileName
            zosPath
            windowsPath
            as400Pth
            unixPath
      datamapInstances
         seqDatamapInstance
            genConfig
               schemaName
               datamapName
               datamapRecordName
```

```
            findRecordIds
            excludeUnmatchedRecords
         importCopybookDetails
            filePath
               zosPath
               windowsPath
               as400Path
               unixPath
            parserConfig
               startColumn
               endColumn
               maxRedefines
               decimalPointIsComma
         datamapProperties
            seqFileName
               zosPath
               windowsPath
               as400Pth
               unixPath
            skipRecordCount
            ridConfig
               readRecordLimit
               recordTypeLimit
               fieldWidth
               matchFieldWidth
               fieldOffset
```

Control File Hierarchy for VSAM Data Maps

Use the following control file hierarchy for VSAM data maps:

```
DatamapGeneration
   vsamGen
      globalCopybookParserConfig
         startColumn
         endColumn
         maxRedefines
         decimalPointIsComma
      cacheConfig
         cachePath
         flushDataMode
      globalGenConfig
         schemaName
         datamapName
         datamapRecordName
         findRecordIds
         excludeUnmatchedRecords
      globalVsamProperties
         skipRecordCount
         ridConfig
            readRecordLimit
            recordTypeLimit
            fieldWidth
            matchFieldWidth
            fieldOffset
         vsamFileName
            zosPath
            windowsPath
            as400Path
            unixPath
      globalMapType
      datamapInstances
         vsamDatamapInstance
            genConfig
               schemaName
               datamapName
               datamapRecordName
               findRecordIds
               excludeUnmatchedRecords
            importCopybookDetails
```

```
                  filePath
                     zosPath
                     windowsPath
                     as400Path
                     unixPath
                  parserConfig
                     startColumn
                     endColumn
                     maxRedefines
                     decimalPointIsComma
               datamapProperties
                  vsamFileName
                     zosPath
                     windowsPath
                     as400Path
                     unixPath
                  skipRecordCount
                  ridConfig
                     readRecordLimit
                     recordTypeLimit
                     fieldWidth
                     matchFieldWidth
                     fieldOffset
```

Control File Hierarchy for IMS Data Maps

Use the following control file hierarchy for IMS data maps:

```
DatamapGeneration
   imsGen
      globalCopybookParserConfig
         startColumn
         endColumn
         maxRedefines
         decimalPointIsComma
      globalGenConfig
         schemaName
         datamapName
         datamapRecordName
         createTablesForHierPath
      globalImsProperties
         mapType
         imsSSID
         pcbNumber
         psbName
         pcbName
      datamapInstances
         imsDatamapInstance
            genConfig
               schemaName
               datamapName
               datamapRecordName
               createTablesForHierPath
            importDBDDetails
               filePath
                  zosPath
                  windowsPath
                  as400Path
                  unixPath
            overlayDetails
               nativeRecordName
               overlayCopybookDetails
                   filePath
                     zosPath
                     windowsPath
                     as400Path
                     unixPath
                 parserConfig
                    startColumn
                     endColumn
```

```
            maxRedefines
            decimalPointIsComma
datamapProperties
   mapType
   imsSSID
   pcbNumber
   psbName
   pcbName
```

Schema File for Describing Control Files

Each control file must conform to the schema that is installed with the Informatica Client and with Informatica services.

The schema file, named DatamapGeneration.xsd, is provided in the following directories on the machines where the Informatica Client and the Informatica services are installed:

- *Informatica_client_installation_directory*\clients\DeveloperClient\osgi_mf_plugins\jars\resources\
- *Informatica_services_installation_directory*\pwxmfplugins\resources\

The schema consists of definitions of complex types and elements, where a complex type consists of multiple elements.

Complex Type Definitions in the Schema File

For each complex type, the schema file includes the following information:

- Name
- Base type, for complex types that inherit properties from a base type
- Definitions of each child element

Element Definitions in the Schema File

For each element, the schema file might include the following information, depending on the data type:

- Name
- Cardinality, that is, the minimum and maximum number of occurrences
- Type
- Documentation
- Valid values
- Minimum and maximum length

Using the Schema File to Determine the Control File Hierarchy

You can open the schema file in an XML editor to determine the allowable hierarchy of elements in a control file. You can also derive this information from the schema file.

For example, the root element in the schema is DatamapGeneration. The definition of the DatamapGeneration complex type specifies a choice of one of the following child elements:

- seqGen, of type SEQGen
- vsamGen, of type VSAMGen
- imsGen, of type IMSGen

The definition of the SEQGen type specifies that it is an extension of the GenBase base type, which includes the following elements: globalCopybookParserConfig and cacheConfig. Elements of type SEQGen include the elements defined by the GenBase type in addition to the following elements:

- globalGenConfig, of type SEQGenConfig
- globalSeqProperties, of type SEQDatamapProperties
- datamapInstances, which consists of one element, seqDatamapInstance, that can occur one or more times

The control file for generating SEQ data maps therefore has the following high-level structure:

```
DatamapGeneration
     seqGen
           globalCopybookParserConfig (optional)
           cacheConfig (optional)
           globalGenConfig (optional)
           globalSeqProperties (optional)
           datamapInstances (required)
                seqDatamapInstance (at least one required)
```

You can continue in this manner to determine the complete hiearchy for SEQ control files, as well as for VSAM and IMS control files.

Schema File Reference

The following topics describe each complex type and element in the schema file, including its type, allowed values, cardinality, and description.

DatamapGeneration Complex Type

The DatamapGeneration complex type defines the top-level element in the control file.

The DatamapGeneration complex type includes the following elements:

Choice (cardinality = 1):

- SeqGen
- VSAMGen
- IMSGen

Exactly one occurrence of SeqGen, VSAMGen, or IMSGen is required.

seqGen

Element for defining sequential data maps.

Type = SEQGen

vsamGen

Element for defining VSAM data maps.

Type = VSAMGen

imsGen

Element for defining IMS data maps.

Type = IMSGen

The DatamapGeneration complex type includes the required xmlSchemaVersion attribute.

xmlSchemaVersion

Latest schema version with which the XML instance is compatible.

Value = 1.0

GenBase Complex Type

The GenBase complex type is the base type for the complex types that define the sequential, VSAM, and IMS data maps.

The GenBase complex type is the base type for the following complex types:

- SEQGen
- VSAMGen
- IMSGen

The GenBase complex type includes the following elements:

- globalCopybookParserConfig
- cacheConfig

globalCopybookParserConfig

Default copybook parser configuration properties that are applied at a global level.

Type = CopybookParserConfig

Cardinality = 0 to 1

cacheConfig

Controls the data cache on disk.

Type = CacheConfig

Cardinality = 0 to 1

DataConfigBase Complex Type

The DataConfigBase complex type defines configuration properties for reading data records. It is the base type for the RIDConfig complex type.

The DataConfigBase complex type includes the following elements:

readRecordLimit

Maximum number of data records to read from each data file. A value of 0 means that no limit exists. The utility reads all records.

Type = integer

Range = 0 to 2147483647

Default = 10000

Cardinality = 0 to 1

SEQGen Complex Type

The SEQGen complex type defines one or more data maps for sequential data sources.

The SEQGen complex type is an extension of the GenBase complex type. SEQGen extends GenBase with the following elements:

- globalGenConfig
- globalSeqProperties
- datamapInstances

globalGenConfig

Default generator configuration properties that are applied at a global level.

Type = SEQGenConfig

Cardinality = 0 to 1

globalSeqProperties

Global defaults for SEQ data map properties.

Type = SEQDatamapProperties

Cardinality = 0 to 1

datamapInstances

Element that contains one or more occurrence of SEQ data map instance details.

Type = Container of SeqDatamapInstance elements

Cardinality = 1

VSAMGen Complex Type

The VSAMGen complex type defines one or more data maps for VSAM data sources.

The VSAMGen complex type extends the GenBase complex type with the following elements:

- globalGenConfig
- globalVSAMProperties
- globalMapType
- datamapInstances

globalGenConfig

Default generator configuration properties that are applied at a global level.

Type = VSAMGenConfig

Cardinality = 0 to 1

globalVsamProperties

Global defaults for VSAM data map properties.

Type = VSAMDatamapProperties

Cardinality = 0 to 1

globalMapType

Global value for data map access method type.

Valid values: ESDS, KSDS, RRDS

Default: KSDS

Cardinality = 0 to 1

datamapInstances

Element that contains one or more occurrence of VSAM data map instance details.

Type = Container of vsamDatamapInstance elements

Cardinality = 1

IMSGen Complex Type

The IMSGen complex type defines one or more data maps for IMS data sources.

The IMSGen complex type extends the GenBase complex type with the following elements:

- globalGenConfig
- globalIMSProperties
- datamapInstances

globalGenConfig

Default generator configuration properties that are applied at a global level.

Type = IMSGenConfig

Cardinality = 0 to 1

globalIMSProperties

Global defaults for IMS data map properties.

Type = IMSDatamapProperties

Cardinality = 0 to 1

datamapInstances

Element that contains one or more occurrences of IMS data map instance details.

Type = Container of IMSDatamapInstance elements

Cardinality = 1

GenConfigBase Complex Type

The GenConfigBase complex type is the base type for configuration properties for the data map generation process. For example, these properties affect how the generated data maps or their records or tables are named.

The GenConfigBase complex type is the base type for the following complex types:

- SEQGenConfig
- VSAMGenConfig
- IMSGenConfig

The GenConfigBase complex type includes the following elements:

- schemaName

- datamapName
- datamapRecordName

schemaName

Data map schema name.

Type = string, length = 1 to 10

Default = SCHEMA

Cardinality = 0 to 1

datamapName

Data map name. Used as a data map name prefix when used at a global level. When multiple data maps are created, an integer is appended to the prefix to form the data map name.

Type = string, length = 1 to 10

Default = MAP

Cardinality = 0 to 1

datamapRecordName

Prefix for names of records in the data map. When multiple records are created, an integer is appended to the prefix to form the record name.

Type = string, length = 1 to 256

Default = name of native import object name, such as a COBOL 01-level record name or a DBD segment name.

Cardinality = 0 to 1

SEQGenConfig Complex Type

The SEQConfigBase complex type defines configuration properties for the data map generation process for sequential data sources.

The SEQGenConfig complex type is an extension of the GenConfigBase complex type. It includes the following additional element:

- findRecordIds
- excludeUnmatchedRecords

findRecordIds

Enables or disables discovery of record IDs.

Type = boolean

Default = false

Cardinality = 0 to 1

excludeUnmatchedRecords

If true, generates a data map record only for those layouts for which the utility finds a valid data record ID.

Type = boolean

Default = false

Cardinality = 0 to 1

VSAMGenConfig Complex Type

The VSAMGenConfig complex type defines configuration properties for the data map generation process for VSAM data sources.

The VSAMGenConfig complex type is an extension of the GenConfigBase complex type. It includes the following additional element:

- findRecordIds
- excludeUnmatchedRecords

findRecordIds

Enables or disables discovery of record IDs.

Type = boolean

Default = false

Cardinality = 0 to 1

excludeUnmatchedRecords

If true, generates a data map record only for those layouts for which the utility finds a valid data record ID.

Type = boolean

Default = false

Cardinality = 0 to 1

IMSGenConfig Complex Type

The IMSConfigBase complex type defines configuration properties for the data map generation process for IMS data sources.

The IMSGenConfig complex type is an extension of the GenConfigBase complex type. It includes the following additional elements:

- CreateTablesForHierPath

CreateTablesForHierPath

Whether to create complex tables that include all of the records in the hierarchical path.

Type = boolean

Default = true

Cardinality = 0 to 1

DatamapPropertiesBase Complex Type

The DatamapPropertiesBase complex type defines common data map properties.

The DatamapPropertiesBase complex type is the base type for the following complex types:

- SEQDatamapProperties
- VSAMDatamapProperties

- IMSDatamapProperties

The DatamapPropertiesBase complex type does not include any elements.

ParserConfigBase Complex Type

The ParserConfigBase complex type includes common parser configuration properties.

The ParserConfigBase complex type is the base type for the following complex types:

- CopybookParserConfig

ParserConfigBase does not include any elements.

CopybookParserConfig Complex Type

The CopybookParserConfig complex type includes parser configuration properties for copybooks.

The CopybookParserConfig complex type is an extension of the ParserConfigBase complex type and includes the following elements:

- startColumn
- endColumn
- maxRedefines
- decimalPointIsComma

startColumn

Starting column of data to be parsed.

Type = integer, range = 1 to 999

Default = 7

Cardinality = 0 to 1

endColumn

Ending column of data to be parsed.

Type = integer, range = 1 to 999

Default = 72

Cardinality = 0 to 1

maxRedefines

Maximum number of record layouts to generate from REDEFINE statements in the copybook. If the copybook overlays a DBD, maxRedefines defaults to 1 and cannot be overridden. If the copybook for a sequential or VSAM data map contains multiple 01-level records, maxRedefines applies to each 01-level record in the copybook.

Type = integer, range = 1 to 4096

Default = 10000 if the control file is configured to find RID fields, otherwise default = 1

Cardinality = 0 to 1

decimalPointIsComma

Defines whether a comma represents a decimal point character in fields that contain noninteger numbers. Set this value to match the value of the DECPOINT statement in the DBMOVER configuration file.

Type = boolean

Default = false

Cardinality = 0 to 1

CacheConfig Complex Type

The CacheConfig complex type controls the data cache on disk. You can configure CacheConfig properties at a global level but not a data map instance level.

The CacheConfig complex type includes the following elements:

cachePath

The full path to the folder for temporary working files. The cache path is written to the message log.

Type = string

Cardinality = 0 to 1

Default = *current_working_directory*/temp

flushDataMode

Specifies when to flush the cache of the data records that were downloaded from the z/OS system.

Type = string

Valid values:

- e - Flush the cache when the createdatamaps utility finishes.
- d - Flush the cache after each data map is created.

The default value of "e" allows data to be shared by multiple data map generations during one createdatamaps session.

RIDConfig Complex Type

The RIDConfig complex type defines criteria for finding record ID (RID) fields.

The RIDConfig complex type is an extension of the DataConfigBase complex type. RIDConfig extends DataConfigBase with the following elements:

- recordTypeLimit
- fieldWidth
- matchFieldWidth
- fieldOffset

The RIDConfig complex type includes the following elements:

recordTypeLimit

Maximum number of record types in the data file.

Type = integer, range = 1 to 2147483647

Default = 10

Cardinality = 0 to 1

fieldWidth

Maximum width or exact width in bytes of an RID field, depending on matchFieldWidth.

Type = integer, range = 1 to 2147483647

Default = 4

Cardinality = 0 to 1

matchFieldWidth

If true, the RID field must exactly match the fieldWidth value. If false, fieldWidth represents the maximum field width.

Type = boolean

Default = false

Cardinality = 0 to 1

fieldOffset

Byte offset of the RID field from the start of the record, beginning at offset 0. If not specified, the utility finds the RID field.

Type = integer, range = -1 to 2147483647

Default = -1, meaning not specified

Cardinality = 0 to 1

FilePath Complex Type

The FilePath complex type defines the path and name of a file on the file system. The file path can be absolute or relative to the current directory.

The FilePath complex type includes the following elements:

- Choice (cardinality = 1):
  - zosPath
  - windowsPath
  - as400Path
  - unixPath

zosPath

File path on a z/OS system.

Type = string, length = 1 to 256

Cardinality = 0 to 1

windowsPath

File path on a Windows system.

Type = string, length = 1 to 1024

Cardinality = 0 to 1

as400Path

File path on an i5/OS system.

Type = string, length = 1 to 256

Cardinality = 0 to 1

unixPath

File path on a UNIX system.

Type = string, length = 1 to 1024

Cardinality = 0 to 1

ImportMetadataBase Complex Type

The ImportMetadataBase complex type defines common metadata import properties.

ImportMetadataBase is the base type for the following complex types:

- CopybookImportMetadata
- DBDImportMetadata

ImportMetadataBase includes the following elements:

- filePath

filePath

File system location of the metadata source.

Type = FilePath

Cardinality = 1

CopybookImportMetadata Complex Type

The CopybookImportMetadata complex type defines common metadata import properties for copybooks.

CopybookImportMetadata extends the ImportMetadataBase complex type with the following elements:

- parserConfig

parserConfig

Fields related to copybook parser configuration.

Type = CopybookParserConfig

Cardinality = 0 to 1

DBDImportMetadata Complex Type

The DBDImportMetadata complex type defines common metadata import properties for DBDs.

The DBDImportMetadata complex type is an extension of the ImportMetadataBase complex type. It does not define any additional elements.

OverlayMetadata Complex Type

The OverlayMetadata complex type defines properties for overlaying metadata, such as overlaying DBD metadata with COBOL copybook metadata.

The OverlayMetadata complex type includes the following elements:

- nativeRecordName
- overlayCopybookDetails

nativeRecordName

Native name of the data map record to overlay, such as the name of the DBD segment.

Type = string, length = 1 to 256

Cardinality = 1

overlayCopybookDetails

Details about the copybook that overlays the data map record.

Type = CopybookImportMetadata

Cardinality = 1

DatamapInstanceBase Complex Type

The DatamapInstanceBase complex type defines properties for a data map instance.

DatamapInstanceBase is the base type for the following complex types:

- SEQDatamapInstance
- VSAMDatamapInstance
- IMSDatamapInstance

The DatamapInstanceBase complex type does not define any elements.

SEQDatamapProperties Complex Type

The SEQDatamapProperties complex type defines common data map properties for sequential data sources.

The SEQDatamapProperties complex type is an extension of the DatamapPropertiesBase complex type. It extends DatamapPropertiesBase with the following elements:

- seqFileName
- skipRecordCount
- ridConfig

seqFileName

Full path and file name of the sequential data set or flat file that is a data source.

Type = FilePath

Default = current Windows path with value of "file.dat"

Cardinality = 0 to 1

skipRecordCount

Specifies the number of initial records to skip when reading the data file

Type = integer, range = 0 to 2147483647

Default = 0

Cardinality = 0 to 1

ridConfig

Defines record ID (RID) configuration parameters.

Type = RIDConfig

Cardinality = 0 to 1

SEQDatamapInstance Complex Type

The SEQDatamapInstance complex type defines properties for SEQ data maps.

SEQDatamapInstance extends the DatamapInstanceBase complex type with the following elements:

- genConfig
- importCopybookDetails
- datamapProperties

genConfig

SEQ generator configuration applied at the instance level.

Type = SEQGenConfig

Cardinality = 0 to 1

importCopybookDetails

Copybook definitions for importing data map instance metadata.

Type = CopybookImportMetadata

Cardinality = 1 to unbounded

datamapProperties

Data map properties at the instance level.

Type = SEQDatamapProperties

Cardinality = 0 to 1

VSAMDatamapProperties Complex Type

The VSAMDatamapPropertiesBase complex type defines common data map properties for VSAM data sources.

The VSAMDatamapProperties complex type is an extension of the DatamapPropertiesBase complex type. It extends DatamapPropertiesBase with the following elements:

- vsamFileName
- skipRecordCount
- ridConfig

vsamFileName

Fully qualified data set name of the of VSAM source file.

Type = FilePath

Default = Current Windows path with value of "file.dat"

Cardinality = 0 to 1

skipRecordCount

Specifies the number of initial records to skip when reading the data file

Type = integer, range = 0 to 2147483647

Default = 0

Cardinality = 0 to 1

ridConfig

Defines RID configuration parameters.

Type = RIDConfig

Cardinality = 0 to 1

VSAMDatamapInstance Complex Type

The VSAMDatamapInstance complex type defines properties for VSAM data maps.

SEQDatamapInstance extends the DatamapInstanceBase complex type with the following elements:

- genConfig
- importCopybookDetails
- datamapProperties

genConfig

VSAM generator configuration applied at the instance level.

Type = VSAMGenConfig

Cardinality = 0 to 1

importCopybookDetails

Copybook definitions for importing data map instance metadata.

Type = CopybookImportMetadata

Cardinality = 1 to unbounded

datamapProperties

Data map properties at the instance level.

Type = VSAMDatamapProperties

Cardinality = 0 to 1

IMSDatamapProperties Complex Type

The IMSDatamapPropertiesBase complex type defines common data map properties for IMS data sources.

The IMSDatamapProperties complex type extends the DatamapPropertiesBase complex type with the following elements:

- mapType
- imsSSID
- Choice (cardinality = 0 to 1, default = 1):
  - pcbNumber
  - psbName, pcbName

mapType

Data map type. Represents IMS DL/1 batch or IMS ODBA.

Type = string, valid values = ODBA, DL1

Default = DL1

Cardinality = 0 to 1

imsSSID

IMS subsystem ID.

Type = string, maximum length = 4

Cardinality = 0 to 1

pcbNumber

PCB number for the database. Optional field for DL/1 data maps at the instance level.

Type = string

Default = 1

Cardinality = 0 to 1

psbName

PSB name. Optional field for ODBA data maps at the instance level.

Type = string

Default = PSBNAME

Cardinality = 0 to 1

pcbName

For the specified PSB, the named PCB that references the specified DBD. Optional field for ODBA data maps at the instance level.

Type = string

Default = PCBNAME

Cardinality = 0 to 1

IMSDatamapInstance Complex Type

The IMSDatamapInstance complex type defines properties for IMS data maps.

IMSDatamapInstance extends the DatamapInstanceBase complex type with the following elements:

- genConfig
- importDBDDetails
- overlayDetails
- datamapProperties

genConfig

IMS generator configuration applied at an instance level.

Type = IMSGenConfig

Cardinality = 0 to 1

importDBDDetails

DBD definitions for importing data map instance metadata.

Type = DBDImportMetadata

Cardinality = 1

overlayDetails

Metadata to overlay imported records.

Type = OverlayMetadata

Cardinality = 0 to unbounded

datamapProperties

Data map properties at the instance level.

Type = IMSDatamapProperties

Cardinality = 0 to 1

Log File for Data Maps Creation Utility

The createdatamaps utility writes informational, warning, and error messages to the log file that you specify when you run the command. If you do not specify a log file, the output is sent to the console.

After creating a data map, the utility writes informational messages to the log file. The messages report basic statistics such as the number of records and fields per data map.

Before executing the control file, the utility checks for syntax errors. If the utility encounters an error, it stops after the first error and reports the error in the log file.

The createdatamaps utility checks for the following kinds of syntax errors:

- Missing mandatory elements
- General syntax errors, such as unexpected property names
- Illegal values, where the element definition provides enumerated values

- Invalid characters in names
- Maximum length of name fields exceeded

Each message in the log file might have one of the following prefixes, depending on the module that reported it:

Prefix	Module Description
PWXCMD	Infacmd user interface
PWXLog	Log file implementation for data map generation
MDO	Component that interprets the XML control file and acts upon it
PWXNative	PowerExchange connection support for communcation with the PowerExchange Listener
MDAdapter	Component that reads the metadata
Parser	Component that parses the COBOL, DBD, or VSAM metadata text files and turns them into Informatica object models
JDMX2	Component that interprets the Informatica object models and writes them to PowerExchange data maps

For examples of log files, see "Examples" on page 52.

For error message descriptions, see the *Informatica Message Reference*.

JAXB Error Messages

In addition to the error messages that PowerExchange generates, the createdatamaps log file might include Java Architecture for XML Binding (JAXB) error messages. These messages might appear standalone or wrapped in PowerExchange message MDO_34611.

The following table shows some of the eror conditions that result in a JAXB error message:

Error Condition	JAXB Error Message
An element is spelled incorrectly	The element type "*element_name*" must be terminated by the matching end-tag "</*element_name*".
An incorrect data type or value is specified for an element.	Not a Valid Datatype: *value* cvc-datatype-valid.1.2.1: '*value*' is not a valid value for 'Datatype name' cvc-type.3.1.3: The value '*value*' of element '*element_name*' is not valid.
An end tag is missing.	The element type "*element_name*" must be terminated by the matching end-tag "</*element_name*>".
Data is specified outside of the element tags.	Element '*element_name*' cannot have character [children], because the type's content type is element-only.
The DatamapGeneration tag does not include the xmlSchemaVersion and xmlns attributes.	Unexpected element (uri:"*uri*", local:"*element_name*"). Expected elements are *element_list*.

Error Condition	JAXB Error Message
An element is specified in an incorrect location.	cvc-complex-type.2.4.a: Invalid content was found starting with element '*element_name*'. One of '{*element_list*}' is expected.
The order of elements is rearranged.	Invalid content was found starting with element '*element1_name*'. No child element is expected at this point.. *element2_name*
A reserved character, such as a greater-than or less-than sign, is specified as part of a value for an element.	Value '*value*' is not facet-valid with respect to pattern '[a-zA-Z][a-zA-Z0-9_]*' for type '*#type*'.. XML parser message: SEVERITY: 2, MESSAGE: cvc-type.3.1.3: The value '*value*' of element '*element_name*' is not valid..

Redefinitions and Record IDs in COBOL Copybooks

COBOL copybooks can define multiple record layouts. The layout of each data record is often determined by a record ID (RID) field.

The following types of redefinitions in a COBOL copybook define multiple record layouts:

- REDEFINE statements
- Multiple 01 levels that define multiple record types

The createdatamaps utility can generate data maps for COBOL copybooks that include REDEFINE statements, multiple 01 levels, or both. And you can configure the control file so that the utility finds RID fields and associates RID values with data records.

How the createdatamaps Utility Creates Records for Redefined Fields and Groups

The createdatamaps utility builds an internal model that represents all of the combinations of redefined fields and groups.

For example, consider the following record from a COBOL copybook:

```
01 MASTER-REC.
      05 MASTER-DATE.
            07 some fields
      05 MASTER-DOB REDFEINES MASTER-DATE.
            07 some fields
      05 OTHER-DATE.
            07 some fields
      05 OTHER-DOB REDFEINES OTHER-DATE.
            07 some fields
```

The utility creates an internal model with four layouts. the layouts correspond to the following combinations of fields:

- MASTER-DATE / OTHER-DATE
- MASTER-DATE / OTHER-DOB
- MASTER-DOB / OTHER-DATE
- MASTER-DOB / OTHER-DOB

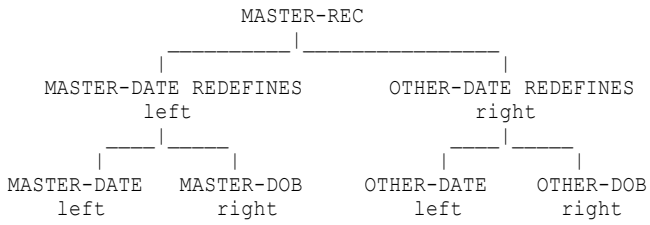
The utility creates a record and table for each of the first *maxRedefines* layouts in the model. For more information, see "Use of the maxRedefines Element to Limit the Number of Records" on page 45.

Use of the maxRedefines Element to Limit the Number of Records

Use the maxRedefines element to limit the number of layouts that the createdatamaps utility generates when it parses a copybook that contains REDEFINE statements. The utility writes only the first *maxRedefines* records to the data map.

If the copybook includes multiple 01 levels, MaxRedefines applies to each 01 level.

To determine the order of records, the utility builds a hierarchical internal model, such as the following one:



When the utility begins reading records from the COBOL copybook, it selects the left-most redefine in each branch. In this example, the utility selects the combination of MASTER-DATE / OTHER-DATE. The utility then progresses through the redefines of the right-most branch, that is, MASTER-DATE / OTHER-DOB, until all combinations are added to the model. The utility returns to the next redefine in the left branch, that is, MASTER-DOB. The utility combines MASTER-DOB with each redefine from the right branch, that is, MASTER-DOB / OTHER-DATE and then MASTER-DOB / OTHER-DOB.

Because this process can lead to a large number of records, you can include the maxRedefines element in the control file to limit the number of layouts that the utility processes to the first *maxRedefines* layouts.

After you create the data map, you can edit the data map in the PowerExchange Navigator to delete unwanted records and tables. Also, if you configured the control file to find RID fields, you can view the assigned RID values.

Redefinitions Without Record IDs in COBOL Copybooks

A COBOL copybook might include REDEFINE statements but no RID field. In this case, the createdatamaps utility defines one record and one table for each possible combination of redefined fields, up to the maximum number that the maxRedefines element specifies. By default, maxRedefines specifies 1. For IMS data maps, the default of 1 always applies.

If a copybook for a sequential or VSAM data map contains multiple 01-level records, MaxRedefines applies to each 01-level record in the copybook.

After you create the data map, you can edit the data map in the PowerExchange Navigator as needed to delete unwanted records and tables.

Redefinitions with RID Fields in COBOL Copybooks

If a COBOL copybook includes redefinitions and RID fields, you can configure the control file to find RID fields. The utility reads the COBOL copybook and the data files that you specify in the control file to find likely RID fields and the data values that they contain.

Alternatively, if you know the location of the RID field, you can specify it in the fieldOffset element in the control file. The createdatamaps utility validates the fieldOffset value that you provide, reads RID values from sample data, and matches record layouts with the data records.

By default, the utility defines one record and one table in the data map for each layout that the copybook defines, up to the maximum number that the maxRedefines element specifies. Also, for each layout that matches all of the data records that have a given RID value, the utility assigns an RID value to the record in the data map.

Alternatively, you can configure the utility to create data map records only for those layouts that match all of the data records that have a given RID value.

Example COBOL Copybooks with RID Fields

The following examples show COBOL copybooks with redefinitions and RID fields. For both of these examples, you can configure the createdatamaps utility fo find the RID field, or use the field offset that you specify, and associate RID values with different record layouts.

The following example shows a COBOL copybook with a single 01 level, REDEFINE statements, and an RID field:

```
00001 * TRAIN6 EXAMPLE COBOL COPYBOOK
00002   01 MASTER_REC.                                              COL 73-80
00003      05  ACCOUNT_NO                 PIC X(9).                 COL 73-80
00004      05  REC_TYPE                   PIC X.                    COL 73-80
00005      05  AMOUNT                     PIC S9(4)V99 COMP-3.      COL 73-80
00006      05  BIN-NO                     PIC S9(8) COMP.           COL 73-80
00007      05  BIN-NO-X REDEFINES BIN-NO  PIC XXXX.                 COL 73-80
00008      05  DECIMAL-NO                 PIC S999.                 COL 73-80
00009      05  MASTER-DATE.                                         COL 73-80
00010          10  DATE-YY            PIC 9(2).                     COL 73-80
00011          10  DATE-MM            PIC 9(2).                     COL 73-80
00012          10  DATE-DD            PIC 9(2).                     COL 73-80
00013      05  MASTER-DOB REDEFINES MASTER-DATE.                    COL 73-80
00014          10  YYMMDD             PIC XXXXXX.                   COL 73-80
00015      05  ACT_TYPE                   PIC X.                    COL 73-80
00016      05  OTHER-DATE.                                          COL 73-80
00017          10  ODATE-YY           PIC 9(2).                     COL 73-80
00018          10  ODATE-MM           PIC 9(2).                     COL 73-80
00019          10  ODATE-DD           PIC 9(2).                     COL 73-80
00020      05  OTHER-DOB REDEFINES OTHER-DATE.                      COL 73-80
00021          10  OYYMMDDTT          PIC 9(8).                     COL 73-80
00022      05  OTHER_TYPE                 PIC X.                    COL 73-80
```

The createdatamaps utility can identify the RID field, REC_TYPE. Also, for each record layout that matches all of the data records that have a given RID value, the utility can assign an RID value to the record in the data map.

The following example shows a COBOL copybook with multiple 01 levels, where each 01 level defines a record type and has an RID field:

```
* train3.cob, fixed length records 60 bytes long
 01  NAME_REC.
     04  ACCOUNT               PIC 9(3).
     04  RECTYPE               PIC 9(2).
     04  NAME                  PIC X(20).
     04  SEX                   PIC X.
     04  ITEMCT                PIC 9.
     04  ITEMS OCCURS 3 DEPENDING ON ITEMCT PIC X(10).
     04  FILLER                PIC XXX.
01  ACCOUNT_REC.
     04  ACCOUNT               PIC 9(3).
     04  RECTYPE               PIC 9(2).
     04  AMOUNT                PIC 9(9)V99.
     04  POLICY_DATE           PIC X(8).
     04  FILLER                PIC X(36).
```

The createdatamaps utility can identify the RID field, RECTYPE, for each record type and associate the record type with an RID value.

Requirements and Limitations for Finding RID Fields

When you use the createdatamaps utility to find RID fields, the following requirements and limitations apply:

- The utility determines the field or fields that are most likely to be RID fields and associates the RID values with the record layouts that are the best match. Be sure to open the data map in the PowerExchange Navigator and confirm or edit the results.
- The utility can find RID fields only for sequential or VSAM data sources on z/OS.
- Both the data and the metadata must reside on the z/OS machine. When you start the utility from the command line, include the -pwxLocation parameter to specify the location of the PowerExchange Listener.
- The same user ID and password are required to access both data and metadata. When you start the utility from the command line, include the -pwxUserName and -pwxPassword parameters.
- A copybook can define variable-length arrays and groups. For example, a copybook might include the following line:

  ```
  05  ARRAY OCCURS 3 DEPENDING ON ITEMCT PIC X(5).
  ```

  The following limitations apply to variable-length arrays and groups:

  - Nested variable-length arrays or groups are not supported.
  - If the metadata is variable length, the createdatamaps utility assumes that the record format is variable (RECFM=V).
  - An RID field cannot follow a variable-length array or group.

Control File Elements for Finding Record IDs

You can include the following elements in the control file to configure finding RID fields:

cacheConfig

Controls the data cache on disk. You can define this element at a global level but not at a data map instance level.

The CacheConfig element includes the following elements:

cachePath

The full path to the folder for temporary working files. The cache path is written to the message log.

flushDataMode

Specifies when to flush the cache of the data records that were downloaded from the z/OS system.

Valid values:

- e - Flush the cache when the createdatamaps utility finishes.
- d - Flush the cache after each data map is created.

The default value of "e" allows data to be shared by multiple data map generations during one createdatamaps session.

excludeUnmatchedRecords

If true, generates a data map record only for those layouts for which the utility finds a valid data record ID.

findRecordIds

Enables or disables finding RIDs. You can specify findRecordIds at the global or data map instance level.

maxRedefines

Maximum number of record layouts to generate from redefinitions. You can specify maxRedefines at the global or data map instance level.

ridConfig

Defines parameters for finding RID fields. It includes following elements:

readRecordLimit

Maximum number of data records to read from each data file.

recordTypeLimit

Maximum number of record types in the data file.

fieldWidth

Maximum width in bytes of an RID field.

fieldOffset

Byte offset of the RID field from the start of the record, beginning at offset 0. If not specified, the utility finds the RID field.

matchFieldWidth

If true, the RID field must exactly match the fieldWidth value. If false, fieldWidth represents the maximum field width.

Related Topics:

- "Schema File Reference" on page 28
- "Control File Structure" on page 24
- "How the createdatamaps Utility Determines RID Fields" on page 48

How the createdatamaps Utility Determines RID Fields

When you configure the control file to find RID fields, the utility performs the following steps:

1. Identifies candidate RID fields in the metadata model. Alternatively, if you define the fieldOffset element, validates the value that you provide.
2. Obtains the candidate RID values from sample data records.
3. Matches the possible metadata layouts against sample data records, and sets the RID field and values when a good match is achieved.

If the copybook includes multiple 01 levels, the utility performs these steps for each 01 level.

Step 1 - Identifying Candidate RID Fields

The createdatamaps utility examines the COBOL copybook metadata to find candidate RID fields. The utility then determines the offset and length of the candidate RID fields.

Alternatively, if you define the fieldOffset element, the utility validates the value that you provide.

The utility uses the following principles to find candidate RID fields:

- The utility must identify at least one candidate RID field from the metadata records. If the utility finds no candidate RID fields, it stops processing the copybook for RID fields.
- RID fields must be at the same offset and have the same width for all record layouts based on a copybook.

The following element in the control file configures Step 1 processing:

- ridConfig.fieldWidth. Maximum number of bytes of an RID field.

Step 2 - Reading RID Values from Sample Data

The createdatamaps utility reads data records from the data file that is specified in the seqFileName or vsamFileName element in the control file. The utility uses the field lengths and offsets of each candidate RID field from Step 1 to read the values of the candidate RID fields in the data file. Alternatively, if you define the fieldOffset element, the utility uses the fieldOffset value to find RID values in the data file.

The utility rejects candidate RID fields based on certain checks. For example, the number of data values might exceed the maximum number of record types.

The utility uses the following principles:

- The output of Step 2 must be one RID field selected from the candidates. If more than one candidate field conforms to all the checks, the utility selects the first candidate field.
- If no candidate RID fields remain after Step 2, the utility stops processing the copybook for RID fields.
- Candidate fields are rejected if any of the following conditions apply:
  - The list of discovered RID values for the field exceeds the limit.
  - Two data records of different lengths have the same RID value.

The following elements in the control file are used in Step 2 processing:

- ridConfig.recordTypeLimit. Maximum number of distinct values in a valid RID field.
- ridConfig.readRecordLimit. Maximum number of data records to read from each data source.
- seqFileName or vsamFileName. Full path and file name of the sequential or VSAM data set that the utility reads.

Step 3 - Matching Record Layouts Against Data Records

The utility internally generates all possible record layouts, up to the number of layouts that the maxRedefines element specifies. The utility matches each possible layout against data records.

For each record layout, the utility checks the following conditions:

- The record length matches the length of at least one data record.
- Each field in the layout can possibly describe the data.

For each record layout that meets both conditions, the utility creates a record and a table for the layout and assigns an RID value to the record.

For each record layout that does not meet both conditions, the utility performs one of the following actions, depending on how you define the excludeUnmatchedRecords element:

- If excludeUnmatchedRecords = true, the utility excludes the layout from the data map.
- If excludeUnmatchedRecords = false or is not defined, the utility creates a record and a table for the layout but does not assign an RID value to the record.

The utility uses the following principles:

- This step is successful if at least one matching record layout exists for each data record type, that is, for each known RID value. This result is not guaranteed if the number of generated record layouts, limited by maxRedefines, is less than the number of data record types.
- Multiple record layouts might match a data record. All of these matching layouts are included in the data map. You can open the data map in the PowerExchange Navigator to view and select the correct record.

The following elements in the control file configures Step 3 processing:

- CopybookParserConfig.maxRedefines. The maximum number of redefines is the upper limit on the number of generated record layouts, and therefore the maximum number of record layouts to match against data records.
- excludeUnmatchedRecords. If true, generates a data map record only for those layouts for which the utility finds a valid data record ID.

Cache Operation

If you configure the control file to find RID fields, the utility connects to the PowerExchange Listener on the z/OS system and reads data records. The utility saves downloaded data records to a temporary disk cache to process them. The utility deletes the cache files when it finishes execution or after it generates each data map, depending on the value of the flushDataMode element that you specify in the control file.

The default read record limit is 10,000 records. The utility stores up to this number of records in memory. Any records above this limit are spilled to disk. for example, if the control file specifies readRecordLimit=15,000, 10,000 records are cached in memory and 5,000 records are spilled to disk.

The maximum record length that PowerExchange supports is 144 KB. This record length correlates to approximately 1.37GB of RAM (10000*144*1024 bytes). Ensure that the infacmd JVM is configured to run with a suitably large heap size. A setting of -Xmx1500m is usually sufficient.

To configure cache operation, define the cacheConfig element in the control file. The cacheConfig element includes the following elements:

- cachePath
- flushDataMode

For more information about cacheConfig, see "CacheConfig Complex Type" on page 35.

Redefine Filler Fields

Because a REDEFINE statement defines the same section of memory in multiple ways, redefine fields and groups must be the same length as each other. If the copybook does not define these fields to be the same length, the createdatamaps utility inserts FILLER fields in the appropriate locations.

An exception to this rule is a REDEFINE statement that is the last item in a copybook. In this case, the redefine fields or groups can be different lengths because they typically describe different data record types, which can be different lengths.

Copybook and DBD Metadata for IMS Data Maps

For each IMS data map instance, you must specify a DBD. You can optionally specify a COBOL copybook overlay for each segment that the DBD defines. You can specify a different copybook for each segment or the same copybook for multiple segments.

Within an imsDatamapInstance element, specify the following elements:

- To define a DBD only, include an importDBDDetails element within the imsDatamapInstance element.

  For an example, see "Example: IMS DBD Import with No COBOL Overlay" on page 60

- To define a DBD with copybook overlays, include an importDBDDetails element within the imsDatamapInstance element, and include an OverlayDetails element for each segment.

  For an example, see "Example: IMS DBD Import with COBOL Overlays" on page 62.

When you import a DBD with multiple segments, by default, the createdatamaps utility creates a record and a table for each segment and also creates a complex table that includes the columns from each segment in the hierarchy. To disable the creation of complex tables, specify false for the createTablesForHierPath element.

The MaxRedefines element is not supported for IMS data maps. For IMS data maps, the utility always selects the first redefine.

Unavailable Data Map Properties

The createdatamaps utility does not provide the ability to define certain data map properties. To change the defaults for these properties, you must edit the data map in the PowerExchange Navigator.

For SEQ data maps, you cannot define the following properties in the control file:

- Fixed
- Variable
- Default
- Size
- Field Separator
- Merge Adjacent Separators
- Field Delimiter
- Encoding
- Codepage
- File List Processing

For VSAM data maps, you cannot define the following properties in the control file:

- CI ACCESS
- Data Codepage
- Number of Data Buffers
- Number of Index Buffers
- Prefix record with RRN value
- Prefix record with XRBA value
- File List Processing

For IMS data maps, you cannot define the following properties in the control file:

- Data Codepage

The defaults for these properties are described in the *PowerExchange Navigator User Guide*.

Examples

The following examples show how to create data maps for sequential, VSAM, and IMS data sources.

Control files for most of the examples are installed in the following directories on the machines where the Informatica Client and the Informatica services are installed:

- *Informatica_client_installation_directory*\clients\DeveloperClient\osgi_mf_plugins\jars\resources\examples
- *Informatica_services_installation_directory*\pwxmfplugins\resources\examples

The following control files are provided:

- ims_advanced.xml
- ims_simple.xml
- seq_advanced.xml
- seq_simple.xml
- vsam_advanced.xml
- vsam_simple.xml

To use the example control files without editing them, you must run the createdatamaps command from the directory on the Informatica Client or Informatica services machine in which they are installed. If you run the command from a different location, you must replace all the metadata file names in the control files with fully qualified file names before you run the command.

The following examples run the createdatamaps command from the Informatica services machine.

Example: Simple SEQ Data Map

This example describes a control file that creates a data map for a sequential data source and imports simple copybook metadata.

The control file defines the following properties:

- Global properties: schema name
- Data map instance properties: copybook location

Default values for the following elements are defined in the schema file:

- seqFileName = file.dat
- datamapName = MAP
- maxRedefines = 1

The copybook includes REDEFINES statements that define six possible layouts. Because maxRedefines = 1, a record and table are created for the first combination only: BIN-NO, MASTER-DATE. For more information, see the following topics:

- "Example: SEQ Data Map with Multiple Records and Tables" on page 54
- "Redefinitions Without Record IDs in COBOL Copybooks" on page 45

Command Line

To run this example on the Informatica services machine, enter the following command at the command line:

```
Informatica_services_installation_directory\isp\bin\infacmd pwx createDatamaps
-datamapOutputDir Output -controlFile seq_simple.xml -logFile Output\seq_simple.log
-verbosity INFO
```

Control File

The control file for this example, seq_simple.xml, contains the following lines:

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
NOTE: Metadata file paths in this sample control file are relative to current directory.
If 'infacmd' command is issued from a different directory, all relative file paths must
be replaced with absolute file paths.
-->

<!-- xmlSchemaVersion set to 1.0 -->
<DatamapGeneration xmlSchemaVersion="1.0" xmlns="http://
com.informatica.cmd.pwx.createdatamaps/DatamapGeneration">
    <seqGen>

        <!-- Global settings for datamap file name and contents -->
        <globalGenConfig>
            <schemaName>SEQSIMPLE</schemaName>
        </globalGenConfig>

        <datamapInstances>
            <!-- Import from a copybook with default properties -->
            <seqDatamapInstance>
                <importCopybookDetails>
                    <filePath>
                        <windowsPath>metadata\train61.cob</windowsPath>
                    </filePath>
                </importCopybookDetails>
            </seqDatamapInstance>
        </datamapInstances>
    </seqGen>
</DatamapGeneration>
```

COBOL Copybook File

The COBOL copybook for the data map in this example, tran61.cob, contains the following lines:

```
00001 * TRAIN6 EXAMPLE COBOL COPYBOOK
00002   01 MASTER_REC.                                              COL 73-80
00003      05  ACCOUNT_NO                 PIC X(9).                COL 73-80
00004      05  REC_TYPE                   PIC X.                   COL 73-80
00004      05  AMOUNT                     PIC S9(4)V99 COMP-3.     COL 73-80
00005      05  BIN-NO                     PIC S9(8) COMP.          COL 73-80
00006      05  BIN-NO-X REDEFINES BIN-NO  PIC XXXX.                COL 73-80
00006      05  BIN-NO-9 REDEFINES BIN-NO  PIC 9(4).                COL 73-80
00007      05  DECIMAL-NO                 PIC S999.                COL 73-80
00008      05  MASTER-DATE.                                        COL 73-80
00009          10  DATE-YY           PIC 9(2).                     COL 73-80
00010          10  DATE-MM           PIC 9(2).                     COL 73-80
00011          10  DATE-DD           PIC 9(2).                     COL 73-80
00012      05  OTHER-DATE REDEFINES MASTER-DATE.                   COL 73-80
00013          10  OTHER-YY           PIC 9(2).                    COL 73-80
00014          10  OTHER-MM           PIC 9(2).                    COL 73-80
00015          10  OTHER-DD           PIC 9(2).                    COL 73-80
```

Log File

The log file this example, seq_simple.log, contains the following lines:

```
2013-12-05 15:29:41 INFO [MDO_34613] Configuration for this run: location=, user name=,
datamap directory=Output, control file=seq_simple.xml
2013-12-05 15:29:49 INFO [MDAdapter_34100] Finding metadata. Path filter = file.dat
2013-12-05 15:29:49 INFO [MDAdapter_34100] Finding metadata. Path filter = metadata
\train61.cob
2013-12-05 15:29:49 INFO [MDAdapter_34101] Fetching file metadata\train61.cob
2013-12-05 15:29:50 INFO [MDO_34612] Copybook 'MASTER_REC' has 6 possible layouts
(Maximum configured limit is 1).
2013-12-05 15:29:50 INFO [JDMX2_34801] 1 records imported.
```

```
2013-12-05 15:29:50 INFO [JDMX2_34802] 9 fields imported.
2013-12-05 15:29:50 INFO [JDMX2_34803] 1 tables imported.
2013-12-05 15:29:50 INFO [MDO_34619] Datamap file 'Output\SEQSIMPLE.MAP.dmp' was written.
2013-12-05 15:29:55 INFO [MDO_34614] Run complete: 1 datamap(s) created. 0 error and 0
warning messages.
```

Data Map Files

This example creates a data map with the following file name and relative path:

- Output\SEQSIMPLE.MAP.dmp

Example: SEQ Data Map with Multiple Records and Tables

This example describes a control file that creates two SEQ data maps. The imported copybook metadata includes REDEFINES statements, and the data maps that are created contain multiple records and tables.

The example contains global elements and two DatamapInstance elements. At the global level, maxRedefines is set to 2. The second data map instance overrides this setting with a value of 6.

Both data maps import metadata from train61.dat, which includes the following redefinitions:

- BIN-NO-X and BIN-NO-9 both redefine BIN-NO.
- OTHER-DATE redefines MASTER-DATE.

These redefintions result in six combinations of fields. Because the global MaxRedefines setting of 2 is in effect for the first data map, records are created for only the first two combinations:

- BIN-NO, MASTER-DATE
- BIN-NO, OTHER-DATE

Because the MaxRedefines setting of 6 is in effect for the second data map, records are created for all six combinations:

- BIN-NO, MASTER-DATE
- BIN-NO, OTHER-DATE
- BIN-NO-X, MASTER-DATE
- BIN-NO-X, OTHER-DATE
- BIN-NO-9, MASTER-DATE
- BIN-NO-9, OTHER-DATE

For more information, see "Redefinitions Without Record IDs in COBOL Copybooks" on page 45.

Command Line

To run this example on the Informatica services machine, enter the following command at the command line:

```
Informatica_services_installation_directory\isp\bin\infacmd pwx createDatamaps
-datamapOutputDir Output -controlFile seq_advanced.xml -logFile Output\seq_advanced.log
-verbosity INFO
```

Control File

The control file for this example, seq_advanced.xml, contains the following lines:

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
NOTE: Metadata file paths in this sample control file are relative to current directory.
If 'infacmd' command is issued from a different directory, all relative file paths must
be replaced with absolute file paths.
-->

<!-- xmlSchemaVersion set to 1.0 -->
```

```
<DatamapGeneration xmlSchemaVersion="1.0" xmlns="http://
com.informatica.cmd.pwx.createdatamaps/DatamapGeneration">
    <seqGen>

        <!-- Global settings for copybooks -->
        <globalCopybookParserConfig>
            <startColumn>7</startColumn>
            <endColumn>72</endColumn>
            <maxRedefines>2</maxRedefines>
        </globalCopybookParserConfig>

        <!-- Global settings for datamap file name and contents -->
        <globalGenConfig>
            <schemaName>SEQADV</schemaName>
            <datamapName>TRAIN6</datamapName>
        </globalGenConfig>

        <datamapInstances>

            <!-- Datamap 1: maxRedefines is 2 from global settings, resulting in 2
datamap records -->
            <seqDatamapInstance>
                <importCopybookDetails>
                    <filePath>
                        <windowsPath>metadata\train61.cob</windowsPath>
                    </filePath>
                </importCopybookDetails>
                <datamapProperties>
                    <seqFileName>
                        <zosPath>COM.INFA.SEQ1</zosPath>
                    </seqFileName>
                </datamapProperties>
            </seqDatamapInstance>

            <!-- Datamap 2: maxRedefines value overridden to 6, resulting in 6 datamap
records -->
            <seqDatamapInstance>
                <genConfig>
                    <datamapName>TRN6REDEF</datamapName>
                </genConfig>
                <importCopybookDetails>
                    <filePath>
                        <windowsPath>metadata\train61.cob</windowsPath>
                    </filePath>
                    <parserConfig>
                        <maxRedefines>6</maxRedefines>
                    </parserConfig>
                </importCopybookDetails>
                <datamapProperties>
                    <seqFileName>
                        <zosPath>COM.INFA.SEQ2</zosPath>
                    </seqFileName>
                </datamapProperties>
            </seqDatamapInstance>
        </datamapInstances>

    </seqGen>
</DatamapGeneration>
```

COBOL Copybook File

The COBOL copybook for the data map in this example, tran61.cob, contains the following lines:

```
00001 * TRAIN6 EXAMPLE COBOL COPYBOOK
00002   01 MASTER_REC.                                          COL 73-80
00003      05  ACCOUNT_NO                 PIC X(9).             COL 73-80
00004      05  REC_TYPE                   PIC X.                COL 73-80
00004      05  AMOUNT                     PIC S9(4)V99 COMP-3.  COL 73-80
00005      05  BIN-NO                     PIC S9(8) COMP.       COL 73-80
00006      05  BIN-NO-X REDEFINES BIN-NO  PIC XXXX.             COL 73-80
```

```
00006      05  BIN-NO-9 REDEFINES BIN-NO   PIC 9(4).                  COL 73-80
00007      05  DECIMAL-NO                  PIC S999.                  COL 73-80
00008      05  MASTER-DATE.                                           COL 73-80
00009          10  DATE-YY            PIC 9(2).                      COL 73-80
00010          10  DATE-MM            PIC 9(2).                      COL 73-80
00011          10  DATE-DD            PIC 9(2).                      COL 73-80
00012      05  OTHER-DATE REDEFINES MASTER-DATE.                      COL 73-80
00013          10  OTHER-YY            PIC 9(2).                      COL 73-80
00014          10  OTHER-MM            PIC 9(2).                      COL 73-80
00015          10  OTHER-DD            PIC 9(2).                      COL 73-80
```

Log File

The log file for this example, seq_advanced.log, contains the following lines:

```
2013-12-05 15:29:57 INFO [MDO_34613] Configuration for this run: location=, user name=,
datamap directory=Output, control file=seq_advanced.xml
2013-12-05 15:30:05 INFO [MDAdapter_34100] Finding metadata. Path filter = COM.INFA.SEQ1
2013-12-05 15:30:05 INFO [MDAdapter_34100] Finding metadata. Path filter = metadata
\train61.cob
2013-12-05 15:30:05 INFO [MDAdapter_34101] Fetching file metadata\train61.cob
2013-12-05 15:30:05 INFO [MDO_34612] Copybook 'MASTER_REC' has 6 possible layouts
(Maximum configured limit is 2).
2013-12-05 15:30:06 INFO [JDMX2_34801] 2 records imported.
2013-12-05 15:30:06 INFO [JDMX2_34802] 18 fields imported.
2013-12-05 15:30:06 INFO [JDMX2_34803] 2 tables imported.
2013-12-05 15:30:06 INFO [MDO_34619] Datamap file 'Output\SEQADV.TRAIN6.dmp' was written.
2013-12-05 15:30:06 INFO [MDAdapter_34100] Finding metadata. Path filter = COM.INFA.SEQ2
2013-12-05 15:30:06 INFO [MDAdapter_34100] Finding metadata. Path filter = metadata
\train61.cob
2013-12-05 15:30:06 INFO [MDAdapter_34101] Fetching file metadata\train61.cob
2013-12-05 15:30:06 INFO [MDO_34612] Copybook 'MASTER_REC' has 6 possible layouts
(Maximum configured limit is 6).
2013-12-05 15:30:06 INFO [JDMX2_34801] 6 records imported.
2013-12-05 15:30:06 INFO [JDMX2_34802] 54 fields imported.
2013-12-05 15:30:06 INFO [JDMX2_34803] 6 tables imported.
2013-12-05 15:30:06 INFO [MDO_34619] Datamap file 'Output\SEQADV.TRN6REDEF.dmp' was
written.
2013-12-05 15:30:12 INFO [MDO_34614] Run complete: 2 datamap(s) created. 0 error and 0
warning messages.
```

Data Map Files

This example creates data maps with the following file names and relative paths:

- Output\SEQADV.TRAIN6.MAP.dmp
- Output\SEQADV.TRN6REDEF.dmp

The schema name for each data map is taken from the global settings. The data map name for the first and second data maps is taken from the global setting and the data map instance setting, respectively.

Example: Simple VSAM KSDS Data Map

This example describes a control file that creates a KSDS data map and imports simple copybook metadata.

The control file is similar to the one for creating a simple SEQ data map. The control file defines the following properties:

- Global properties: schema name
- Data map instance properties: copybook location

Default values for all other properties are defined in the schema file. Because KSDS is the default for globalMapType, this example creates a KSDS data map.

Command Line

To run this example on the Informatica services machine, enter the following command at the command line:

```
Informatica_services_installation_directory\isp\bin\infacmd pwx createDatamaps
-datamapOutputDir Output -controlFile vsam_simple.xml -logFile Output\vsam_simple.log
-verbosity INFO
```

Control File

The control file for this example, vsam_simple.xml, contains the following lines:

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
NOTE: Metadata file paths in this sample control file are relative to current directory.
If 'infacmd' command is issued from a different directory, all relative file paths must
be replaced with absolute file paths.
-->

<!-- xmlSchemaVersion set to 1.0 -->
<DatamapGeneration xmlSchemaVersion="1.0" xmlns="http://
com.informatica.cmd.pwx.createdatamaps/DatamapGeneration">
    <vsamGen>

        <!-- Global settings for datamap file name and contents -->
        <globalGenConfig>
            <schemaName>VSAMSIMPLE</schemaName>
        </globalGenConfig>

        <datamapInstances>

            <!-- Import from a copybook with default properties -->
            <vsamDatamapInstance>
                <importCopybookDetails>
                    <filePath>
                        <windowsPath>metadata\train61.cob</windowsPath>
                    </filePath>
                </importCopybookDetails>
            </vsamDatamapInstance>
        </datamapInstances>

    </vsamGen>
</DatamapGeneration>
```

COBOL Copybook

This example imports metadata from the train61.cob copybook. For the contents of this copybook, see "Example: Simple SEQ Data Map" on page 52.

Log File

The log file for this example, vsam_simple.log, contains the following lines:

```
2013-12-05 15:30:14 INFO [MDO_34613] Configuration for this run: location=, user name=,
datamap directory=Output, control file=vsam_simple.xml
2013-12-05 15:30:21 INFO [MDAdapter_34100] Finding metadata. Path filter = file.dat
2013-12-05 15:30:21 INFO [MDAdapter_34100] Finding metadata. Path filter = metadata
\train61.cob
2013-12-05 15:30:21 INFO [MDAdapter_34101] Fetching file metadata\train61.cob
2013-12-05 15:30:22 INFO [MDO_34612] Copybook 'MASTER_REC' has 6 possible layouts
(Maximum configured limit is 1).
2013-12-05 15:30:22 INFO [JDMX2_34801] 1 records imported.
2013-12-05 15:30:22 INFO [JDMX2_34802] 9 fields imported.
2013-12-05 15:30:22 INFO [JDMX2_34803] 1 tables imported.
2013-12-05 15:30:22 INFO [MDO_34619] Datamap file 'Output\VSAMSIMPLE.MAP.dmp' was
written.
2013-12-05 15:30:28 INFO [MDO_34614] Run complete: 1 datamap(s) created. 0 error and 0
warning messages.
```

Data Map File

This example creates a data map with the following file name and relative path:

- Output\VSAMSIMPLE.MAP.dmp

Example: VSAM RRDS Data Maps with Multiple Records and Tables

This example describes a control file that creates two VSAM RRDS data maps. The imported copybook metadata includes REDEFINES statements, and the data maps that are created contain multiple records and tables.

This example is similar to the one for creating SEQ data maps with multiple records and tables. For more information about how the REDEFINES statements in the COBOL copybook result in multiple records and tables, see "Example: SEQ Data Map with Multiple Records and Tables" on page 54.

Because the control file sets the value of GlobalMapType to RRDS, the example creates RRDS data maps.

Command Line

To run this example on the Informatica services machine, enter the following command at the command line:

```
Informatica_services_installation_directory\isp\bin\infacmd pwx createDatamaps -
datamapOutputDir Output -controlFile vsam_advanced.xml -logFile Output\vsam_advanced.log
-verbosity INFO
```

Control File

The control file for this example, vsam_advanced.xml, contains the following lines:

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
NOTE: Metadata file paths in this sample control file are relative to current directory.
If 'infacmd' command is issued from a different directory, all relative file paths must
be replaced with absolute file paths.
-->

<!-- xmlSchemaVersion set to 1.0 -->
<DatamapGeneration xmlSchemaVersion="1.0" xmlns="http://
com.informatica.cmd.pwx.createdatamaps/DatamapGeneration">
    <vsamGen>

        <!-- Global settings for copybooks -->
        <globalCopybookParserConfig>
            <startColumn>7</startColumn>
            <endColumn>72</endColumn>
            <maxRedefines>2</maxRedefines>
        </globalCopybookParserConfig>

        <!-- Global settings for datamap file name and contents -->
        <globalGenConfig>
            <schemaName>VSAMADV</schemaName>
            <datamapName>TRAIN6</datamapName>
        </globalGenConfig>

        <!-- Global access method for VSAM datamaps -->
        <globalMapType>RRDS</globalMapType>

        <datamapInstances>

            <!-- Datamap 1: maxRedefines is 2 from global settings, resulting in 2
datamap records -->
            <vsamDatamapInstance>
                <importCopybookDetails>
                    <filePath>
                        <windowsPath>metadata\train61.cob</windowsPath>
```

```
                    </filePath>
                </importCopybookDetails>
                <datamapProperties>
                    <vsamFileName>
                        <zosPath>COM.INFA.RRDS1</zosPath>
                    </vsamFileName>
                </datamapProperties>
            </vsamDatamapInstance>

            <!-- Datamap 2: maxRedefines value overridden to 6, resulting in 6 datamap
records -->
            <vsamDatamapInstance>
                <genConfig>
                    <datamapName>TRN6REDEF</datamapName>
                </genConfig>
                <importCopybookDetails>
                    <filePath>
                        <windowsPath>metadata\train61.cob</windowsPath>
                    </filePath>
                    <parserConfig>
                        <maxRedefines>6</maxRedefines>
                    </parserConfig>
                </importCopybookDetails>
                <datamapProperties>
                    <vsamFileName>
                        <zosPath>COM.INFA.RRDS2</zosPath>
                    </vsamFileName>
                </datamapProperties>
            </vsamDatamapInstance>
        </datamapInstances>

    </vsamGen>
</DatamapGeneration>
```

COBOL Copybook

This example imports metadata from the train61.cob copybook. For the contents of this copybook, see "Example: Simple SEQ Data Map" on page 52.

Log File

The log file for this example, vsam_advanced.log, contains the following lines:

```
2013-12-05 15:30:30 INFO [MDO_34613] Configuration for this run: location=, user name=,
datamap directory=Output, control file=vsam_advanced.xml
2013-12-05 15:30:38 INFO [MDAdapter_34100] Finding metadata. Path filter = COM.INFA.RRDS1
2013-12-05 15:30:38 INFO [MDAdapter_34100] Finding metadata. Path filter = metadata
\train61.cob
2013-12-05 15:30:38 INFO [MDAdapter_34101] Fetching file metadata\train61.cob
2013-12-05 15:30:38 INFO [MDO_34612] Copybook 'MASTER_REC' has 6 possible layouts
(Maximum configured limit is 2).
2013-12-05 15:30:39 INFO [JDMX2_34801] 2 records imported.
2013-12-05 15:30:39 INFO [JDMX2_34802] 18 fields imported.
2013-12-05 15:30:39 INFO [JDMX2_34803] 2 tables imported.
2013-12-05 15:30:39 INFO [MDO_34619] Datamap file 'Output\VSAMADV.TRAIN6.dmp' was
written.
2013-12-05 15:30:39 INFO [MDAdapter_34100] Finding metadata. Path filter = COM.INFA.RRDS2
2013-12-05 15:30:39 INFO [MDAdapter_34100] Finding metadata. Path filter = metadata
\train61.cob
2013-12-05 15:30:39 INFO [MDAdapter_34101] Fetching file metadata\train61.cob
2013-12-05 15:30:39 INFO [MDO_34612] Copybook 'MASTER_REC' has 6 possible layouts
(Maximum configured limit is 6).
2013-12-05 15:30:39 INFO [JDMX2_34801] 6 records imported.
2013-12-05 15:30:39 INFO [JDMX2_34802] 54 fields imported.
2013-12-05 15:30:39 INFO [JDMX2_34803] 6 tables imported.
2013-12-05 15:30:39 INFO [MDO_34619] Datamap file 'Output\VSAMADV.TRN6REDEF.dmp' was
written.
2013-12-05 15:30:44 INFO [MDO_34614] Run complete: 2 datamap(s) created. 0 error and 0
warning messages.
```

Data Map Files

This example creates data maps with the following file names and relative paths:

- Output\VSAMADV.TRAIN6.dmp
- Output\VSAMADV.TRN6REDEF.dmp

Example: IMS DBD Import with No COBOL Overlay

This example describes a control file that creates an IMS data map and imports DBD metadata. This example does not import COBOL copybook metadata to overlay each segment.

The createdatamaps utility creates a record and a table for each of the two segments defined in the DBD. The utility also creates a complex table that includes columns from the STUDENT parent record and the CORSECN child record.

Although the DBD redefines the CRSEKEY field in the CORSECN record, the utility generates a table and record for the first redefine only. For IMS data maps, maxRedefines always equals 1. The log file reports which redefine was used and which redefines were skipped.

Command Line

To run this example on the Informatica services machine, enter the following command at the command line:

```
Informatica_services_installation_directory\isp\bin\infacmd pwx createDatamaps
-datamapOutputDir Output -controlFile ims_simple.xml -logFile Output\ims_simple.log
-verbosity INFO
```

Control File

The control file for this example, ims_simple.xml, contains the following lines:

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
NOTE: Metadata file paths in this sample control file are relative to current directory.
If 'infacmd' command is issued from a different directory, all relative file paths must
be replaced with absolute file paths.
-->

<!-- xmlSchemaVersion set to 1.0 -->
<DatamapGeneration xmlSchemaVersion="1.0" xmlns="http://
com.informatica.cmd.pwx.createdatamaps/DatamapGeneration">
    <imsGen>

        <!-- Global settings for datamap file name and contents -->
        <globalGenConfig>
            <schemaName>IMSSIMPLE</schemaName>
        </globalGenConfig>

        <datamapInstances>
            <!-- Import from a DBD with default properties -->
            <imsDatamapInstance>
                <importDBDDetails>
                    <filePath>
                        <windowsPath>metadata\train8.dbd</windowsPath>
                    </filePath>
                </importDBDDetails>
            </imsDatamapInstance>
        </datamapInstances>

    </imsGen>
</DatamapGeneration>
```

DBD File

The DBD file used in this example, train8.dbd, contains the following lines:

```
 DBD      NAME=DTLSTDNT,ACCESS=(HIDAM,VSAM)
  DATASET DD1=DTLSTDNT
   SEGM                                                              \
    NAME=STUDENT,PARENT=0,FREQ=10000,BYTES=210,PTR=TB
    LCHILD NAME=(STUDIDX,DTLSTDIX),PTR=INDX
    FIELD TYPE=C,START=162,BYTES=12,NAME=(ID,SEQ,U)
    FIELD TYPE=C,START=01,BYTES=40,NAME=PNAME
    FIELD TYPE=C,START=41,BYTES=40,NAME=ADDRESS1
    FIELD TYPE=C,START=81,BYTES=40,NAME=ADDRESS2
    FIELD TYPE=C,START=121,BYTES=30,NAME=CITY
    FIELD TYPE=C,START=151,BYTES=2,NAME=STATE
    FIELD TYPE=C,START=153,BYTES=9,NAME=ZIP
    FIELD TYPE=C,START=174,BYTES=6,NAME=BDATE
    FIELD TYPE=C,START=180,BYTES=1,NAME=SEX
    FIELD TYPE=C,START=181,BYTES=2,NAME=HEIGHT
    FIELD TYPE=C,START=183,BYTES=3,NAME=WEIGHT
    FIELD TYPE=C,START=186,BYTES=5,NAME=HAIR
    FIELD TYPE=C,START=191,BYTES=5,NAME=EYES
    FIELD TYPE=C,START=196,BYTES=4,NAME=ENRLMMYY
    FIELD TYPE=C,START=200,BYTES=4,NAME=GRADMMYY
*
   SEGM NAME=CORSECTN,PARENT=((STUDENT,SNGL)),FREQ=05,BYTES=14,PTR=TB
    FIELD TYPE=C,START=01,BYTES=14,NAME=(CRSEKEY,SEQ,U)
    FIELD TYPE=C,START=01,BYTES=08,NAME=CRSCOURS
    FIELD TYPE=C,START=09,BYTES=01,NAME=CRSSECTN
    FIELD TYPE=C,START=10,BYTES=01,NAME=CRSDAY
    FIELD TYPE=C,START=11,BYTES=04,NAME=CRSBEG
  DBDGEN
  FINISH
 END
```

Log File

The log file for this example, ims_simple.log, contains the following lines:

```
2013-12-05 15:30:46 INFO [MDO_34613] Configuration for this run: location=, user name=,
datamap directory=Output, control file=ims_simple.xml
2013-12-05 15:30:54 INFO [MDAdapter_34100] Finding metadata. Path filter = metadata
\train8.dbd
2013-12-05 15:30:54 INFO [MDAdapter_34101] Fetching file metadata\train8.dbd
2013-12-05 15:30:55 INFO [MDAdapter_34108] Definition CRSEKEY selected
2013-12-05 15:30:55 INFO [MDAdapter_34109] Redefinition CRSCOURS skipped
2013-12-05 15:30:55 INFO [MDAdapter_34109] Redefinition CRSSECTN skipped
2013-12-05 15:30:55 INFO [MDAdapter_34109] Redefinition CRSDAY skipped
2013-12-05 15:30:55 INFO [MDAdapter_34109] Redefinition CRSBEG skipped
2013-12-05 15:30:55 INFO [JDMX2_34801] 2 records imported.
2013-12-05 15:30:55 INFO [JDMX2_34802] 16 fields imported.
2013-12-05 15:30:55 INFO [JDMX2_34803] 3 tables imported.
2013-12-05 15:30:55 INFO [MDO_34619] Datamap file 'Output\IMSSIMPLE.MAP.dmp' was written.
2013-12-05 15:31:01 INFO [MDO_34614] Run complete: 1 datamap(s) created. 0 error and 0
warning messages.
```

Data Map File

This example creates a data map with the following file name and relative path:

- Output\IMSSIMPLE.MAP.dmp

Example: IMS DBD Import with COBOL Overlays

This example describes a control file that creates two IMS data maps. The first data map imports DBD metadata only. The second data map imports DBD metadata and the overlaying COBOL copybook metadata for each segment.

The records in the resulting data map derive the fields and CCKs from the COBOL copybooks but retain the search fields from the DBD.

To define the DBDs and COBOL copybooks to import, the second imsDatamapInstance element includes the following elements:

- A importDBDDetails element defines the file name and path of the DBD.
- For each of the two segments defined in the DBD, an overlayDetails element defines the file path of the COBOL copybook and the name of the segment for which the copybook provides overlaying metadata.

The second imsDatamapInstance element also includes a datamapProperties element that defines the data map type, IMS SSID, PSB name, and PCB name.

As in "Example: IMS DBD Import with No COBOL Overlay" on page 60, although the DBD redefines the CRSEKEY field in the CORSECN record, the utility generates a table and record for the first redefine only.

Command Line

To run this example on the Informatica services machine, enter the following command at the command line:

```
Informatica_services_installation_directory\isp\bin\infacmd pwx createDatamaps
-datamapOutputDir Output -controlFile ims_advanced.xml -logFile Output\ims_advanced.log
-verbosity INFO
```

Control File

The control file for this example, ims_advanced.xml, contains the following lines:

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
NOTE: Metadata file paths in this sample control file are relative to current directory.
If 'infacmd' command is issued from a different directory, all relative file paths must
be replaced with absolute file paths.
-->

<!-- xmlSchemaVersion set to 1.0 -->
<DatamapGeneration xmlSchemaVersion="1.0" xmlns="http://
com.informatica.cmd.pwx.createdatamaps/DatamapGeneration">
    <imsGen>

        <!-- Global settings for copybooks -->
        <globalCopybookParserConfig>
            <startColumn>7</startColumn>
            <endColumn>72</endColumn>
        </globalCopybookParserConfig>

        <!-- Global settings for datamap file name and contents -->
        <globalGenConfig>
            <schemaName>IMSADV</schemaName>
            <datamapName>TRAIN8</datamapName>
        </globalGenConfig>

        <datamapInstances>

            <!-- Datamap 1: Import from a DBD without segment overlays -->
            <imsDatamapInstance>

                <importDBDDetails>
                    <filePath>
                        <windowsPath>metadata\train8.dbd</windowsPath>
                    </filePath>
```

```
            </importDBDDetails>

            <datamapProperties>
                <mapType>DL1</mapType>
                <imsSSID>SS1</imsSSID>
                <pcbNumber>1</pcbNumber>
            </datamapProperties>
        </imsDatamapInstance>


        <!-- Datamap 2: Import from a DBD with both segments overlaid -->
        <imsDatamapInstance>
            <genConfig>
                <datamapName>TRAIN8OVR</datamapName>
            </genConfig>

            <importDBDDetails>
                <filePath>
                    <windowsPath>metadata\train8.dbd</windowsPath>
                </filePath>
            </importDBDDetails>

            <!-- Overlay segment 'STUDENT' with a Cobol copybook -->
            <overlayDetails>
                <nativeRecordName>STUDENT</nativeRecordName>
                <overlayCopybookDetails>
                    <filePath>
                        <windowsPath>metadata\student.cob</windowsPath>
                    </filePath>
                </overlayCopybookDetails>
            </overlayDetails>

            <!-- Overlay segment 'CORSECTN' with a Cobol copybook -->
            <overlayDetails>
                <nativeRecordName>CORSECTN</nativeRecordName>
                <overlayCopybookDetails>
                    <filePath>
                        <windowsPath>metadata\course.cob</windowsPath>
                    </filePath>
                </overlayCopybookDetails>
            </overlayDetails>

            <datamapProperties>
                <mapType>ODBA</mapType>
                <imsSSID>SS1</imsSSID>
                <psbName>psb</psbName>
                <pcbName>pcb</pcbName>
            </datamapProperties>
        </imsDatamapInstance>

    </datamapInstances>
</imsGen>
</DatamapGeneration>
```

DBD File

This example uses the train8.dbd file. For the contents of this file, see "Example: IMS DBD Import with No COBOL Overlay" on page 60.

COBOL Copybook Files

The following lines show the contents of the student.cob copybook. This copybook overlays the DBD metadata for the first segment in the second data map.

```
      ****************************************************************
      *
      *   COBOL FD DEFINITION FOR STUDENT FILE
      *
      ****************************************************************
       01  STUDENT-RECORD.
```

```
    04  ST-NAME                     PIC X(040).
    04  ST-ADDRESS-1               PIC X(040).
    04  ST-ADDRESS-2               PIC X(040).
    04  ST-CITY                    PIC X(030).
    04  ST-STATE                   PIC X(002).
    04  ST-ZIP                     PIC X(009).
    04  ST-NUMBER                  PIC 9(012).
    04  ST-BIRTH-DATE.
        08  ST-BIRTH-MM            PIC 9(002).
        08  ST-BIRTH-DD            PIC 9(002).
        08  ST-BIRTH-YY            PIC 9(002).
    04  ST-SEX                     PIC X(001).
    04  ST-HEIGHT                  PIC 9(002).
    04  ST-WEIGHT                  PIC 9(003).
    04  ST-HAIR                    PIC X(005).
    04  ST-EYES                    PIC X(005).
    04  ST-DATE-ENROLL-MM          PIC 9(002).
    04  ST-DATE-ENROLL-YY          PIC 9(002).
    04  ST-DATE-GRAD-MM            PIC 9(002).
    04  ST-DATE-GRAD-YY            PIC 9(002).
    04  ST-TUITION-FEES            PIC S9(8) COMP.
    04  ST-COURSE-COUNT            PIC X(003).
    04  ST-COURSE-DATA OCCURS 10.
        08 ST-COURSE-CODE          PIC 9(005).
        08 ST-COURSE-HOURS         PIC 9(002).
        08 ST-COURSE-TIME          PIC X(005).
        08 ST-COURSE-DAY           PIC X(005).
        08 ST-COURSE-INSTRUCTOR    PIC X(015).
        08 ST-COURSE-BLDG          PIC 9(002).
```

The following lines show the contents of the course.cob copybook. This copybook overlays the DBD metadata for the second segment in the second data map.

```
*****************************************************************
*
*   COBOL FD DEFINITION FOR COURSE
*
*****************************************************************
 01  COURSE.
     04  CRS-COURSE                     PIC X(8).
     04  CRS-SECTN                      PIC 9(1).
     04  CRS-DAY                        PIC 9(1).
     04  CRS-BEG                        PIC X(4).
     04  CRS-END                        PIC X(4).
     04  FILLER                         PIC X(4).
```

Log File

The log file for this example, ims_advanced.log, contains the following lines:

```
2013-12-05 15:31:03 INFO [MDO_34613] Configuration for this run: location=, user name=,
datamap directory=Output, control file=ims_advanced.xml
2013-12-05 15:31:10 INFO [MDAdapter_34100] Finding metadata. Path filter = metadata
\train8.dbd
2013-12-05 15:31:10 INFO [MDAdapter_34101] Fetching file metadata\train8.dbd
2013-12-05 15:31:11 INFO [MDAdapter_34108] Definition CRSEKEY selected
2013-12-05 15:31:11 INFO [MDAdapter_34109] Redefinition CRSCOURS skipped
2013-12-05 15:31:11 INFO [MDAdapter_34109] Redefinition CRSSECTN skipped
2013-12-05 15:31:11 INFO [MDAdapter_34109] Redefinition CRSDAY skipped
2013-12-05 15:31:11 INFO [MDAdapter_34109] Redefinition CRSBEG skipped
2013-12-05 15:31:11 INFO [JDMX2_34801] 2 records imported.
2013-12-05 15:31:11 INFO [JDMX2_34802] 16 fields imported.
2013-12-05 15:31:11 INFO [JDMX2_34803] 3 tables imported.
2013-12-05 15:31:11 INFO [MDO_34619] Datamap file 'Output\IMSADV.TRAIN8.dmp' was written.
2013-12-05 15:31:11 INFO [MDAdapter_34100] Finding metadata. Path filter = metadata
\train8.dbd
2013-12-05 15:31:11 INFO [MDAdapter_34101] Fetching file metadata\train8.dbd
2013-12-05 15:31:11 INFO [MDAdapter_34100] Finding metadata. Path filter = metadata
\student.cob
```

```
2013-12-05 15:31:11 INFO [MDAdapter_34101] Fetching file metadata\student.cob
2013-12-05 15:31:11 INFO [MDAdapter_34100] Finding metadata. Path filter = metadata
\course.cob
2013-12-05 15:31:11 INFO [MDAdapter_34101] Fetching file metadata\course.cob
2013-12-05 15:31:11 INFO [MDAdapter_34108] Definition CRSEKEY selected
2013-12-05 15:31:11 INFO [MDAdapter_34109] Redefinition CRSCOURS skipped
2013-12-05 15:31:11 INFO [MDAdapter_34109] Redefinition CRSSECTN skipped
2013-12-05 15:31:11 INFO [MDAdapter_34109] Redefinition CRSDAY skipped
2013-12-05 15:31:11 INFO [MDAdapter_34109] Redefinition CRSBEG skipped
2013-12-05 15:31:11 INFO [JDMX2_34801] 2 records imported.
2013-12-05 15:31:11 INFO [JDMX2_34802] 35 fields imported.
2013-12-05 15:31:11 INFO [JDMX2_34803] 3 tables imported.
2013-12-05 15:31:11 INFO [MDO_34619] Datamap file 'Output\IMSADV.TRAIN8OVR.dmp' was
written.
2013-12-05 15:31:17 INFO [MDO_34614] Run complete: 2 datamap(s) created. 0 error and 0
warning messages.
```

Data Map Files

This example creates data maps with the following file names and relative paths:

- Output\IMSADV.TRAIN8.dmp
- Output\IMSADV.TRAIN8OVR.dmp

Example: Finding RID Fields

This example describes a control file that is configured to find RID fields.

The control file includes the following global properties that control finding RIDs:

- cacheConfig (cachePath, flushDataMode)

The control file includes the following data map instance properties that control RID processing:

- findRecordIds
- ridConfig (readRecordLimit, recordTypeLimit, fieldWidth)

ridConfig settings are based on the following assumptions about the data:

- readRecordLimit is set to 5000 because all record types are expected to occur within the first 5000 records of the data set.
- recordTypeLimit is set to 2 because there are no more than two distinct record types in any data set matching this copybook.
- fieldWidth is set to 2 because RID values for all data records are two bytes wide.

Alternatively, you could specify the findRecordIds and ridConfig properties at the global level instead of the data map instance level.

Command Line

To run this example on the Informatica services machine, enter the following command at the command line:

```
Informatica_services_installation_directory\isp\bin\infacmd pwx createDatamaps -
pwxLocation pwx_location -datamapOutputDir Output -controlFile seq_rid.xml -logFile
Output\seq_rid.log -verbosity INFO
```

For the -pwxLocation parameter, specify the location of the PowerExchange Listener as specified in a NODE statement in the PowerExchange DBMOVER configuration file.

Control File

The control file for this example, seq_rid.xml, contains the following lines:

```
<?xml version="1.0" encoding="UTF-8"?>

<DatamapGeneration xmlSchemaVersion="1.0" xmlns="http://
```

```
com.informatica.cmd.pwx.createdatamaps/DatamapGeneration">
    <seqGen>

        <!-- New for 10.0. Data cache configuration. Global setting only. -->
        <cacheConfig>
            <cachePath>c:\temp\imgcache</cachePath>
            <flushDataMode>e</flushDataMode>
        </cacheConfig>

        <globalGenConfig>
            <schemaName>SEQRID</schemaName>
            <datamapName>MAP</datamapName>
        </globalGenConfig>

        <datamapInstances>

            <seqDatamapInstance>
                <genConfig>
                    <!-- New for 10.0. Flag triggers new behaviour. -->
                    <findRecordIds>true</findRecordIds>
                </genConfig>

                <importCopybookDetails>
                    <filePath>
                        <zosPath>DEV.IMG.COBOL(RID1)</zosPath>
                    </filePath>
                </importCopybookDetails>

                <datamapProperties>
                    <seqFileName>
                        <zosPath>DEV.IMG.DATA(RID1)</zosPath>
                    </seqFileName>

                    <!-- New for 10.0. Skip records and record ID configuration -->
                    <skipRecordCount>0</skipRecordCount>

                    <ridConfig>
                        <readRecordLimit>5000</readRecordLimit>
                        <recordTypeLimit>2</recordTypeLimit>
                        <fieldWidth>2</fieldWidth>
                    </ridConfig>
                </datamapProperties>
            </seqDatamapInstance>
        </datamapInstances>
    </seqGen>
</DatamapGeneration>
```

COBOL Copybook File

The COBOL copybook for the data map in this example, DEV.IMG.COBOL(RID1), contains the following lines:

```
 01  NAME_REC.
     04  ACCOUNT               PIC 9(3).
     04  RECTYPE               PIC X(2).
     04  NAME                  PIC X(20).
     04  SEX                   PIC X.
     04  ITEMCT                PIC 9.
     04  ITEMS OCCURS 3 DEPENDING ON ITEMCT PIC X(10).
     04  FILLER                PIC X.
 01  ACCOUNT_REC.
     04  ACCOUNT               PIC 9(3).
     04  RECTYPE               PIC X(2).
     04  AMOUNT                PIC 9(9)V99.
     04  POLICY_DATE           PIC X(8).
     04  FILLER                PIC X.
```

The copybook includes two 01-level records and no REDEFINE statements. The copybook defines two record layouts: one for each 01-level record.

Data File

The data file in this example, DEV.IMG.DATA(RID1), contains the following lines:

```
00501Mark Jones          M3apple     orange    pear      .
0050200000123450 87-12-31.
01001Shirley Wong        F1raspberry .
0100200000000025657-07-04.
02001John Jackson        M2pansy     daisy     .
03001Donald Leary        M0.
0300200000000004566-01-12.
04001David Wu            M1fox       .
05001Jean Connor         F3dog       cat       rabbit    .
0500200000000091196-02-29.
06001Ronald Rose         M2horse     pony      .
0600200001000000 01-01-01.
07001Betsy Martin        F1wolf      .
0700200000640001 41-12-07.
```

The first three characters of each record in this file are the ACCOUNT field. The next two characters of each record are the RECTYPE field.

Identifying RID Fields

For each 01-level record, the createdatamaps utility goes through a three-step process to identify the RID field and associate its values with specific record layouts.

For the **NAME_REC** record, the utility goes through the following steps:

Step 1 examines the metadata model to find candidate RID fields.

The following table shows the results of Step 1 for the NAME_REC record:

Field in NAME_REC Record	Candidate RID Field?
ACCOUNT	No - field exceeds maximum width
REC_TYPE	Yes
NAME	No - field exceeds maximum width
SEX	Yes
ITEMCT	Yes
ITEMS	No - field exceeds maximum width, RID field cannot be a variable-length field

Based on these findings, the createdatamaps utility determines the offset and length of candidate RID fields.

Step 2 reads data records from the DEV.IMG.DATA(RID1) data set that is specified in the control file. Using the offsets and lengths of the candidate fields from Step 1, Step 2 saves the data values for each candidate RID field. The utility compares the number of distinct data values with the value of recordTypeLimit in the control file. Because recordTypeLimit=2, the utility rejects candidate fields with more than two distinct data values in the data file.

The result must be one RID field selected from the candidates. In this example, the REC_TYPE and SEX fields both meet the recordTypeLimit=2 criteria and remain valid candidate fields. The utility chooses the first candidate field, REC_TYPE, as the RID field.

Step 3 generates all possible record layouts. Because the COBOL copybook in this example includes no REDEFINE statements, Step 3 generates only one record layout for the ACCOUNT_REC record type.

After completing Step 3 for the ACCOUNT_REC record type, the utility repeats the three steps for the **NAME_REC** record type.

Step 1 examines the metadata model to find candidate RID fields.

The following table shows the results of Step 1 for the ACCOUNT_REC record:

Field in ACCOUNT_REC record	Candidate RID Field?
ACCOUNT	No - field exceeds maximum width
REC_TYPE	Yes
AMOUNT	No - field exceeds maximum width
POLICY_DATE	No - field exceeds maximum width

Step 2 reads data records from the DEV.IMG.DATA(RID1) dat set. The utility compares the number of distinct data values with the value of recordTypeLimit (2, in this example) in the control file. The result must be one RID field selected from the candidates. In this example, the REC_TYPE field is the only candidate from Step 1 and also meets the Step 2 criteria.

Step 3 generates all possible record layouts. Because the COBOL copybook in this example includes no REDEFINE statements, the step generates only one record layout for the ACCOUNT_REC record type.

After performing the three steps for each 01-level record, the utility creates a data map with the following records:

Record	REC_TYPE Value
NAME_REC	01
ACCOUNT_REC	02

Log File

The log file this example, seq_rid.log, contains the following lines:

```
2015-10-09 12:10:41 INFO [MDO_34613] Configuration for this run: location=MyListener,
user name=, datamap directory=Output, control file=seq_rid.xml
2015-10-09 12:10:46 INFO [MDAdapter_34100] Finding metadata. Path filter =
DEV.IMG.COBOL(RID1)
2015-10-09 12:10:46 INFO [MDAdapter_34101] Fetching file DEV.IMG.COBOL(TST101)
2015-10-09 12:10:47 INFO [MDAdapter_34100] Finding metadata. Path filter =
DEV.IMG.DATA(RID1)
2015-10-09 12:10:47 INFO [MDO_34641] Metadata record 'NAME_REC' - Candidate record id
field 'RECTYPE' found.
2015-10-09 12:10:47 INFO [MDO_34641] Metadata record 'NAME_REC' - Candidate record id
field 'SEX' found.
2015-10-09 12:10:47 INFO [MDO_34641] Metadata record 'NAME_REC' - Candidate record id
field 'ITEMCT' found.
2015-10-09 12:10:48 INFO [MDO_34643] Metadata record 'NAME_REC' - Valid record id field
'RECTYPE' found from reading the data records.
2015-10-09 12:10:48 INFO [MDO_34612] Copybook 'NAME_REC' has 1 possible layouts (Maximum
configured limit is 2).
2015-10-09 12:10:48 INFO [MDO_34641] Metadata record 'ACCOUNT_REC' - Candidate record id
field 'RECTYPE_1' found.
2015-10-09 12:10:48 INFO [MDO_34643] Metadata record 'ACCOUNT_REC' - Valid record id
field 'RECTYPE_1' found from reading the data records.
2015-10-09 12:10:48 INFO [MDO_34612] Copybook 'ACCOUNT_REC' has 1 possible layouts
(Maximum configured limit is 2).
```

```
2015-10-09 12:10:48 INFO [JDMX2_34801] 2 records imported.
2015-10-09 12:10:48 INFO [JDMX2_34802] 12 fields imported.
2015-10-09 12:10:48 INFO [JDMX2_34803] 2 tables imported.
2015-10-09 12:10:48 INFO [MDO_34619] Datamap file 'Output\SEQ_RID.MAP.dmp' was written.
2015-10-09 12:10:48 INFO [MDO_34614] Run complete: 1 datamap(s) created. 0 error and 0
warning messages.
```

Data Map Files

This example creates a data map with the following file name and relative path:

- Output\SEQ_RID.MAP.dmp

CHAPTER 3

DTLCCADW - Adabas PCAT Utility

This chapter includes the following topics:

DTLCCADW Utility Overview

The PCAT utility program, DTLCCADW, is used by the Adabas ECCR process to manipulate the contents of the PCAT file. The PCAT utility is controlled by settings of the parameters passed via the PARM= on the EXEC statement. There are examples of the JCL required for each function in the PowerExchange DTLEXPL library with names DTLCCAD*x*, where *x* corresponds to the parameter value.

Typically, these functions are used only internally by PowerExchange. However, there may be times when manual overrides are desired, which are described below. When in doubt about usage, contact Informatica Global Customer Support.

DTLCCADW Utility Functions

The DTLCCADW utility has the following functions:

- P (Populate PCAT control file)
- R (Report on PCAT control file)
- I (Insert)
- D (Delete)
- L (Reset latest sequence number)
- V (Rebuild the PCAT control file)
- A (Add)
- S (Submit ADASEL)
- T (Submit ET record extraction)
- E (ET/BT record extraction)

P (Populate PCAT Control File) Function

Example job DTLCCADP - no other parameters are required.

This function may be used after the VSAM Control File has been initially established with its 999999999 control record, to pre-populate the PCAT file with previously-created PLOG data set names. By default, when the Adabas PowerExchange ECCR is started, only the most recent archived PLOG will be recognized. So, if there is a need to collect older captured changes, this is the function to use. The list of data set names is input through DDCARD DTLCCADF either directly as SYSIN, or in a file of 80-byte card images. It is the user's responsibility to obtain those PLOG data set names. The 999999999 PCAT control record is then updated with the highest sequence number added.

Note: Use this function only after initializing the control file, not after normal operation has begun.

R (Report on PCAT Control File) Function

Example job DTLCCADR - optionally, a second parameter of control file sequence number.

Prints to SYSOUT with a DD Name of DTLCCRPT. The optional second parameter allows you to specify a file sequence number from where the report will commence. If no second parameter is specified then the whole file is printed to SYSOUT.

Note: The following functions may be of use in case of operational PLOG difficulties, not related to the Adabas PowerExchange Change processing. For instance, if the PLOG files get out of sequence operationally, these functions will ensure that the PCAT can be reset to correct data set name sequence, as well.

I (Insert) Function

Example job DTLCCADI - requires two further parameters.

The first is a PCAT control file sequence number, which must not already exist. The second is the data set name of a PLOG to be inserted. Note - DTLCCADW does NOT check that the PLOG is in the correct chronological sequence - it is the user's responsibility to ensure this.

D (Delete) Function

Example job DTLCCADD - requires a second parameter of control file sequence number.

DTLCCADW reads the PCAT control record and deletes it. If you delete the record which was the latest to be added, you must immediately run the L function (see below) to reset the latest key value in the 999999999 control record.

L (Reset Latest Sequence Number) Function

Example job DTLCCADL - no other parameters are required.

This function re-populates the “latest sequence number added” field in the 999999999 PCAT control record. The only circumstance that this function would be necessary is if the user deletes the record which is the latest added, which would invalidate the '999999999' control record.

V (Rebuild the PCAT Control File) Function

Example job DTLCCADV - no other parameters are required.

This function can be used to delete and re-build the overall PCAT control record '999999999'.

Note: The following functions should be used only under the direction of Informatica Global Customer Support.

A (Add) Function

Example job DTLCCADA - no other parameters required.

Takes the PLOG specified by the data set name in the DDCARD DTLCCPLG and creates an entry in the PCAT file, taking the highest sequence number so far added and adding 100 to it (gaps are left in the sequence in case older PLOGs need to be inserted into the sequence later). This function is automatically invoked during the PLOG flip in the JCL executing the PLCOPY function and so should not be necessary to invoke manually, in normal operation.

S (Submit ADASEL) Function

Example job DTLCCADS - requires a second parameter of PCAT file sequence number.

DTLCCADW reads the PCAT control record specified by the sequence number and constructs an ADASEL job for the PLOG data set name recorded in the control record. It submits the job (by default, DTLSELJC), which runs the ADASEL and creates an output file, the data set name of which is recorded in the control record. This function is automatically invoked by the ECCR and so should not be necessary to invoke manually in normal operation.

T (Submit ET Record Extraction) Function

Example job DTLCCADT - requires a second parameter of PCAT file sequence number.

DTLCCADW reads the PCAT control record specified by the sequence number and constructs another DTLCCADW job for the PLOG recorded in the control record, building a data set name for the output ET file using date and time parameters. It submits the job (by default, DTLETLJC), which reads the PLOG specified in the control record and creates an output file of ET/BT records, the data set name of this file then being recorded in the control record. This function is normally invoked by the ECCR and so should not be necessary to invoke manually in normal operation.

E (ET/BT Record Extraction) Function

Example job DTLCCADE - requires a second parameter of PCAT file sequence number.

This function is in fact the same as the job which is dynamically created and submitted by the T function above - the difference being that the user has to explicitly define the data set name of the output ET/BT file in the JCL, DDNAME DTLCCETL, and the name of the archived PLOG being processed in DDNAME DTLCCPLG. The ECCR normally controls this operation and this function is only provided in case of difficulties which might require manual intervention.

CHAPTER 4

DTLCUIML - IMS Log Marker Utility

This chapter includes the following topics:

DTLCUIML Utility Overview

Use the DTLCUIML utility to define a marker for the IMS log-based ECCR in the IMS system log data set (SLDS). Once the IMS log-based ECCR encounters one of the markers, it triggers a message in the PowerExchange Logger which stipulates a Restart and Sequence Token for the affected Registration Tags.

These Tokens can then be used as input for the Application Maintenance Utility (DTLUAPPL) to define the start point for an extraction.

There is no limit or restriction on the number of markers being set in the IMS SLDS. The IMS Log Record ID chosen has to be unique for the individual installation, and the number needs to be part of the input parameters for the IMS log-based ECCR.

This utility is used to write user-defined records to the IMS log.

The parameters controlling the utility are specified in the SYSIN file in the JCL.

The utility runs as a standard IMS application program. There is no need to provide a specific PSB. The utility can use any PSB as long as the first PCB in the PSB is an IOPCB. The utility uses the IMS LOG Call to write IMS log records.

This utility must run as an IMS BMP job. This ensures that the IMS Log record is written into the IMS logstream and that the associated log is read by the IMS log-based Collector. In an IMS DCI situation the DTLUAPPL utility has to be used to establish an extraction point for the changed data.

DTLCUIML Utility Parameters

Each SYSIN record contains the following parameters:

- DBDNAME. IMS DBD name.
- DBID. IMS instance (Recon Identifier).
- RECID. A value in (uppercase) hexadecimal from A0 through FF. It defines the log record type for the user-defined IMS log record, so it should be different to any other user-defined values which the site is using.

Leading spaces are ignored. Records are ignored where the first non-space characters are /* so can be used as comments.

Example:

```
//SYSIN    DD *
  DBDNAME=DTLD004,DBID=IMS7,RECID=A0
  DBDNAME=DTLD006,DBID=IMS7,RECID=A0
  DBDNAME=DTLD007,DBID=IMS7,RECID=A0
/*
```

DTLCUIML Utility Reports

File SYSPRINT reports validation of the input parameters and progress in writing to the IMS log.

File DFSSTAT reports IMS activity.

Sample JCL is supplied in member IMSLOGW.

SYSPRINT: Control Report

The control report shows the following information:

- Date and time when the program started. This time is also used on each user-defined log record written to the IMS log.
- Validation messages for the SYSIN records. If any record is invalid, the run aborts and no records are written to the IMS log.
- Progress messages as the records are written to the IMS log.

Example:

```
2002-10-15 14:06:14                  DTLCUIML  REPORT
======================================================================
.
Input Records Read
------------------
  DBDNAME=DTLD004,DBID=IMS1,RECID=A0
  DBDNAME=DTLD006,DBID=IMS1,RECID=A0
  DBDNAME=DTLD007,DBID=IMS1,RECID=A0
3 record(s) validated from the input file
.
LOG record processing begins
-----------------------------
Processing dbname=DTLD004 dbid=IMS1 recid=A0 timestamp=20021015140614
Processing dbname=DTLD006 dbid=IMS7 recid=A0 timestamp=20021015140614
Processing dbname=DTLD007 dbid=IMS7 recid=A0 timestamp=20021015140614
.
Number of LOG calls = 3
.
Run completed successfully
```

DFSSTAT: IMS Activity Report

Counts for SYS LOG CALLS will match the number of records processed from file SYSIN. All other counts are zero.

Example:

```
//DFSSTAT STATISTICS FOR: JOB=UIMLRUN   STEP=G
-----------------------------------------------
            *** PST ACCOUNTING STATISTICS ***
SYS LOG CALLS               3
```

User-Defined Log Records

Each user-defined log record contains 35 bytes of user data. The actual IMS log record adds the standard IMS suffix to this data.

The following table describes the user-defined log records:

Field	Start	Length	Type	Description
Length	1	2	unsigned binary	Length of user-defined log record = 35 bytes.
Zeros	3	2	unsigned binary	Always hex '0000'.
Recid	5	1	char	Record ID supplied in SYSIN parameters, such as hex 'A0'.
Dbname	6	8	char	IMS DBNAME.
Dbid	14	8	char	IMS instance (Recon Identifier).
Timestamp	22	14	char	Time when program DTLCUIML ran.

CHAPTER 5

DTLINFO - Release Information Utility

This chapter includes the following topics:

DTLINFO Utility Overview

Use the DTLINFO utility to perform the following functions:

- Display the version, release, and release level for PowerExchange or for a specific PowerExchange module.
- Verify the installation of the product, a service pack, or a hotfix. For example, use the utility to determine the maintenance level of your PowerExchange software at the request of Informatica Global Customer Support.

Supported Operating Systems for the DTLINFO Utility

The DTLINFO utility can run on the following operating systems:

- i5/OS
- Linux and UNIX

- Windows
- z/OS

Control Statement Syntax for the DTLINFO Utility

Use the following syntax:

```
DTLINFO [module_name]
```

To view the release information for the PowerExchange product do not specify the *module_name* parameter.

To view the release information for a specific PowerExchange module, use the *module_name* parameter. The module name is the name of any program included in your PowerExchange installation.

Control Statement Parameters for the DTLINFO Utility

The DTLINFO utility has the following optional parameter:

module_name

Displays the version, release, and release level for a specific PowerExchange module, such as DTLREXE.

Running the DTLINFO Utility on i5/OS

To run the DTLINFO utility on i5/OS:

▶ To view release information for PowerExchange, enter:

```
CALL PGM(dtllib/DTLINFO)
```

To view release information for a PowerExchange module, enter:

```
CALL PGM(dtllib/DTLINFO) parm ('module_name')
```

Running the DTLINFO Utility on Linux, UNIX, and Windows

To run the DTLINFO utility on Linux, UNIX, and Windows:

1. Navigate to the Informatica PowerExchange directory.
2. Enter the dtlinfo statement in one of the following ways:

To view release information for PowerExchange, enter:

```
dtlinfo
```

To view release information for a specific PowerExchange module, enter:

```
dtlinfo module_name
```

Running the DTLINFO Utility on z/OS

The JCL for the DTLINFO utility is located in *hlq*.RUNLIB(DTLINFO), where *hlq* is the high-level qualifier used for installing PowerExchange. The DTLINFO program is located in *hlq*.LOADLIB(DTLINFO)

You can incorporate the DTLINFO job step into a batch job, or add a job card and run the DTLINFO job separately.

To run the DTLINFO utility on z/OS:

1. Define the JCL EXEC statement for the DTLINFO program.

   To view release information for the PowerExchange product, do not specify a PARM value or SYSIN DD as shown in the following syntax:

   ```
   //BLDSTEP   EXEC PGM=DTLINFO
   //STEPLIB   DD DISP=SHR,DSN=hlq.LOADLIB
   //SYSPRINT  DD SYSOUT=*
   ```

   To view release information for a specific PowerExchange module, specify a module name as the PARM value. Also, supply the library and member name for the module by using the SYSIN DD as shown in the following sample:

   ```
   //BLDSTEP   EXEC PGM=DTLINFO,PARM=('DTLREXE')
   //STEPLIB   DD DISP=SHR,DSN=hlq.LOADLIB
   //SYSIN     DD DISP=SHR,DSN=hlq.LOADLIB(DTLREXE)
   //SYSPRINT  DD SYSOUT=*
   ```

   The JCL statements are:

   EXEC PGM=DTLINFO

   Invokes the utility.

   STEPLIB DD

   Defines the PowerExchange LOAD library that contains the utility.

   SYSPRINT DD

   Defines the print location for the report.

2. Submit the DTLINFO job.

DTLINFO Utility on i5/OS Examples

The following are examples of the DTLINFO utility on i5/OS.

DTLINFO Utility on i5/OS - Example 1

The following command displays copyright and release build information for the PowerExchange installation:

```
CALL DTLINFO
```

Example output:

```
DTLINFO Latest Version:
Copyright ⌐ 1993-2016 Informatica LLC. All rights reserved.
See patents at https://www.informatica.com/legal/patents.html.
Segment#Revision :<//pwx/prod/v961/main/source/dtlinfod/dtlinfod.
Build            :<V961_HOTFIX4_4333731><Jan  5 2016 00:13:38>
```

DTLINFO Utility on i5/OS - Example 2

The following command displays copyright and release build information for the PowerExchange module DTLREXE:

```
CALL DTLINFO DTLREXE
```

Example output:

```
DTLINFO Embedded Version History:
Copyright ⌐ 1993-2016 Informatica LLC. All rights reserved.
See patents at https://www.informatica.com/legal/patents.html.

Segment#Revision :<//pwx/prod/v961/main/source/dtlinfo/dtlinfo.c
Build            :<V961_HOTFIX4_4333731><Jan  5 2016 00:13:52>

Segment#Revision :<//pwx/prod/v961/main/source/dtlrexe/dtlrexe.c
Build            :<V961_HOTFIX4_4263007><Nov  9 2015 12:15:52>
```

DTLINFO Utility on Linux, UNIX, and Windows Examples

The following are examples of the DTLINFO utility on Linux, UNIX, and Windows.

DTLINFO Utility on Linux, UNIX, and Windows - Example 1

The following command displays the release information for PowerExchange:

```
dtlinfo
```

DTLINFO Utility on Linux, UNIX, and Windows - Example 2

The following command displays the release information for the PowerExchange module DTLREXE:

```
dtlinfo dtlrexe.exe
```

DTLINFO Utility on z/OS Examples

The following are examples of the DTLINFO utility on z/OS.

DTLINFO Utility on z/OS - Example 1

The following JCL EXEC statement does not specify a PARM value or SYSIN DD for the DTLINFO program:

```
//BLDSTEP EXEC PGM=DTLINFO
//STEPLIB DD DISP=SHR,DSN=hlq.LOADLIB
//SYSPRINT DD SYSOUT=*
```

DTLINFO Utility on z/OS - Example 2

The following JCL EXEC statement specifies the PowerExchange module DTLREXE as the PARM value. Also, the following SYSIN DD provides the library and member name for the module:

```
//BLDSTEP EXEC PGM=DTLINFO,PARM=('DLTREXE')
//STEPLIB DD DISP=SHR,DSN=hlq.LOADLIB
//SYSIN DD DISP=SHR,DSN=hlq.LOADLIB(DLTREXE)
//SYSPRINT DD SYSOUT=*
```

CHAPTER 6

DTLREXE - Remote Execution Utility

This chapter includes the following topics:

DTLREXE Utility Overview

Use the DTLREXE utility to perform the following tasks:

- Ping a remote PowerExchange Listener.
- Submit a remote z/OS job.
- Delete a file from a remote system.
- Run a file on a remote system.

Supported Operating Systems for the DTLREXE Utility

The DTLREXE utility can run on the following operating systems:

- i5/OS
- UNIX and Linux

- Windows
- z/OS

Control Statement Syntax for the DTLREXE Utility

Use the following syntax for the DTLREXE utility control statements:

```
prog=delete
loc=location
parms=file_name
[uid=userid]
[{pwd=password|epwd=epassword}]
prog=ping
loc=location
[uid=userid]
[{pwd=password|epwd=epassword}]
prog=submit
loc=location
[uid=userid]
[{pwd=password|epwd=epassword}]
[fn="your jcl"]
[mode=(job|task},{wait|nowait|timed)]
[time=<time_in_seconds>]
[submittimeout=timeout_in_seconds]
[output=output.file]
[result=result.file]
prog=system
loc=location
parms=file_name
```

Control Statement Parameters for the DTLREXE Utility

DTLREXE has the following statements:

- DELETE
- PING
- SUBMIT
- SYSTEM

DELETE Statement

Use DTLREXE DELETE to delete a file from the platform where the PowerExchange Listener is running.

DELETE has the following parameters:

loc

Optional. Location of the PowerExchange Listener as defined in a NODE statement in the DBMOVER configuration file.

parms

Required. The name of the file to delete. On z/OS, if you do not enclose the name in quotes, the utility provides them.

prog

Required. Set to DELETE.

{pwd|epwd}

Optional. A password or encrypted password for the specified user ID. For a location on i5/OS or z/OS, you can enter a passphrase or encrypted passphrase instead.

- **pwd**. A clear text password for the specified user, which allows access to the target. If a password contains nonalphanumeric characters, it must be enclosed in double quotation marks (“). A password cannot contain embedded double quotation marks.

  For access to an i5/Os or z/OS location, you can enter a valid PowerExchange passphrase instead of password. An i5/OS passphrase can be from 9 to 31 characters in length. A z/OS passphrase can be from 9 to 128 characters in length. A passphrase can contain the following characters:

  - Uppercase and lowercase letters
  - The numbers 0 to 9
  - Spaces
  - The following special characters:

    ```
    ' - ; # \ , . / ! % & * ( ) _ + { } : @ | < > ?
    ```

  Note: The first character is an apostrophe.

  Passphrases cannot include single quotation marks (‘), double quotation marks (“), or currency symbols.

  If a passphrase contains spaces, you must enclose it with double-quotation marks ("), for example, `"This is a passphrase"`. If a passphrase contains special characters, you must enclose it with triple double-quotation characters ("""), for example, `"""This passphrase contains special characters ! % & *."""`. If a passphrase contains only alphanumeric characters without spaces, you can enter it without delimiters.

  To use passphrases, ensure that the PowerExchange Listener runs with a security setting of SECURITY=(1,N) or higher in the DBMOVER configuration member. On z/OS, the DTLCFG DD statement in the DTLREXE JCL points to the DBMOVER member. On i5/OS, the DBMOVER member is the CFG file in the PowerExchange *datalib* library. For more information, see "SECURITY Statement" in the *PowerExchange Reference Manual*.

  Note: On z/OS, a valid RACF passphrase can be up to 100 characters in length. PowerExchange truncates passphrases longer than 100 characters when passing them to RACF for validation.

- **epwd**. An encrypted password for the specified user.

  For access to an i5/OS or z/OS location, you can enter an encrypted PowerExchange passphrase instead of an encrypted password. Do not encrypt a passphrase that contains invalid characters, such as double-quotation marks, single quotation marks, or currency symbols.

  Note: You can use the PowerExchange Navigator to encrypt a password or passphrase.

uid

Optional. A user ID that allows access to the specified location. If you specify a user ID, you must also enter a **pwd** or **epwd** value, but do not enter both.

For a location on a supported Linux, UNIX, or Windows system, if you have enabled PowerExchange LDAP user authentication, the user ID is the enterprise user name. For more information, see the *PowerExchange Reference Manual*.

PING Statement

Use DTLREXE PING to prove basic connectivity to a PowerExchange Listener.

You must configure a NODE statement in the DBMOVER configuration file on the machine from which you issue the DTLREXE PING.

PING has the following parameters:

loc

Optional. Location of the PowerExchange Listener as defined in a NODE statement in the DBMOVER configuration file.

prog

Required. Set to PING.

{pwd|epwd}

Optional. A password or encrypted password for the specified user ID. For a location on i5/OS or z/OS, you can enter a passphrase or encrypted passphrase instead.

- **pwd**. A clear text password for the specified user, which allows access to the target. If a password contains nonalphanumeric characters, it must be enclosed in double quotation marks (“). A password cannot contain embedded double quotation marks.

  For access to an i5/Os or z/OS location, you can enter a valid PowerExchange passphrase instead of password. An i5/OS passphrase can be from 9 to 31 characters in length. A z/OS passphrase can be from 9 to 128 characters in length. A passphrase can contain the following characters:

  - Uppercase and lowercase letters
  - The numbers 0 to 9
  - Spaces
  - The following special characters:

    ```
    ' - ; # \ , . / ! % & * ( ) _ + { } : @ | < > ?
    ```

  Note: The first character is an apostrophe.

  Passphrases cannot include single quotation marks (‘), double quotation marks (“), or currency symbols.

  If a passphrase contains spaces, you must enclose it with double-quotation marks ("), for example, `"This is a passphrase"`. If a passphrase contains special characters, you must enclose it with triple double-quotation characters ("""), for example, `"""This passphrase contains special characters ! % & *."""`. If a passphrase contains only alphanumeric characters without spaces, you can enter it without delimiters.

  To use passphrases, ensure that the PowerExchange Listener runs with a security setting of SECURITY=(1,N) or higher in the DBMOVER configuration member. On z/OS, the DTLCFG DD statement in the DTLREXE JCL points to the DBMOVER member. On i5/OS, the DBMOVER member is the CFG file in the PowerExchange *datalib* library. For more information, see "SECURITY Statement" in the *PowerExchange Reference Manual*.

  Note: On z/OS, a valid RACF passphrase can be up to 100 characters in length. PowerExchange truncates passphrases longer than 100 characters when passing them to RACF for validation.

- **epwd**. An encrypted password for the specified user.

  For access to an i5/OS or z/OS location, you can enter an encrypted PowerExchange passphrase instead of an encrypted password. Do not encrypt a passphrase that contains invalid characters, such as double-quotation marks, single quotation marks, or currency symbols.

  Note: You can use the PowerExchange Navigator to encrypt a password or passphrase.

uid

Optional. A user ID that allows access to the specified location. If you specify a user ID, you must also enter a **pwd** or **epwd** value, but do not enter both.

For a location on a supported Linux, UNIX, or Windows system, if you have enabled PowerExchange LDAP user authentication, the user ID is the enterprise user name. For more information, see the *PowerExchange Reference Manual*.

SUBMIT Statement

Use DTLREXE SUBMIT to submit a job to a remote z/OS platform or server.

Alternatively, you can supply the cs parameter to point to a parameter file that contains all the required parameters.

```
dtlrexe cs=<parameter_file>
```

SUBMIT has the following parameters:

fn

Optional. The name of the file that contains the JCL to be submitted, including the job name. Use the following format:

```
fn="dtlusr.jcl(yourjob)"
```

On Windows, use the following format:

```
fn=\"dtlusr.jcl(yourjob)\"
```

loc

Location of the PowerExchange Listener as defined in a NODE statement in the DBMOVER configuration file.

mode

Optional. Specifies the submit mode. Use the following format:

```
mode=(job|task},{wait|nowait|timed)
```

The available modes are:

- **job**. A submitted job
- **task**. A started task. (Not currently supported.)
- **wait**. Synchronous. Report result at the end and wait for completion.
- **nowait**. Asynchronous. Submit the job but do not wait until report completion.
- **timed**. Synchronous. Waits for the length of time that is specified by the time parameter.

output

Optional. The file name for the file that contains the results from the job. Use the following format:

```
output=dtlusr.output
```

If the output is a PDS member, the same format requirements are in place as for the **fn** parameter.

prog

Required. Set to SUBMIT.

{pwd|epwd}

Optional. A password or encrypted password for the specified user ID. For a location on i5/OS or z/OS, you can enter a passphrase or encrypted passphrase instead.

- **pwd**. A clear text password for the specified user, which allows access to the target. If a password contains nonalphanumeric characters, it must be enclosed in double quotation marks (“). A password cannot contain embedded double quotation marks.

  For access to an i5/Os or z/OS location, you can enter a valid PowerExchange passphrase instead of password. An i5/OS passphrase can be from 9 to 31 characters in length. A z/OS passphrase can be from 9 to 128 characters in length. A passphrase can contain the following characters:

  - Uppercase and lowercase letters
  - The numbers 0 to 9
  - Spaces
  - The following special characters:

    ```
    ' - ; # \ , . / ! % & * ( ) _ + { } : @ | < > ?
    ```

  Note: The first character is an apostrophe.

  Passphrases cannot include single quotation marks (‘), double quotation marks (“), or currency symbols.

  If a passphrase contains spaces, you must enclose it with double-quotation marks ("), for example, `"This is a passphrase"`. If a passphrase contains special characters, you must enclose it with triple double-quotation characters ("""), for example, `"""This passphrase contains special characters ! % & *."""`. If a passphrase contains only alphanumeric characters without spaces, you can enter it without delimiters.

  To use passphrases, ensure that the PowerExchange Listener runs with a security setting of SECURITY=(1,N) or higher in the DBMOVER configuration member. On z/OS, the DTLCFG DD statement in the DTLREXE JCL points to the DBMOVER member. On i5/OS, the DBMOVER member is the CFG file in the PowerExchange *datalib* library. For more information, see "SECURITY Statement" in the *PowerExchange Reference Manual*.

  Note: On z/OS, a valid RACF passphrase can be up to 100 characters in length. PowerExchange truncates passphrases longer than 100 characters when passing them to RACF for validation.

- **epwd**. An encrypted password for the specified user.

  For access to an i5/OS or z/OS location, you can enter an encrypted PowerExchange passphrase instead of an encrypted password. Do not encrypt a passphrase that contains invalid characters, such as double-quotation marks, single quotation marks, or currency symbols.

  Note: You can use the PowerExchange Navigator to encrypt a password or passphrase.

result

Optional. The file to which the results from the job are written on the client platform where DTLREXE runs.

The file specification must be suitable for the platform.

If the output is a PDS member, the same format requirements are in place as for the **fn** parameter.

submittimeout

Optional. The time, in seconds, to wait for the submitted job to start running.

time

Optional. The time, in seconds, to wait for the job to return results. This wait period starts when the job is submitted.

uid

Optional. A user ID that allows access to the specified location. If you specify a user ID, you must also enter a **pwd** or **epwd** value, but do not enter both.

For a location on a supported Linux, UNIX, or Windows system, if you have enabled PowerExchange LDAP user authentication, the user ID is the enterprise user name. For more information, see the *PowerExchange Reference Manual*.

SYSTEM Statement

Use DTLREXE SYSTEM to run a file from the path or steplib. You can specify any file that can be run, such as a batch file, rexx, or executable file.

SYSTEM has the following parameters.

loc

Location as defined in the dbmover.cfg as a node giving the address of the PowerExchange Listener.

parms

Required. The name of the file to run.

prog

Required. Set to SYSTEM.

Running the DTLREXE Utility on i5/OS

To run the DTLREXE utility on i5/OS:

- Enter the following command:

```
CALL PGM(DTLREXE) PARM('prog=submit loc=mvs fn=dtlusr.load.jcl mode=(job,wait)
output=dtlusr.output, result=dtlusr.result')
```

Running the DTLREXE Utility on Linux and UNIX

You can run the DTLREXE utility on Linux and UNIX specifying either a PDS member or a sequential MVS data set.

Submitting a Remote z/OS Job Specifying a PDS Member

To submit a remote z/OS job specifying a PDS member:

- Enter the following command, specifying a PDS member as follows:

```
dtlrexe prog=submit loc=remlist fn=\"dtlusr.jcl.cntl'('db2load')'\",
mode='('job,wait')', output=dtlusr.output, result=/usr/pwx/output.txt
```

Alternatively, you can enter the following command:

```
dtlrexe prog=submit cs=/usr/pwx/MyParameterFile.txt
```

Submitting a Remote z/OS Job Specifying a Sequential MVS Data Set

To submit a remote z/OS job specifying a sequential MVS data set:

- Enter the following command, specifying a sequential MVS data set as follows:

```
dtlrexe prog=submit loc=remlist fn=dtlusr.load.jcl, mode='('job,wait')',
output=dtlusr.output, result=/usr/pwx/output.txt
```

  Alternatively, you can enter the following command:

```
dtlrexe prog=submit loc=remlist fn="dtlusr.load.jcl", mode='('job,wait')',
output=dtlusr.output, result=/usr/pwx/output.txt
```

Deleting a File from a Remote System

To delete a file from a remote system:

- Enter the following command:

```
dtlrexe prog=delete loc=location parms=file_name
```

Running a File on a Remote System

To run a file on a remote system:

- Enter the following command:

```
dtlrexe prog=system loc=location parms=file_name
```

  For example:

```
dtlrexe prog=system loc=node1 parms=Q:/\mydir/\myprog.bat
```

Running the DTLREXE Utility on Windows

To run the DTLREXE utility on Windows:

- Enter the following command:

```
dtlrexe prog=submit loc=remlist fn=\"dtlusr.jcl.cntl(db2load)\" mode=(job,nowait)
output=dtlusr.output result=c:\submit\output\output.txt uid=user01 pwd=pass01
```

  Alternatively, you can enter the following command:

```
dtlrexe prog=submit cs=c:\PowerExchange\MyParameterFile.txt
```

Related Topics:

- "Running the DTLREXE Utility on Linux and UNIX" on page 87

Running the DTLREXE Utility on z/OS

You can run the DTLREXE utility on z/OS with PROG=SUBMIT, PROG=PING, PROG=DELETE, or PROG=SYSTEM. In each case, the JCL statements are:

JOB

Initiates the job.

EXEC PGM=DTLREXE

Invokes the utility.

STEPLIB DD

Defines the PowerExchange LOAD library that contains the utility.

SYSPRINT DD

Defines the print location for the report.

Running the DTLREXE Utility with PROG=SUBMIT

To run the DTLREXE utility with PROG=SUBMIT:

1. Edit the DTLREXE job JCL. The following two lines must be the first step of the job:

```
//START    EXEC PGM=DTLNTS,PARM='"%STRTJOB"'
//STEPLIB  DD DSN=&HLQ..LOADLIB,DISP=SHR
```

   Then use the following JCL for the DTLREXE job step:

```
//STEP1   EXEC PGM=DTLREXE,
//             PARM=('CS=DD:INCMD'),
//             REGION=0M,TIME=NOLIMIT

//INCMD    DD *
LOC=NODE1 PROG=SUBMIT FN="DTLUSR.JCL(MYJOB)"
MODE=(JOB,WAIT) OUTPUT=DTLUSR.DB2LOAD.SYSPRINT
RESULT="DTLUSR.JCLRESTXT)"
```

   After the final step, you must add the following lines:

```
//         IF ((RC > 4) | (ABEND=TRUE)) THEN
//*
//ENDERR   EXEC PGM=DTLNTS,
//   PARM='"%ENDJOB" C 16'
//STEPLIB  DD DSN=&HLQ..LOADLIB,DISP=SHR
//SYSPRINT DD SYSOUT=*
//*
//         ELSE
//*
//ENDOK    EXEC PGM=DTLNTS,
//  PARM='"%ENDJOB"'
//STEPLIB  DD DSN=&HLQ..LOADLIB,DISP=SHR
//SYSPRINT DD SYSOUT=*
//         ENDIF
```

2. Verify the JCL.
3. Submit the DTLREXE job.

Running the DTLREXE Utility with PROG=PING

To run the DTLREXE utility with PROG=PING:

1. Edit the job that you intend to submit using DTLREXE as follows:

```
//STEP1     EXEC PGM=DTLREXE,
//          PARM='loc=node1 prog=ping'
//STEPLIB   DD DSN=CEE.SCEERUN,
//             DISP=SHR
//          DD DSN=&HLQ..LOADLIB,
```

```
//             DISP=(SHR)
//SYSPRINT  DD SYSOUT=*
```

Enter the location of the PowerExchange Listener in the loc parameter.

2. Submit the DTLREXE job.

Running the DTLREXE Utility with PROG=DELETE

To run the DTLREXE utility with PROG=DELETE:

1. Edit the job that you intend to submit using DTLREXE as follows:

```
//STEP1     EXEC PGM=DTLREXE,
//          PARM='loc=node1 prog=delete parms=file_name'
//STEPLIB   DD DSN=CEE.SCEERUN,
//             DISP=SHR
//          DD DSN=&HLQ..LOADLIB,
//             DISP=(SHR)
//SYSPRINT  DD SYSOUT=*
```

Enter the location of the PowerExchange Listener in the loc parameter.

2. Submit the DTLREXE job.

Running the DTLREXE Utility with PROG=SYSTEM

To run the DTLREXE utility with PROG=SYSTEM:

1. Edit the job that you intend to submit using DTLREXE as follows:

```
//STEP1     EXEC PGM=DTLREXE,
//          PARM='loc=node1 prog=system parms=file_name'
//STEPLIB   DD DSN=CEE.SCEERUN,
//             DISP=SHR
//          DD DSN=&HLQ..LOADLIB,
//             DISP=(SHR)
//SYSPRINT  DD SYSOUT=*
```

Enter the location of the PowerExchange Listener in the loc parameter.

2. Submit the DTLREXE job.

DTLREXE Utility Usage Notes

Consider the following points before using the DTLREXE utility:

- DTLREXE submits the job on the host named in the loc parameter.
- If the mode is (job,nowait), the output and result data sets are of no interest.
- If the mode is (job,wait) or (job,timed), PowerExchange waits for the job to complete and reads the return code. The parameters are required to ensure that the job has completed and the output data set is available.
- Substitution is performed on the job for the %STRTJOB and %STRTJOB tokens.

The following table describes the %STRTJOB and %STRTJOB tokens:

Parameter	Description
%STRTJOB	The name token for the first step in the JCL of the job that is to be submitted. - If the mode parameter is set to (job,wait/timed), %STRTJOB is substituted with a name token generated by the submitter. - If the mode parameter is not set to (job,wait/timed), %STRTJOB is set to DONOTRETURNTOKEN.
%ENDJOB	The name token for the last step in the JCL of the job that is to be submitted. The wait/timed processing retrieves these values to determine if the job has started, is running, or has finished. The %ENDJOB steps have to be included manually and are shown in the sample JCL. If the submitted job fails with a return code greater than four, rc=16 is returned back to DTLREXE on the client.

- To print help on the utility, run DTLREXE without any parameters.

DTLREXE Utility on z/OS Example

To launch DTLREXE from a z/OS job you must use PowerExchange command set syntax as follows:

```
//STEP1   EXEC PGM=DTLREXE,
//             PARM=('CS=DD:INCMD'),
//             REGION=0M,TIME=NOLIMIT
```

An inline DD is specified in the JCL above. You can change this to an external member.

The specified inline or external DD contains the parameters of the DTLREXE command. The following JCL defines the inline DD:

```
//INCMD    DD *
LOC=NODE1 PROG=SUBMIT FN="DTLUSR.JCL(MYJOB)"
MODE=(JOB,WAIT) OUTPUT=DTLUSR.DB2LOAD.SYSPRINT
RESULT="DTLUSR.JCLRESTXT)"
```

The following JCL specifies the external member:

```
//INCMD    DD DSN=HLQ..RUNLIB(MYCS)
```

The member MYCS has the following contents:

```
LOC=NODE1 PROG=SUBMIT FN="DTLUSR.JCL(MYJOB)"
MODE=(JOB,WAIT) OUTPUT=DTLUSR.DB2LOAD.SYSPRINT
RESULT="DTLUSR.JCLRESTXT)"
```

DTLREXE Utility on z/OS Example JCL

This example uses the following JCL:

```
//DTLREXE  JOB 'DTLREX',MSGLEVEL=(1,1),MSGCLASS=X,CLASS=A,
//             NOTIFY=&SYSUID
//*
//         SET HLQ=DTLUSR.V850
//*
//STEP1   EXEC PGM=DTLREXE,REGION=24M,
//             PARM=('CS=DD:INCMD')
//STEPLIB  DD  DSN=CEE.SCEERUN,DISP=SHR
//         DD  DSN=&HLQ..LOADLIB,DISP=SHR
```

```
//DTLCFG   DD  DSN=&HLQ..RUNLIB(DBMOVER),DISP=SHR
//DTLKEY   DD  DSN=&HLQ..RUNLIB(LICENSE),DISP=SHR
//DTLMSG   DD  DSN=&HLQ..DTLMSG,DISP=SHR
//DTLLOG   DD  DSN=&HLQ..LOG,DISP=SHR
//SYSUDUMP DD  SYSOUT=*
//SYSOUT   DD  SYSOUT=*
//SYSPRINT DD  SYSOUT=*
//CEEDUMP  DD  SYSOUT=*
//SYSIN    DD  *
//* EXECUTE THE PROCEDURE
//*
//DTLLSTN EXEC DBMOVE
//INCMD    DD *
 LOC=NODE1 PROG=SUBMIT FN="DTLUSR.JCL(MYJOB)"
 MODE=(JOB,WAIT) OUTPUT=DTLUSR.DB2LOAD.SYSPRINT
 RESULT="DTLUSR.JCL(RESTXT)"
```

DTLREXE Utility on z/OS Output Data Set

The output parameter indicates a data set that should contain the results of the submitted job.

When the job completes, the output is read and transferred back to the client where it is written to a file specified by the `result=` parameter.

The format of the output is:

timestamp|jobid|text

An example of the output is:

```
20060223172636000000|JOB03370|1DSNU000I DSNUGUTC - OUTPUT START FOR UTILITY, UTILID =
DB2LDJCL |
20060223172636000000|JOB03370|0DSNU050I DSNUGUTC - LOAD DATA RESUME NO REPLACE LOG YES|
20060223172636000000|JOB03370| DSNU650I -DSN7 DSNURWI - INTO TABLE DTLUSR.T3|
20060223172636000000|JOB03370| DSNU650I -DSN7 DSNURWI - (COL1 POSITION(3) CHAR(100)
NULLIF(1='Y'),|
20060223172636000000|JOB03370| DSNU650I -DSN7 DSNURWI - COL2 POSITION(*) CHAR(100)
NULLIF(2='Y'))|
20060223172636000000|JOB03370| DSNU350I -DSN7 DSNURRST - EXISTING RECORDS DELETED FROM
TABLESPACE|
20060223172636000000|JOB03370| DSNU304I -DSN7 DSNURWT - (RE)LOAD PHASE STATISTICS -
NUMBER OF RECORDS=3 FOR TABLE DTLUSR.T3 |
20060223172636000000|JOB03370| DSNU302I DSNURILD - (RE)LOAD PHASE STATISTICS - NUMBER OF
INPUT RECORDS PROCESSED=3 |
20060223172636000000|JOB03370| DSNU300I DSNURILD - (RE)LOAD PHASE COMPLETE, ELAPSED
TIME=00:00:08|
20060223172636000000|JOB03370| DSNU010I DSNUGBAC - UTILITY EXECUTION COMPLETE, HIGHEST
RETURN CODE=0 |
```

CHAPTER 7

DTLUAPPL - Restart Token Utility

This chapter includes the following topics:

DTLUAPPL Utility Overview

Use the DTLUAPPL utility to generate restart tokens for CDC sessions and to add, modify, or print application information in the CDEP file. DTLUAPPL can generate restart tokens for all PowerExchange CDC source types and for CDC sessions that use either ODBC or PowerExchange Client for PowerCenter (PWXPC) connections.

Restart tokens determine the point in the change stream at which CDC sessions begin extracting change data. Restart tokens are composed of a pair of sequence and restart token values. The lengths of the token values depend on the data source type and whether you use ODBC or PWXPC.

If you use ODBC connections to extract change data, PowerExchange maintains application names, registrations associated with each application name, and the restart tokens in the CDEP file. When you use the utility to add or modify applications, the utility updates the CDEP file. If an application name entry does not exist when the initial extraction process runs, PowerExchange creates an application entry in the CDEP file. To generate current restart tokens for the initial extraction process, run the utility immediately after target materialization and before any change data has been captured or applied. After the initial extraction process runs, you can use DTLUAPPL to generate current restart tokens for an application.

If you use PWXPC connections to extract change data, the CDEP file is not used. After you materialize the target tables and before you start the CDC session, run DTLUAPPL to generate restart tokens that identify the starting point for extraction processing. Then update the PWXPC restart token file with these generated restart tokens. If you rematerialize target tables for any reason, you can run the utility again to generate current restart tokens.

Note: The utility generates sequence tokens in the format that is used by ODBC extractions. To use generated sequence tokens for a PWXPC session, use the utility to print the sequence and restart token values for the capture registrations associated with the application name. Then add eight zeroes to the end of the printed sequence values.

Alternatively, you can generate current restart tokens from the PowerExchange Navigator. For more information, see the *PowerExchange Navigator User Guide*.

Supported Operating Systems for the DTLUAPPL Utility

The DTLUAPPL utility can run on the following operating systems:

- i5/OS
- Linux and UNIX
- Windows
- z/OS

DTLUAPPL Control Statement Syntax

Use the following syntax to add or modify an application, generate restart tokens at the application level or registration level, or print information for an application:

```
UID user_ID EPWD encrypted_password [CONN_OVER capi_connection_name]
{ADD|MOD} APPL application_name instance [RSTTKN GENERATE]
    [CAPTMETH=access_method]
    [CONDTYPE=P]
    [JRN=library/journal]
    [ORACOLL=collection_id] [ORACONN=connection] [ORAINST=instance]
    [ORASCHEMA=schema]
    [UDBDB=database]

    {ADD|MOD} RSTTKN registration_name [DB=library/table] [GENERATE]
            [SEQUENCE sequence_token]
            [RESTART restart token]
 END APPL application_name
[PRINT APPL {application_name|ALL}]
```

In this syntax:

- The UID, EPWD, and CONN_OVER parameters comprise the connection statement that is used for generating restart tokens or specifying a CAPI_CONNECTION override.
- The ADD or MOD APPL statement is required to add or modify an application name. Under the ADD or MOD APPL statement, you can define optional parameters that depend on the source type and also specify one or more ADD or MOD RSTTKN statements for specifying restart tokens for registrations associated with the application. The GENERATE parameter can be specified at either the application level or registration level.
- The END APPL statement terminates the ADD or MOD APPL statement.
- The PRINT APPL statement prints information from the CDEP file for specific applications or all applications that are involved in ODBC extractions.

You can specify a combination of ADD, MOD, and PRINT statements in a single request. The following example adds an application, specifies sequence and restart tokens for three capture registrations, generates restart tokens at the application level for the remaining capture registrations, and prints information about the application:

```
ADD  APPL IMSAPP1 IMS1 rsttkn GENERATE
   add rsttkn d002long
     sequence 00000A036E160000000000000A036BAA00000000
     RESTART  AAAAAAAA4040000000002BA700000000
   add rsttkn d002root
     SEQUENCE 00000A036E160000000000000A036BBBB0000000
     RESTART  AAAAAAAA4040000000002BA700000000
   add rsttkn d003root
   add rsttkn d008addr
   add rsttkn d008pay
   add rsttkn d008skil
     SEQUENCE 00000A036E160000000000000A036CCC00000000
     RESTART  AAAAAAAA4040000000002BA700000000
END IMSAPP1
PRINT APPL ALL
```

Connection Statement

The connection statement is comprised of the UID and EPWD parameters and optional CONN_OVR parameter.

The UID and EPWD parameters specify a user ID and encrypted password. Only PowerExchange-encrypted passwords are allowed. These parameters are required for generating the following types of restart tokens;

- Restart tokens for DB2 for i5/OS, Microsoft SQL Server, Oracle, and DB2 for Linux, UNIX, and Windows capture registrations
- Restart tokens that are used when PowerExchange security for i5/OS or z/OS is enabled by the SECURITY statement in the DBMOVER configuration file

The optional CONN_OVR parameter specifies a CAPI_CONNECTION override when the default CAPI_CONNECTION in the DBMOVER conifguration file is not appropriate for generating restart tokens.

Parameters:

UID *user_ID*

Specifies the system user identifier.

EPWD *encrypted_password*

Specifies the PowerExchange-encrypted password for the specified user ID. You can generate encrypted passwords in the PowerExchange Navigator.

CONN_OVR *capi_connection_name*

Specifies the name of the CAPI_CONNECTION statement that the DTLUAPPL utility uses when the default CAPI_CONNECTION statement is not suitable. Ensure that the override CAPI_CONNECTION statement is defined in the DBMOVER configuration file. If you do not specify this parameter, DTLUAPPL uses the default CAPI_CONNECTION statement.

ADD and MOD Statements

Use the ADD and Mode statements to generate or specify restart tokens for ODBC or PWXPC extractions. You can also use these statements to add or modify applications and the capture registrations associated with an application. For example, if you create a new capture registration, you can modify an existing application to include it.

For ODBC extractions, the utility updates application and registration information in the CDEP file based on the ADD and MOD statements.

Note: The information in the CDEP file does not apply to PWXPC extractions.

You can configure the ADD and MOD statement statements to generate restart tokens at the application level or registration level or to add the sequence and restart tokens that you specify for a capture registration in these statements to the CDEP file.

To generate restart tokens, use the RSTTKN GENERATE keyword at the application level or the GENERATE keyword at the registration level. Restart tokens generated at the registration level override those generated at the application level. Restart tokens generated at the application level apply to any capture registrations in ADD or MOD RSTTKN statements that do not include a GENERATE parameter or specific SEQUENCE and RESTART values.

If you use ODBC extractions, PowerExchange maintains restart tokens for each registration associated with an application in the CDEP file.

If you use PWXPC extractions, PWXPC maintains the restart tokens in the state table for a relational target database on the target or in the state file for a nonrelational target on the Integration Service machine. For more information about PWXPC and restart token management, see *PowerExchange Interfaces for PowerCenter*.

When you define ADD or MOD statements, use the following rules and guidelines:

- To add a new application, use the ADD APPL and ADD RSTTKN statements.
- To modify an existing application, use the MOD APPL statement.
- To add or modify an existing capture registration for an existing application, use the MOD RSTTKN statement.
- If you attempt to add an application that already exists, DTLUAPPL produces an error.
- The ADD or MOD APPL statement must always end with an END APPL statement.
- For PWXPC extractions, do not use the application name that is specified in the PWXPC application connection for generating restart tokens.

Parameters:

application_name

Specifies the name of the application to be added or modified. This value is case sensitive.

instance

Specifies the source instance. Ensure that this value matches the instance value that is displayed in the PowerExchange Navigator for the registration group and extraction group. The type of value varies by data source type.

The following table identifies the instance value type for each source type:

Data Source	Instance Value
Adabas	DBID value
DB2 for i5/OS	DBID value in CAPTPARM member of the CFG file
Datacom	MUF name
DB2 for z/OS	Subsystem ID
IDMS	CV name
IMS	IMS system identifier
Microsoft SQL Server	Database name specified for the registration group
Oracle	Collection identifier from the ORACLEID statement in the dbmover.cfg configuration file
DB2 for Linux, UNIX, and Windows	Database name specified for the registration group
VSAM	Instance name specified for the registration group

RSTTKN GENERATE

Generates restart tokens that mark the current end of the change stream at the application level. The generated restart tokens apply to the capture registrations associated with application for which restart tokens are not generated at the registration level or specifically defined in SEQUENCE and RESTART keywords.

CAPTMETH=*access_method*

Specifies one of the following capture access methods:

- CAPXRT. For real-time or continuous extraction mode.
- CAPX. For batch extraction mode.

  Valid for Oracle, Microsoft SQL Server, and DB2 for Linux, UNIX, and Windows CDC only.

CONDTYPE={P|F}

Specifies the condense type for which DTLUAPPL generates restart tokens.

Enter one of the following options:

- P. Partial condense processing by the PowerExchange Logger for Linux, UNIX, or Windows or by PowerExchange Condense on i5/OS or z/OS. Valid for all source types.
- F. Full condense processing by PowerExchange Condense on i5/OS and z/OS. Valid for Datacom, DB2 for z/OS, and VSAM sources on z/OS that specify key columns. Also valid for DB2 for i5/OS tables that have primary keys and DDS files that are defined with a unique key.

No default.

JRN=*library/journal*

Overrides the DB2 journal that is specified in the capture registration.

Valid for DB2 for i5/OS CDC only.

ORACOLL=*collection_id*

Overrides the collection identifier that is recorded for the capture registration in the CCT file. Valid for Oracle CDC only.

Valid for Oracle CDC only.

ORACONN=*source_connect_string*

Overrides the Oracle connection information for a specific Oracle collection ID, which is specified as the third positional parameter in the ORACLEID statement in the dbmover.cfg configuration file. By using this override in conjunction with ORAINST, you can use a single set of capture registrations to capture data from multiple Oracle instances.

You can specify ORACONN or ORAINST or both. If one of these parameter values is not specified, PowerExchange uses the parameter value from the ORACLEID statement in dbmover.cfg configuration file for the missing parameter.

Valid for Oracle CDC only.

ORAINST=*instance*

Overrides the Oracle instance name for a specific Oracle collection ID, which is specified as the second positional parameter in the ORACLEID statement in the dbmover.cfg configuration file. By using this override in conjunction with ORACONN, you can use a single set of capture registrations to capture data from multiple Oracle instances.

Valid for Oracle CDC only.

ORASCHEMA=*schema*

Overrides the Oracle schema name that is specified for the registration group. By using this override, you can use of a single set of capture registrations to capture data from multiple schemas in an Oracle instance.

Valid for Oracle CDC only.

UDBDB=*database*

Specifies the connection database when it is different from the registration database.

Valid for DB2 for Linux, UNIX, and Windows CDC only.

{ADD|MOD} RSTTKN *registration_name* [DB=*library*/*table*} [GENERATE]

Adds a capture registration to an application or generates restart tokens at the registration-level.

registration_name

Specifies the name of the capture registration. This value is case sensitive.

DB=*library*/*table*

Overrides the DB2 table that is specified in the capture registration.

Valid for DB2 for i5/OS CDC only.

GENERATE

Generates restart tokens that mark the current end of the change stream for the capture registration.

END APPL Statement

The END APPL statement is required to designate the end of an entire ADD or MOD APPL control statement, including the ADD or MOD RSTTKN substatements.

PRINT APPL Statement

The PRINT APPL statement is optional. It prints the restart tokens and other information that is stored in the CDEP file for one or more specific applications or for all applications that extract data by using ODBC.

You can specify the PRINT APPL statement only or include it after ADD or MOD statements that add or modify applications or generate restart tokens. The PRINT APPL statement does not require a connection statement.

To print information for multiple applications, concatenate the PRINT APPL statements or use the ALL keyword:

```
PRINT APPL application_name
PRINT APPL application_name
```

or

```
PRINT APPL ALL
```

The utility prints the following information:

- The application name
- The registrations associated with the application, including the registration user-defined names, tag names, and sequence and restart tokens
- The restart token for the first, last, and current runs of an application in the format for ODBC extractions

Running the DTLUAPPL Utility on i5/OS

To run the utility on i5/OS, use the following command:

```
SBMJOB CMD(CALL PGM(DTLLIB/DTLUAPPL)PARM('DATALIB/CFG(TKNPARMS)')) JOB(MYJOB)
JOBD(DATALIB/DTLLIST) PRTDEV(*JOBD) OUTQ(*JOBD) CURLIB(DATALIB) INLLIBL(*JOBD)
```

The CFG/TKNPARMS member in the *datalib* library contains the utility control statements.

Running the DTLUAPPL Utility on Linux, UNIX, and Windows

To run the utility on Linux, UNIX, or Windows, use the following commands:

Command	Description
dtluappl	This command assumes the utility control statements are in the dtltknp.txt file. The command displays output in the command window. **Note:** The dtltknp.txt file resides in the PowerExchange installation directory, along with the DTLUAPPL program. The file contains sample control statements for the utility, which you can customize.
dtluappl > logname.txt	This command assumes the utility control statements are in the dtltknp.txt file. The command writes output to the logname.txt file.
dtluappl *parm_file*.txt > logname.txt	This command reads the utility control statements from a user-defined parameter file. If you create the file in a directory other than the PowerExchange base installation directory, provide the full path and file name, for example, `C:\mydir\runuappl.txt`. The command writes output to the logname.txt file.

Running the DTLUAPPL Utility on z/OS

PowerExchange provides sample JCL for the DTLUAPPL utility in the DTLUAPPL member of the RUNLIB library.

Customize the sample JCL, as needed, and then submit the job.

For example, your customized JCL might contain the following statements:

```
//jobname  JOB
//LIBSRCH  JCLLIB ORDER=your.RUNLIB
//INCS1    INCLUDE MEMBER=GENBULK
//INCS3    INCLUDE MEMBER=GENCHNG
//STEP1    EXEC PGM=DTLUAPPL
//STEPLIB  DD DISP=SHR,DSN=&HLQ..LOADLIB
//         DD DISP=SHR,DSN=&HLQ..LOAD
//         DD DISP=SHR,DSN=&SCERUN
//EDMPARMS DD DISP=SHR,DSN=&HLQEDM..&LOGGER&SUFFIX..USERLIB
//SYSPRINT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//EDMMSG   DD SYSOUT=*
//***
//SYSIN DD *
   MOD APPL tokens DSN9 RSTTKN GENERATE
     ADD RSTTKN db2demo1
   END APPL tokens
   PRINT APPL tokens
//*
//*
//DTLAMCPR  DD DSN=&HLQVS..CCT,
//             DISP=(SHR)
//DTLCACDE  DD DSN=&HLQVS..CDEP,
//             DISP=(SHR)
//*
//DTLMSG    DD DSN=&HLQ..DTLMSG,
//             DISP=(SHR)
//DTLOUT    DD SYSOUT=*
```

```
//DTLCFG    DD DSN=&RUNLIB(DBMOVER),
//             DISP=(SHR)
//DTLKEY    DD DSN=&RUNLIB(LICENSE),
//             DISP=(SHR)
//DTLLOG    DD SYSOUT=*
//DTLLOG01  DD SYSOUT=*
//SYSPRINT  DD SYSOUT=*
```

The JCL statements are:

JOB

Defines the DTLUAPPL job card to MVS, including the job name.

EXEC PGM=DTLUAPPL

Executes the DTLUAPPL program. This PGM name must be DTLUAPPL.

STEPLIB DD

Points to the PowerExchange LOADLIB and LOAD libraries and the Language Environment (LE) common runtime library.

EDMPARMS DD

Points to the USERLIB library, which contains the EDMSDIR module options that are used to connect to the PowerExchange Agent and PowerExchange Logger for z/OS.

SYSPRINT DD

Defines a SYSOUT data set to which job output is printed.

SYSUDUMP DD

Defines a SYSOUT data set for dump data that can be used to diagnose DTLUAPPL problems.

EDMMSG DD

Defines a SYSOUT data set for messages from the PowerExchange Logger, ECCRs, PowerExchange Agent, Log Read API (LRAPI), and Log Write API (LWRAPI).

SYSIN DD

Defines the input control statements for the DTLUAPPL utility. You can specify the control statements in stream or point to a data set in which you define the control statements. The example JCL contains in-stream statements for adding an application name, specifying its restart tokens, and printing information for all application names.

DTLAMCPR DD

Points to the CCT data set, which contains the capture registrations.

DTLCACDE DD

Points to the CDEP data set, which contains information for the application names that are used for ODBC change data extraction processes, PowerExchange Navigator row tests, and some other PowerExchange processes.

DTLMSG DD

Points to the data set that contains the PowerExchange messages that can be issued during PowerExchange processing, including utility processing.

DTLOUT DD

Defines a SYSOUT data set that includes messages from the DTLUAPPL utility.

DTLCFG DD

Points to the DBMOVER configuration file for PowerExchange.

DTLKEY DD

Points to the PowerExchange LICENSE member in the RUNLIB library, which contains your PowerExchange license key.

DTLLOG DD

Defines a SYSOUT data set for logging PowerExchange messages that report the status and events of some PowerExchange processes and components.

DTLLOG01 DD

Defines a SYSOUT data set for logging PowerExchange messages that report the status and events of some PowerExchange processes and components when alternative logging is enabled.

DTLUAPPL Utility Examples

The following examples demonstrate how to use the DTLUAPPL utility to perform certain functions.

Example 1. Generating Restart Tokens at the Application Level

In this example, the DTLUAPPL utility generates restart tokens for the DB2DEMO1 source registration that is associated with the application name of "tokens." The utility generates restart tokens at the application level and then prints the generated restart tokens.

The example uses the following control statements:

```
UID user1 EPWD CDFB2EE51CFC16C7
ADD APPL tokens DSN7 RSTTKN GENERATE
   ADD RSTTKN db2demo1
END APPL tokens
PRINT APPL tokens
```

The GENERATE keyword is specified in the ADD APPL statement. The generated restart tokens apply to the capture registration that is specified in the ADD RSTTKN statement because no GENERATE keyword or specific SEQUENCE and RESTART values are included in the ADD RSTTKN statement.

If you use the PowerExchange Client for PowerCenter (PWXPC), after the restart tokens are printed, you can add them to the restart token file that is specified in the application connection. The token values can then be used for extraction processing. You must add eight trailing zeroes to the sequence token value for PWXPC extractions.

Example 2. Generating Restart Tokens at the Capture Registration Level

In this example, the DTLUAPPL utility generates restart tokens at the registration level for the DB2DEMO1 source registration that is associated with the application name of "tokens." The utility also prints the restart tokens.

The example uses the following control statements:

```
MOD APPL tokens DSN7
   ADD RSTTKN db2demo1 GENERATE
```

```
END APPL tokens
PRINT APPL tokens
```

The GENERATE keyword is specified in the ADD RSTTKAN statement and applies only to the capture registration that is specified in that statement.

If you use the PowerExchange Client for PowerCenter (PWXPC), after the restart tokens are generated, add them to the restart token file that is specified in the application connection for the extraction.

Example 3. Generating Restart Tokens for Continuous Extraction Mode

In this example, the DTLUAPPL utility generates restart tokens at the registration level for the rrtb001 capture registration that is associated with the existing "dummy" application and FOX920 source instance. The utility overrides the default CAPI_CONNECTION statement with the CAPX CAPI_CONNECTION named CAPXORA. The utility also prints the generated restart tokens.

The example uses the following control statements:

```
UID user01 EPWD 40ABC4B0E32FD99F CONN_OVR CAPXORA
MOD  APPL dummy FOX920 RSTTKN GENERATE CAPTMETH=CAPXRT CONDTYPE=P
   MOD RSTTKN rrtb001
END APPL dummy
PRINT APPL dummy
```

Based on these control statements, the utility generates restart tokens in the format that is required for continuous extraction mode. The utility generates restart tokens for continuous extraction of Oracle data from PowerExchange Logger for Linux, UNIX, and Windows log files because the CAPMETH parameter value is CAPXRT, the CONDTYPE parameter value is P, and the override CAPI_CONNECTION statement type is CAPX.

Example 4. Adding an Application with Restart Tokens

In this example, the DTLUAPPL utility adds the IMSAPP1 application with capture registrations d002long, d002root, d003root, and d008addr to the CDEP file.

For the d002long and d002root registrations, the ADD RSTTKN statements specify the sequence and restart tokens. For the d008addr capture registration, the utility generates the restart tokens. For the d003root capture registration, the utility adds the registration to the CDEP file without restart tokens.

The example uses the following control statements:

```
ADD  APPL IMSAPP1 IMS1
   ADD RSTTKN d002long
     SEQUENCE 00000A036E160000000000000A036BAA00000000
     RESTART  AAAAAAAA4040000000002BA70000000
   ADD RSTTKN d002root
     SEQUENCE 00000A036E160000000000000A036BBBB0000000
     RESTART  AAAAAAAA4040000000002BA700000000
   ADD RSTTKN d003root
   ADD RSTTKN d008addr GENERATE
```

For ODBC extractions, the utility stores the application, associated capture registrations, and restart tokens in the CDEP file. If you use PWXPC instead of ODBC, you must manually add the capture registrations and their restart tokens to the restart token file that is specified in the application connection for the extractions.

Example 5. Adding an Application and Generating Restart Tokens on a Remote Instance

In this example, the DTLUAPPL utility adds the ORAAPP3 application for the remote ORAINST1 instance. The utility also generates restart tokens at the application level, which will be used for the oraemp2 source registration.

The example uses the following control statements:

```
ADD APPL ORAAPP3 ORAINST1 RSTTKN GENERATE ORACONN=OCONN ORAINST=OINST ORACOLL=OCOLL
   ADD RSTTKN oraemp2
END APPL ORAAPP3
```

Example 6. Modifying Restart Tokens in an Application

In this example, the DTLUAPPL utility modifies the restart tokens that are used for the d002long capture registration in the IMSAPP1 application. The sequence and restart tokens are specified in the MOD RSTTKN statement and apply to the registration only.

The example uses the following control statements:

```
MOD  APPL IMSAPP1 IMS1
   MOD RSTTKN d002long
     SEQUENCE 000000032D450000000000000032D4500000000
     RESTART  C4D6C3D34040000000032CBD00000000
END APPL IMSAPP1
```

Example 7. Modifying an Application and Adding a Registration

In this example, the DTLUAPPL utility adds the d003long capture registration with specific sequence and restart token values to the existing IMSAPP1 application.

The example uses the following control statements:

```
MOD  APPL IMSAPP1 IMS1
   ADD RSTTKN d003long
     SEQUENCE 000000032D450000000000000032D4500000000
     RESTART  C4D6C3D34040000000032CBD00000000
END APPL IMSAPP1
```

The ADD RSTTKN statement adds the capture registration and the specified SEQUENCE and RESTART tokens to the CDEP file, which is used for ODBC extractions. If you use PWXPC to extract change data, you must manually add the capture registration and sequence and restart tokens to the restart token file that is specified in the application connection for the extraction.

Example 8. Printing Information for an Application

In this example, the DTLUAPPL utility prints information from the CDEP file for a specific application.

The example uses the following control statement:

```
PRINT APPL {application_name}
```

Note: You can print information for more than one application by concatenating multiple PRINT APPL statements, each with a specific application name, or by using the ALL keyword for all applications.

The utility produces the following example output if no extractions have run for the application:

```
Application name=<DB2APPL5> Rsttkn=<2>  Ainseq=<0> Preconfig=<N>
            FirstTkn  =<>
            LastTkn   =<>
            CurrentTkn=<>
     Registration name=<db2v52c.1> tag=<DB2DSN1db2v52c1>
```

```
                Sequence=<000000035D50000000000000000035D5000000000>
                Restart =<C4D6C3D34040000000035CC800000000>
        Registration name=<db2tst5c.1> tag=<DB2DSN1db2tst5c1>
                Sequence=<000000035D50000000000000000035D5000000000>
                Restart =<C4D6C3D34040000000035CC800000000>
```

After an extraction has run for the application, the utility produces the following output, which includes the FirstTkn and LastTkn values:

```
    Application name=<DB2APPL1> Rsttkn=<1>  Ainseq=<0> Preconfig=<N>
                FirstTkn  =<C4D6C3D340400000000335D000000000>
                LastTkn   =<C4D6C3D3404000000000003453E00000000>
                CurrentTkn=<>
         Registration name=<db2v52c.1> tag=<DB2DSN1db2v52c1>
                Sequence=<00000003191400000000000000003191400000000>
                Restart=<4D6C3D3404000000000003188C00000000>
```

The following table describes the fields in the PRINT APPL output:

Field	Description
Rsttkn	Number of RSTTKN pairs that are recorded for the application.
Ainseq	For internal use only.
Preconfig	Not used.
FirstTkn	The restart token for first successful run of the application when using ODBC.
LastTkn	The restart token for last successful run of the application when using ODBC.
CurrentTkn	The restart token for current active or last failed run of the application when using ODBC.

Note: If you use ODBC, you can also view the restart tokens that are in the printed output in the **Extract Application** dialog box of the PowerExchange Navigator.

CHAPTER 8

DTLUCBRG - Batch Registration Utility

This chapter includes the following topics:

DTLUCBRG Utility Overview

Many customers who use change data capture in production environments need to register hundreds of tables for capture. Using the PowerExchange Navigator to create and manage large numbers of registrations is not practical. The DTLUCBRG utility enables you to create and manage registrations in bulk.

This utility can perform the following tasks:

- Adds capture registrations and extraction maps. The utility creates registrations and extraction maps at specified PowerExchange Listener locations for a set of tables or data maps. You can use a mask to limit the registrations created.
- Modifies inactive or active registrations.
- Performs a test run to report on the scope of the registrations before actually creating them.
- For Microsoft SQL Server sources, changes the status of multiple registrations in one operation and also deletes and rebuilds the associated SQL Server publications. This feature is useful if you need to reset the status of mulitple registrations, or if you need to re-create registrations and regenerate publications because you made DDL changes to a large number of tables.

Note: DTLUCBRG creates all registrations with a version of 1. The utility cannot set the registration status to history and then create a subsequent version of the registration.

Supported Operating Systems for the DTLUCBRG Utility

DTLUCBRG is available on the following operating systems:

- i5/OS
- Linux
- UNIX
- Windows
- z/OS

You can create registrations on other platforms by using PowerExchange Listeners.

DTLUCBRG Utility Parameters

This section describes DTLUCBRG parameters.

The parameters are supplied in the following locations:

- On i5/OS, parameters are defined in a file that you specify on the command line.
- On Linux, UNIX, and Windows, parameters are defined in the dtlucbrg.txt file. The directory from which you run DTLUCBRG should include this file.
- On z/OS, parameters are defined in the SYSIN of the JCL.

The following table describes DTLUCBRG parameters:

Parameter	Default	Description
CONDTYPE	-	Specifies the condense option to use for the capture registrations. Options are: - **FULL**. This option is available if you use PowerExchange Condense on i5/OS or z/OS. PowerExchange accumulates change data in keyed condense files. Because later changes supersede earlier changes, this condense option does not maintain transactional consistency. Also, the following limitations apply: 1) Adabas and IDMS log-based CDC data sources are not supported. 2) On z/OS, data sources must have key columns. The total length of all key columns for a source cannot exceed 250 bytes. 3) On i5/OS, source tables must have primary keys, or DDS files must be defined with a unique key. - **PART**. This option is available if you use PowerExchange Condense on i5/OS or z/OS or the PowerExchange Logger for Linux, UNIX, or Windows. Changes in successfully committed UOWs are written to condense files or PowerExchange Logger log files in chronological order based on UOW end time. PowerExchange writes all changes for the columns of interest, not just the latest changes. This condense type maintains transactional consistency. - **NONE**. Capture registrations are not eligible for full or partial condense processing. For more information, see the *PowerExchange Navigator User Guide* or PowerExchange CDC guides.
CRGNAME	-	Specifies a capture registration name. This value can be up to 13 alphanumeric characters in length and cannot begin with a number. PowerExchange uses this value as the entire registration name. Because PowerExchange does not append a unique number to it, as it does with CRGPREFIX, you can use CRGNAME to replace registrations that were lost or damaged or to regenerate registrations under their original registration names. Do not use CRGNAME under any of the following circumstances: - The TABLE parameter specifies a mask that contains the asterisk (*) wildcard. To use CRGNAME, the table name must be explicitly specified in the same manner as when registering the table. - The REUSECRGNAME parameter is set to Y. - The CRGPREFIX parameter is specified. You must specify CRGPREFIX or CRGNAME, but do not specify both of them.
CRGPREFIX	-	Specify a prefix of one to four characters in length. PowerExchange appends a 4-digit sequential number to this value to form each registration name. Because a sequential number is appended, each registration name is unique. To replace a registration under its previous name, you must use the CRGNAME parameter, rather than the CRGPREFIX parameter. The registration name can have any of the following formats: *xnnnn* *xxnnnn* *xxxnnnn* *xxxxnnnn* Where: - *x*. The value assigned by CRGPREFIX. Allowable characters for the first *x* are a to z. Allowable subsequent characters are a to z and 0 to 9. - *nnnn*. A sequential number starting from 0001. If the table name contains characters that are not allowed, an error message is generated informing the user that the table will be ignored but that processing continues. No registration is generated for the table name shown in the message. You must specify CRGPREFIX or CRGNAME, but do not specify both of them.

Parameter	Default	Description
DBTYPE	-	Specifies the three-character mnemonic for the data source type: - **ADA**. Adabas. - **AS4**. DB2 for i5/OS. - **DB2**. DB2 for z/OS. - **DCM**. Datacom. - **IDL**. IDMS log-based. - **IMS**. IMS. - **MSS**. Microsoft SQL Server. - **ORA**. Oracle. - **UDB**. DB2 for Linux, UNIX, and Windows. - **VSM**. VSAM. **Note:** Use DB2 only for DB2 on z/OS. Use AS4 or UDB for DB2 on other platforms.
EPWD	-	Specifies an encrypted password for the specified user ID. If the utility accesses an i5/Os or z/OS location, you can enter an encrypted PowerExchange passphrase instead of an encrypted password. Do not encrypt a passphrase that contains invalid characters, such as double-quotation marks, single quotation marks, or currency symbols. You can create an encrypted password or passphrase in the PowerExchange Navigator by selecting **File** > **Encrypt Password**. Use EPWD instead of PWD if you are not allowed to store passwords in a readable format.

Parameter	Default	Description
INSTANCE	-	Specifies the source instance for your registrations. The value type depends on the data source type that you specify in the DBTYPE parameter. Based on the DBTYPE option, enter one of the following values: - For **ADA**, enter an Adabas nucleus name. - For **AS4**, enter a DB2 for i5/OS instance value that matches the INST parameter value in the AS4J CAPI CONNECTION statement in the DBMOVER member of the *dtllib*/CFG file. If you use PowerExchange Condense, this instance value must also match the DBID parameter value in the CAPTPARM member. - For **DB2**, enter a DB2 subsystem ID (SSID). - For **DCM**, enter a Datacom Multi-User Facility (MUF) name. - For **IDL**, enter an IDMS log-based CDC instance value that matches the *registration_logsid* parameter in the LOGSID statement in the DBMOVER configuration member. - For **IMS**, enter an IMS subsystem ID that matches the *ims_ssid* parameter value in the IMSID statement in the DBMOVER configuration member. - For **MSS**, optionally enter a unique user-defined instance identifier for the SQL Server database server and database name combination defined in the MSSOPTS DBSERVER and DBNAME parameters. Maximum length is seven characters. This instance identifier is incorporated into the names of the extraction maps that the utility creates. If you use the PowerExchange Logger for Linux, UNIX, and Windows, ensure that the instance identifier matches the DBID parameter value in the Logger configuration file. If you do not enter this instance value, PowerExchange generates a unique instance identifier that is composed of all or part of the publication database name followed by a 3-digit number if a number is required to make the identifier unique. This INSTANCE parameter is useful in migration scenarios. If you need to deploy change capture from one environment to another, such as from test to production, and you do *not* define an instance identifier, PowerExchange uses the generated instance identifier in the new environment. The generated instance identifier might be different than the one in the original source environment. To avoid having to update the extraction map names in PowerCenter workflows and edit the DBID parameter value for the PowerExchange Logger, enter an instance identifier in the INSTANCE parameter that matches the instance identifier in the original environment when creating registrations for the new environment. **Tip:** In this migration scenario, ensure that the dbmover.cfg configuration files in the original environment and new environment specify unique paths in CAP_PATH and CAPT_XTRA statements. - For **ORA**, enter the user-defined collection ID for the Oracle instance that matches the *collection_id* parameter in the ORACLEID statement in the PowerExchange DBMOVER configuration member. - For **UDB**, enter DB2 for Linux, UNIX, and Windows database name. - For **VSM**, enter a VSAM collection identifier.

Parameter	Default	Description
LOCATION	-	Required. Specifies a node name that points to the location of the PowerExchange Listener that manages the capture registrations and extraction maps. If the registrations, data maps, and data source reside on the same system, you can specify LOCATION=LOCAL. In this case, do not define the other LOCATION_*xxx* parameters. **Warning:** Do not specify LOCATION=LOCAL if you add or modify VSAM, IMS synchronous, or DB2 for z/OS capture registrations. Otherwise, the PowerExchange Agent fails to detect the new or updated registrations, and the ECCR cannot retrieve the registration changes during a restart or refresh operation. Instead, specify the LOCATION value for the PowerExchange Listener that manages the registrations and is connected to the PowerExchange Agent. Then, the updated registration information is available for ECCR processing.
LOCATION_CRG	Value of LOCATION	Specifies the location of the registration file (CCT).
LOCATION_DM	Value of LOCATION	Specifies the location of the DATAMAP file.
LOCATION_XDM	Value of LOCATION	Specifies the location of the extraction maps.
NOTIFYCHANGES	Y for supported data sources	Available for DB2 and Oracle sources. If NOTIFYCHANGES=Y, any change to the schema for the table causes PowerExchange CDC to fail and log an error message. For a DB2 for z/OS source, the DB2 ECCR abnormally ends after it reads the first change record for the table after the schema change. For an Oracle source, Oracle CDC fails and logs an error message in the following situations: - If a change record for a table that you registered for capture contains a column that you did not register for capture - If a change record does not contain a column that you registered for capture For Oracle CDC, if a definition for a table changes in a way that is compatible with the PowerExchange capture registration, Oracle CDC continues to capture changes for that table. For example, if the length of a character column decreases but the capture registration does not reflect this change, Oracle CDC continues to capture changes for the table. Conversely, if the datatype of a column changes from numeric to character without a change in the capture registration, Oracle CDC continues to capture changes for the table until it encounters the first change record that contains nonnumeric data for the column. When Oracle CDC encounters a change record containing nonnumeric data for the column, it fails and logs an error message. For data sources other than DB2 and Oracle, this parameter is ignored and defaults to N.
OUTPUT	On z/OS SYSPRINT, on Win STDOUT	Specifies the location and file name of the report from DTLUCBRG. On Windows, the format is: `OUTPUT=c:\pwx\outfile.txt` If the path includes names with spaces, enclose the path in quotes. On z/OS, the report is directed to the SYSPRINT DD output.

Parameter	Default	Description
PWD	-	Specifies a clear text password for the specified user ID. For access to an i5/OS or z/OS location, you can enter a valid PowerExchange passphrase instead of a password. An i5/OS passphrase can be from 9 to 31 characters in length. A z/OS passphrase can be from 9 to 128 characters in length. A passphrase can contain the following characters: - Uppercase and lowercase letters - The numbers 0 to 9 - Spaces - The following special characters: `' - ; # \ , . / ! % & * ( ) _ + { } : @ | < > ?` **Note:** The first character is an apostrophe. Passphrases cannot include single quotation marks (‘), double quotation marks (“), or currency symbols. If a passphrase contains spaces, you must enclose it with double-quotation marks ("), for example, `"This is a passphrase"`. If a passphrase contains special characters, you must enclose it with triple double-quotation characters ("""), for example, `"""This passphrase contains special characters ! % & *."""`. If a passphrase contains only alphanumeric characters without spaces, you can enter it without delimiters. To use passphrases, ensure that the PowerExchange Listener runs with a security setting of SECURITY=(1,N) or higher in the DBMOVER member. For more information, see "SECURITY Statement" in the *PowerExchange Reference Manual*. **Note:** On z/OS, a valid RACF passphrase can be up to 100 characters in length. PowerExchange truncates passphrases longer than 100 characters when passing them to RACF for validation. Do not also specify EPWD.
REPLACE	N	Indicates whether to replace existing inactive registrations. Options are: - **Y**. Replace existing inactive registrations that match the mask specified in the TABLE parameter. - **N**. Do not replace any existing registrations. Add registrations for tables that match the mask and do not yet exist.
REPLACEACTIVE	N	Indicates whether to replace existing active registrations. Options are: - **Y**. Replace existing active registrations that match the mask specified in the TABLE parameter. For this replacement to occur, the REPLACE parameter must also be set to Y. - **N**. Do not replace active registrations.
REUSECRGNAME	N	Specifies one of the following options: - **Y**. For existing registrations, retain the current name. - **N**. Rename existing registrations using the CRGPREFIX and sequential number format.
RPTCOLS	Y	Specifies one of the following options: - **N**. Report on only the table names that were registered during the run. - **Y**. Report on table names and columns that were registered during the run.

Parameter	Default	Description
STATUS	-	Specifies one of the following options: - **A**. Create registrations in an active state. - **I**. Create registrations in an inactive state. You will need to activate the registrations to make them eligible for change capture. For Microsoft SQL Server sources, if you specify UPDATESTATUS=Y in the MSSOPTS parameter, this STATUS parameter resets the status of all registrations that match the filter criteria you specify.
TABLE	-	Specifies a mask that restricts the source tables for which the DTLUCBRG utility will create capture registrations. - For relational database tables, the mask can be specified in the following format: *OWNER.TABLE* - For nonrelational sources, the mask includes the data map name and can be specified in the following format: *SCHEMA.MAPNAME* Optionally, you can define a two-tier parameter value that includes a table name to narrow the selection of tables when multiple tables are defined in a data map but not all of the tables need to be registered. In this case, use the following format: *SCHEMA.MAPNAME_TABLENAME* For IMS sources, a three-tier parameter value is also allowed: *SCHEMA.MAPNAME.TABLENAME* Use an asterisk (*) wildcard to represent one or more characters in any part of the parameter value. For example, the following mask matches all relational tables that have the owner "OWNAB" and a table name that starts with "T": `OWNAB.T*`
TESTRUN	Y	Specifies one of the following options: - **Y**. Run the utility and report on the registrations to be updated or added. No registrations will be affected by this run. - **N**. Run the utility and add/update registrations.
UID	-	Specifies a user ID that allows access to the source. The requirement for this parameter depends on both the data source that is being registered and the value of the SECURITY statement in the PowerExchange DBMOVER configuration file. For a source on a supported Linux, UNIX, or Windows system, if you have enabled PowerExchange LDAP user authentication and, if applicable, disabled relational pass-through authentication, the user ID is the enterprise user name. For more information, see the *PowerExchange Reference Manual*. If you specify a user ID, also specify either a PWD or EPWD value, but not both.

Note: You are not required to specify parameters that have default values.

Specifying Multiple Sets of Parameters on the DTLUCBRG Utility

Multiple sets of parameters can be placed in the same parameters file. These sets must be separated with a ';' positioned on a new line between the sets of parameters. For example, on Linux, UNIX, or Windows, you could include the following lines:

```
DBTYPE DB2
TABLE DTL*
OUTPUT=c:\dtlucdb2.txt
etc. ...
;
```

```
DBTYPE DB2
TABLE PWX*
OUTPUT=c:\dtlucdb2_1.txt
etc. ...
```

Note: To see output from each set of parameters on Linux, UNIX, or Windows, define a different file for OUTPUT=. On z/OS, you cannot specify multiple output files. Each set of parameters is appended to the SYSPRINT DD output.

DTLUCBRG Utility Source-Specific Parameters

For Adabas, IMS, Microsoft SQL Server, and Oracle data sources, the DTLCUBRG utility requires additional information to update registrations. You provide this information in the source-specific parameters.

These parameters have names that begin with the DBTYPE value and end with a suffix of OPTS. Each one has one or more subparameters.

The following table describes the source-specific parameters:

Parameter	Subparameter	Default	Required?	Description
ADAOPTS	FileNo	None	Yes	Adabas file number.
ADAOPTS	DBID	None	Yes	Adabas DBID.
IMSOPTS	TYPE	SYN	No	Determines whether capture processing is for an IMS synchronous or log-based CDC environment. Valid values are SYN or LOG.
IMSOPTS	DBDNAME	From Datamap	No	Database name from the DBD.
IMSOPTS	IMSID	None	No	IMS subsystem ID that matches value in the IMSID statement in the DBMOVER member of RUNLIB.
IMSOPTS	PRIMDSN	None	No	Primary data set name.
MSSOPTS	DBSERVER	None	Yes	Name of the database server.
MSSOPTS	DBNAME	From Datamap	Yes	Name of the database that contains the tables from which changes are captured.

Parameter	Subparameter	Default	Required?	Description
MSSOPTS	UPDATESTATUS	N	No	For Microsoft SQL Server sources, indicates whether the utility can change the status of multiple registrations in one operation and delete and rebuild the associated publications. Use this parameter when you need to switch the status of many registrations at one time, or when you need to make DDL changes to many source tables and do not want to manually re-create the registrations and regenerate the publications. If you specify UPDATESTATUS=Y, the utility performs the following actions, depending on the STATUS setting: - If STATUS is set to I, the active registrations that you select are set to inactive and the associated publications are deleted. - If STATUS is set to A, the inactive registrations that you select are set to active and the associated publications are automatically rebuilt based on the existing registrations. You can filter the registrations for the utility to process by specifying the CONDTYPE, TABLE, and CRGNAME or CRGPREFIX parameters.
ORAOPTS	DDLFILE	dtlucbrg_ora.sql	No	File name of the file that stores Alter DDL for supplemental log groups. This parameter can include a full path and file name, such as `c:\sql\oraopts.sql`. If the value includes spaces, do not use quotes to delimit the path and file name. If you define DDLFILE=, the utility uses the default file name and default directory. To use a different file name, specify the full path and file name. If you specify only a path, the utility returns an error.

Related Topics:

- "DTLUCBRG Utility Parameters" on page 107

ADAOPTS Parameter - Adabas

The syntax of the ADAOPTS parameter is:

```
ADAOPTS=(FileNo=<file number>,DBID=<dbid>)
```

If ADAOPTS is specified for any DBTYPE other than Adabas an error message will result.

IMSOPTS Parameter - IMS

The syntax of the IMSOPTS parameter is:

```
IMSOPTS=(TYPE=<type>,DBDNAME=<name>,IMSID=<name>, PRIMDSN=<dsname>)
```

If IMSOPTS is specified for any DBTYPE other than IMS an error message will result.

MSSOPTS Parameter - Microsoft SQL Server

Use the following syntax for the MSSOPTS parameter:

```
MSSOPTS=(DBSERVER=server_name,DBNAME=database_name,[UPDATESTATUS={Y|N}])
```

If MSSOPTS is specified for a DBTYPE other than MSS, the DTLUCBRG utility issues an error message.

ORAOPTS Parameter - Oracle

The syntax of the ORAOPTS parameter is:

```
ORAOPTS=(DDLFILE=<filename>)
```

DDLFILE is a mandatory sub-parameter of ORAOPTS. To access the default path and file name code:

```
ORAOPTS=(DDLFILE=)
```

The DDL created by the run must be run manually to create the supplemental log groups required for PowerExchange Oracle capture.

Related Topics:

- "DTLUCBRG Utility Source-Specific Parameters" on page 114

Example Input for the DTLUCBRG Utility

The following example input registers all DB2 tables on the DSN1 subsystem that have an owner name beginning with the characters "DTL":

```
DBTYPE DB2
TABLE DTL*
CONDTYPE NONE
INSTANCE DSN1
LOCATION MP3000
LOCATION_CRG MP3000
LOCATION_DM MP3000
LOCATION_XDM MP3000
CRGPREFIX DB2
TESTRUN N
STATUS A
UID dtlusr
PWD dtlusr
OUTPUT=c:\dtlucdb2.txt
REPLACE Y
REPLACEACTIVE Y
RPTCOLS N
```

The TABLE parameter works with the REPLACE and REPLACEACTIVE parameters to indicate that any active or inactive registrations that match the TABLE mask will be replaced.

The CONDTYPE setting of NONE indicates that data will not be available for PowerExchange Condense or PowerExchange Logger for Linux, UNIX, and Windows processing. For information about PowerExchange Condense or the PowerExchange Logger, see the PowerExchange CDC guide for your operating system.

The INSTANCE parameter specifies the DB2 subsystem name.

The LOCATION and LOCATION_*xxx* parameters indicate that the MP3000 system will contain the target data and the registration, data map, and extraction map files. Verify that each LOCATION parameter value maps to a NODE statement in the DBMOVER configuration file.

The STATUS and CRGPREFIX parameters indicate that the registrations will be created with a status of Active and the prefix "DB2."

The RPTCOLS parameter setting of NO indicates that the DTLUCBRG report output will show table names only, without column information. The OUTPUT parameter indicates that the report output will be written to the dtlucdb2.txt file.

Related Topics:

- "DTLUCBRG Utility with RPTCOLS=N Report Description" on page 119
- "DTLUCBRG Utility with RPTCOLS=Y Report Description" on page 120

DTLUCBRG Code Page Processing

DTLUCBRG disregards the CODEPAGE statement in the DBMOVER configuration file and instead uses the code page that is used to store registrations and capture data map metadata, as follows:

- IBM037 on i5/OS
- IBM1047 on z/OS
- UTF8 on Linux, UNIX, or Windows

Whenever the DBMOVER CODEPAGE statement is overridden, PowerExchange issues the following message:

```
Changed client code pages to name (internal_code_page_number)
```

On i5/OS and z/OS, PowerExchange supports table and column names with characters that are present in code pages IBM1037 and IBM1047, respectively. Accented characters that are not in the code pages are not supported. On Linux, UNIX, and Windows, all characters are supported.

When the PowerExchange Llistener retrieves data, code page conversion of SQL and data is performed automatically. For example, a PowerExchange Listener on z/OS might need to use a particular SQL code page to meet the requirements of a DB2 subsystem. SQL is converted to the required code page and is received by the Listener ready for use by DB2. Column data is described according to the DB2 CCSIDs and sent back to DTLUCBRG, which converts it to the required metadata code page.

Code Page Processing for DB2 Registrations in Local Mode on z/OS

Because a PowerExchange Listener is not involved, code page processing for DB2 registrations in local mode on z/OS does not trigger automatic SQL code page conversion. In this case, DTLUCBRG performs the required code page conversion and issues the following message:

```
Using codepage code_page_name (code_page_number for table names in DB2 subsystem
subsystem.
```

Note that because DTLUCBRG uses connection pooling when using a PowerExchange Listener, performance is only slightly slower than when running locally.

Running the DTLUCBRG Utility

It is strongly advised that the utility is run with TESTRUN=Y initially to assess the scope of the changes and additions to registration resulting from a particular run. After you see the changes reported by the TESTRUN=Y execution, change TESTRUN to N and run to see the changes take effect.

Running the DTLUCBRG Utility on i5/OS

On i5/OS, run the utility by entering the following command:

```
call dtlucbrg parm('cs=filepath1/filepath2(myparmfile)')
```

Where *myparmfile* contains the DTLUCBRG control statements.

Running the DTLUCBRG Utility on Linux, UNIX, and Windows

The input parameters should be supplied in the dtlucbrg.txt file by default. If the parameters are coded in this file, run the utility by entering DTLUCBRG on the command line. Parameters may be supplied in a file of another name. The report will be written to the location specified in the OUTPUT parameter.

DTLUCBRG Utility on Linux and UNIX Syntax

On Linux and UNIX, run the utility by entering DTLUCBRG on the command line as follows:

```
dtlucbrg CS=/MyParms/PWX/ucbrgtest.txt
```

DTLUCBRG Utility on Windows Syntax

On Windows, to run with a specified file path and name, use the following syntax:

```
c:\>dtlucbrg CS=C:\MyParms\PWX\ucbrgtest.txt
```

If the path or file name contains embedded blanks, use the following syntax:

```
c:\>dtlucbrg CS="C:\MyParms\PWX\In Quotes for Embedded Blanks.txt"
```

Running the DTLUCBRG Utility on z/OS

The following JCL provides example statements to use when you run this utility on z/OS.

```
//DTLUSRRG JOB  'DTLSETFL',MSGCLASS=X,NOTIFY=&SYSUID,
//         CLASS=A,REGION=64M
//*********************************************************************
//*                                                                   *
//* RUN BATCH REGISTRATION UTILITY                                    *
//*                                                                   *
//*********************************************************************
//INCS1 INCLUDE MEMBER=GENBULK
//***
//RUN      EXEC PGM=DTLUCBRG
/*
/*
//STEPLIB  DD DISP=SHR,DSN=&HLQ..LOADLIB
//         DD DISP=SHR,DSN=&HLQ..LOAD
//         DD DISP=SHR,DSN=&SCERUN
//         DD DISP=SHR,DSN=&DB2LOAD
//         DD DISP=SHR,DSN=&DB2EXIT
//*
//SYSPRINT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//***
//SYSIN DD *
DBTYPE DB2
TABLE DTLUSR.DTL*
CONDTYPE NONE
INSTANCE DSN1
LOCATION node1
LOCATION_CRG node1
LOCATION_DM node1
LOCATION_XDM node1
CRGPREFIX DB2
```

```
TESTRUN N
STATUS A
UID <logonid>
PWD xxxxxx
REPLACE Y
REPLACEACTIVE Y
RPTCOLS N
/*
//* - other parms
//* EPWD
//* REUSECRGNAME
//*
//* CDC Datasets - need to be open if CDC to be used
//*
//DTLMSG    DD DISP=SHR,DSN=&HLQ..DTLMSG
//DTLCFG    DD DISP=SHR,DSN=&RUNLIB(DBMOVER)
//DTLKEY    DD DISP=SHR,DSN=&RUNLIB(LICENSE)
//DTLSGN    DD DISP=SHR,DSN=&RUNLIB(SIGNON)
//DTLLOG    DD SYSOUT=*
//SYSUDUMP  DD SYSOUT=*
//SYSOUT    DD SYSOUT=*
//SYSPRINT  DD SYSOUT=*
//CEEDUMP   DD SYSOUT=*
```

DTLUCBRG Utility with RPTCOLS=N Report Description

The following example report is at the table level with no column detail (RPTCOLS=N). The header contains the value of TESTRUN and this should be checked to ensure the utility has been run in the expected mode. After the header, the values of the input parameter file are echoed in the report.

```
2005-01-24 11:07:16 DTLUCBRG REGISTRATION REPORT (TESTRUN=N)
CONDTYPE     = <None>
CRGPREFIX    = <DB2>
DBTYPE       = <DB2>
INSTANCE     = <DSN1>
LOCATION     = <MP3000>
LOCATION_CRG = <MP3000>
LOCATION_DM  = <MP3000>
LOCATION_XDM = <DB3000>
OUTPUT       = <c:\dtlucdb2.txt>
REPLACE      = <Y>
REPLACEACTIVE= <Y>
REUSECRGNAME = <N>
RPTCOLS      = <N>
STATUS       = <A>
TABLE        = <DTL*>
IMSOPTS: Not relevant for this run
ORAOPTS: Not relevant for this run
MSSOPTS: Not relevant for this run
ADAOPTS: Not relevant for this run
RegName  Old      Table-name                          Old   Old    Vers
         RegName                                      Cond  Status >1
=========================================================================
db20008  db2captc DTLUSR.DTLRESTART                   Part  A      -
db20009  db20001  DTLUSR.DTLSTATUS                    None  A      -
db20010  db20002  DTLUSR.DTLTST4                      None  A      -
db20011  db20003  DTLUSR.DTLTST5                      None  A      -
db20012  db20004  DTLUSR.DTLTST6                      None  A      -
db20013  db20005  DTLUSR.DTLTST8                      None  A      -
db20014  db20006  DTLUSR.DTLTST9                      None  A      -
=========================================================================
Summary of registrations created with  status ACTIVE and
condense type NONE
No of registrations created = 0
No of registrations updated = 7
No of existing registrations not matching update parameters: = 0
2005-01-24 11:08:13 END OF DTLUCBRG REGISTRATION REPORT
```

The following table describes the content in the sample report:

Report Field	Description
Registration Name	The name of the new registration.
Old Registration Name	The name of the old registration name where these have been replaced by new names (determined by setting REUSECRGNAME=N).
Table Name	The table that is being registered for capture.
Old Condense Type	Where a registration is being replaced the old condense option value.
Old Status	Where a registration is being replaced the old status value.

DTLUCBRG Utility with RPTCOLS=Y Report Description

The following example shows the extra information generated when you run the DTLUCBRG utility with RPTCOLS set to Y:

```
------------------------------------------------------------------------
db20030  db20023  DTLUSR.DTLSTATUS                    None  A      -
-Column Name ----------------Type-----Precision--Scale--Nulls-Key--------
 TABLE_NAME                 VARCHAR       255     0   N     Y
 STATUS                     CHAR           20     0   N     N
 STATUS_REASON              CHAR           20     0   N     N
 APPLY_SEQUENCE             VARCHAR       255     0   Y     N
 RESTART_POINT              VARCHAR       255     0   Y     N
------------------------------------------------------------------------
```

The following table describes the fields for the extended report format:

Field	Description
Column Name	Column name
Type	Type, such as CHAR, VARCHAR, and so on
Precision	Length of column
Scale	Decimal places
Nulls	Nulls, Y/N
Key	Key column, Y/N

Column information is displayed immediately after the relevant table registration information.

Note: Table and column names might be truncated in the report.

DTLUCBRG Utility Usage Notes

Review the following considerations:

- When you run the DTLUCBRG utility, you might receive error messages that imply the registrations that are being created might not be usable. For example:

  ```
  PWX-09702  Oracle ID xxxx not found in configuration
  ```

  To avoid the creation of unusable registrations, perform the following actions:

  - Correct the error that the message reports and then re-run the utility. Repeat this process until the utility generates no error messages. It is particularly important to eliminate errors that result from the lookup of existing registrations, because these errors might result in unusable registrations even if no entries are found.
  - First attempt to create a small volume of registrations.
  - Include the following parameter in the utility input to perform a test run of the utility without creating the registrations:

    ```
    TESTRUN Y
    ```

    After you perform a test run without errors for a small volume of registrations, try the test run again for the complete set of registrations that you want to create. After this run completes without errors, specify TESTRUN N in the utility input and run the utility again to create the registrations. Default is TESTRUN Y.

- If you use the DTLUCBRG utility to create capture registrations for data sources on z/OS but the utility does not find the required source data maps in the DATAMAPS member on the LOCATION_DM node, the utility does not create the registrations and does not report the error. Instead, the utility appears to end successfully with RC=0. To avoid this problem, ensure that the data maps for the source objects exist in the DATAMAPS member before you run the utility.

CHAPTER 9

DTLUCDEP - CDEP Maintenance Utility

This chapter includes the following topics:

DTLUCDEP Utility Overview

When you run PowerExchange change capture processes, you might need to delete obsolete or unnecessary applications and extractions from your PowerExchange Capture Extraction Process Control (CDEP) file.

Use the DTLUCDEP utility to modify or print the contents of the CDEP file. This file contains information about the change capture extraction processes that have run, timings, and input. The CDEP file is written to or read by the extraction process to establish the starting point for an extraction.

Warning: It is extremely important that this utility is used appropriately as any modifications performed on the CDEP file are irreversible. This could mean that starting points for your change capture processes may be lost.

It is suggested that a backup copy of the CDEP file is taken before running the DTLUCDEP utility.

Supported Operating Systems for the DTLUCDEP Utility

The DTLUCDEP utility can run on the following operating systems:

- i5/OS
- UNIX and Linux
- Windows
- z/OS

Control Statement Syntax for the DTLUCDEP Utility

Use the following syntax for the DTLUCDEP utility control statements:

```
[USER user_ID {pwd password|EPWD epassword}]
{PRINT|MODIFY} APPL {appname|ALL} days
```

Control Statement Parameters for the DTLUCDEP Utility

Use the DTLUCDEP definition file to control whether the DTLUCDEP utility prints or modifies CDEP information for an application. You can filter the resulting utility output based on a number of days.

The utility has the following parameters:

USER *user_ID*

If security checking is enabled, an operating system user ID.

For an application on a supported Linux, UNIX, or Windows system, if you have enabled PowerExchange LDAP user authentication, the user ID is the enterprise user name. For more information, see the *PowerExchange Reference Manual*.

{PWD *password*|EPWD *encrypted_password*}

A password or encrypted password for the specified user.

- **PWD**. A password for the specified user.
  For access to i5/OS or z/OS, you can enter a valid PowerExchange passphrase instead of a password. An i5/OS passphrase can be from 9 to 31 characters in length. A z/OS passphrase can be from 9 to 128 characters in length. A passphrase can contain the following characters:
  - Uppercase and lowercase letters
  - The numbers 0 to 9
  - Spaces
  - The following special characters:

    ```
    ' - ; # \ , . / ! % & * ( ) _ + { } : @ | < > ?
    ```

Note: The first character is an apostrophe.

Passphrases cannot include single quotation marks (‘), double quotation marks (“), or currency symbols.

If a passphrase contains spaces, you must enclose it with double-quotation marks ("), for example, `"This is a passphrase"`. If a passphrase contains special characters, you must enclose it with triple double-quotation characters ("""), for example, `"""This passphrase contains special characters ! % & *."""`. If a passphrase contains only alphanumeric characters without spaces, you can enter it without delimiters.

To use passphrases, ensure that the PowerExchange Listener runs with a security setting of SECURITY=(1,N) or higher in the DBMOVER member. For more information, see "SECURITY Statement" in the *PowerExchange Reference Manual*.

Note: On z/OS, a valid RACF passphrase can be up to 100 characters in length. PowerExchange truncates passphrases longer than 100 characters when passing them to RACF for validation.

- **EPWD**. An encrypted password for the specified user.

  For access to i5/OS or z/OS, you can enter an encrypted PowerExchange passphrase instead of an encrypted password. Do not encrypt a passphrase that contains invalid characters, such as double-quotation marks, single quotation marks, or currency symbols.

{PRINT|MODIFY}

Specify one of the following keywords:

- **PRINT**. Prints the CDEP details for the specified application.
- **MODIFY**. Removes details for the specified application from the CDEP file based on the days parameter.

APPL

Set to APPL.

appname

Name of the application that you want to print or modify. To specify all applications, enter “ALL.” To specify multiple applications with the same name pattern, include the asterisk (*) wildcard character, for example, LULU*.

days

The number of days of information that the command processes.

For example, the following statement removes all progress details for the application LULU01 that are more than 21 days old:

```
modify appl LULU01 21
```

The following statement prints all progress details for the application LULU01 for the previous 21 days:

```
print appl LULU01 21
```

To remove all details for a particular application use 0 force. For example:

```
modify appl LULU01 0 force
```

If the days parameter is not specified, the utility prints progress details for the last seven days or removes (modifies) details that are older than 40 days.

CDEP Definition Examples

The following are examples of CDEP definitions and meanings:

The following statement prints the progress details of all applications in the CDEP file that occurred within the previous 256 days:

```
print appl ALL 256
```

The following statement removes all progress details for the application LULU03 prior to the last 14 days:

```
modify appl LULU03 14
```

The following statement removes all details of the application LULU06:

```
modify appl LULU06 0 force
```

Running the DTLUCDEP Utility on i5/OS

To run the DTLUCDEP utility on i5/OS:

1. Verify the definitions in the CFG(DTLUCDEP) definition file.
2. Enter the following command:

```
SBMJOB CMD(CALL PGM(DTLLIB/DTLUCDEP)PARM('CS=DATALIB/CFG(DTLUCDEP)')) JOB(MYJOB)
JOBD(DATALIB/DTLLIST) PRTDEV(*JOBD) OUTQ(*JOBD) CURLIB(*CRTDFT) INLLIBL(*JOBD)
```

Running the DTLUCDEP Utility on Linux, UNIX, and Windows

To run the DTLUCDEP utility on Linux, UNIX, and Windows:

1. Verify the definitions in the dtlucdep.txt definition file.
2. Enter the following command:

```
DTLUCDEP
```

Running the DTLUCDEP Utility on z/OS

PowerExchange provides sample JCL for the DTLUCDEP utility in the DTLUCDEP member of the RUNLIB library.

To run the DTLUCDEP utility on z/OS:

1. The following JCL statements are required to run the utility. Specify the DTLUCDEP definitions in-stream, as follows, or in a referenced PDS by using the DD statement.

```
//jobname JOB
//STEP1   EXEC PGM=DTLUCDEP
//*
//* or   EXEC PGM=DTLUCDEP,PARM=('CS=DD:DTLUCDEP')
//* which uses the specified DD instead of sysin
//*
//STEPLIB  DD DISP=SHR,DSN=&HLQ..LOADLIB
//         DD DISP=SHR,DSN=&HLQ..LOAD
//         DD DISP=SHR,DSN=&SCERUN
```

```
//DTLCACDE  DD DSN=&HLQVS..CDEP,
//             DISP=(SHR)
//DTLMSG    DD DSN=&HLQ..DTLMSG,
//             DISP=(SHR)
//DTLCFG    DD DSN=&RUNLIB(DBMOVER),
//             DISP=(SHR)
//DTLKEY    DD DSN=&RUNLIB(LICENSE),
//             DISP=(SHR)
//DTLSGN    DD DSN=&RUNLIB(SIGNON),
//            DISP=(SHR)
//DTLLOG    DD SYSOUT=*
//DTLLOG01  DD SYSOUT=*
//SYSOUT    DD SYSOUT=*
//SYSPRINT  DD SYSOUT=*
//
//SYSIN DD *
  user DTLUSR  epwd A3164A3622798FDC
  print  appl testapp
/*
```

The JCL statements are:

JOB

Initiates the job.

EXEC PGM=DTLUCDEP

Invokes the utility.

STEPLIB DD

Defines the PowerExchange LOAD library that contains the utility.

DTLCACDE DD

Defines the CDEP file.

DTLMSG DD

Defines the PowerExchange message file.

DTLCFG DD

Defines the DBMOVER configuration file.

DTLKEY DD

Defines the license key file.

DTLSGN DD

Defines the selective sign-on file.

DTLLOG DD

Defines the PowerExchange message log file. PowerExchange writes messages to this log file until the alternative logging subtask is initialized.

DTLLOG01 DD

If you enable alternative logging, defines the PowerExchange alternative message log file.

SYSOUT DD

Defines the destination of printed output.

SYSPRINT DD

Defines the print location for the report.

2. Verify the definitions in the JCL.
3. Submit the DTLUCDEP job.

DTLUCDEP Utility on i5/OS Example

The following output is an example of the results of the DTLUCDEP utility:

```
03/11/04 10:01:22                    POWEREXCHANGE/CFG(DTLUCDEP) CARDS
================================================================================
user XXXXXX  pswd 889B042B53F132B7
   print appl ALL 60
Print of requested All Applications
                    since 03/09/05 10:01:22
================================================================================
Print of testdota : All Applications
--------------------------------------------------------------------------------
Application name=<testdota> AS4 Rsttkn=<1>
                       Ainseq=<0>
  First run started  =<03/06/13 16:26:19> ended <03/06/13 17:06:08>
          sequence   =<2A102FE20A3600000000000000000770
                       66F22A102FE20A360000000000000000
                       077066F1>
          restart    =<D9D6C4E3C5E2E3F32A102FE20A360000
                       000000000000077066F0>
   Last run started  =<03/06/13 16:26:19> ended <03/06/13 17:06:08>
          sequence   =<2A102FE20A3600000000000000000770
                       66F22A102FE20A360000000000000000
                       077066F1>
          restart    =<D9D6C4E3C5E2E3F32A102FE20A360000
                       000000000000077066F0>
Current run started  =<> ended <>
          sequence   =<00000000000000000000000000000000
                       00000000000000000000000000000000
                       00000000>
          restart    =<00000000000000000000000000000000
                       0000000000000000000>
   Tokens supplied by the token utility
    Registration name=<dot1.1> tag=<AS4RODTEST3dot11>
          sequence   =<2A2F96A18FC000000000000000000000
                       00F02A2F96A18FC00000000000000000
                       000000F0>
          restart    =<D9D6C4E3C5E2E3F32A2F96A18FC00000
                       00000000000000000F0>
Print of progress for testdota since 03/09/05 10:01:22
 No progress for Application name=<testdota>
Print of testdota1 : All Applications
--------------------------------------------------------------------------------
```

DTLUCDEP Utility on Linux, UNIX, and Windows Example

The output can be piped to a text file if required using the normal command line pipe option. For example:

```
DTLUCDEP > output.txt
```

The following output is an example of the results of the DTLUCDEP utility:

```
2.2.4    DTLUCDEP Example output from the utility
03/10/31 15:46:12                    V:\bin\dtlucdep.txt CARDS
================================================================================
   print appl LULU03

Print of requested Application LULU03 only
                    since 03/10/24 15:46:12
================================================================================
Print of LULU03 : Application LULU03 only
```

```
================================================================================
Application name=<LULU03>  Rsttkn=<0>
                     Ainseq=<0>
  First run started  =<03/10/24 11:17:37> ended <03/10/24 11:18:04>
          sequence   =<0000000002B99600000000002B995>
          restart    =<0000000002B9944D5045584C5F535953
                       54454D5F564F4C554D455F534554>
   Last run started  =<03/10/24 11:17:37> ended <03/10/24 11:18:04>
          sequence   =<0000000002B99600000000002B995>
          restart    =<0000000002B9944D5045584C5F535953
                       54454D5F564F4C554D455F534554>
Current run started  =<> ended <>
          sequence   =<00000000000000000000000000000>
          restart    =<00000000000000000000000000000000
                       00000000000000000000000000000>

Print of progress for LULU03 since 03/10/24 15:46:12
 No progress for Application name=<LULU03>
```

DTLUCDEP Utility on z/OS Example

The following output is an example of the results of the DTLUCDEP utility:

```
03/11/04 12:04:51                    SYSIN CARDS
================================================================================
  user DTLUSR  epwd A3164A3622798FDC
   print  appl testapp
   modify appl all 40
Print of requested Application testapp only
                   since 03/10/28 12:04:51
================================================================================
DTL-04558 Application Index data for <testapp> not found.
 Application name=<testapp> does not exist
Modify for requested All Applications
                    before 03/09/25 12:04:51
================================================================================

Modify of TESTRUN : All Applications

Modify of progress for TESTRUN
                     before 03/09/25 12:04:51
 No progress for Application name=<TESTRUN>
MOD Application name=<TESTRUN>  Rsttkn=<0>  Ainseq=<0>

  First run started  =<03/11/04 12:01:10> ended <03/11/04 12:01:45>
          sequence   =<000000004F0200000000000000004D1B
                       00000000>
          restart    =<C4D6C3D34040000000003D4700000000>
   Last run started  =<03/11/04 12:02:46> ended <03/11/04 12:03:12>
          sequence   =<000000004F0200000000000000004D1B
                       00000000>
          restart    =<C4D6C3D34040000000003D4700000000>
Current run started  =<> ended <>
          sequence   =<00000000000000000000000000000000
                       00000000>
          restart    =<000000000000000000000000000000000>
    Application TESTRUN - 0 progress entries expired
 Application name=<>
                    0 applications 0 progress entries expired

****************************** BOTTOM OF DATA ******************************
```

CHAPTER 10

DTLUCSR2 - IDMS SR2 and SR3 Records Utility

This chapter includes the following topics:

DTLUCSR2 Utility Overview

When an IDMS record no longer fits on its home page, IDMS relocates the record and generates a control record on the home page that points to the relocated record. The relocated record is known as the *SR3* record, and the control record is known as the *SR2* record. If an update or delete occurs on a SR3 record, the IDMS log-based ECCR needs to get the original record ID from the SR2 record to determine if the change is eligible for change capture. To enable the ECCR to find this information, run the DTLUCSR2 utility. The utility records the pairs of matching SR2 and SR3 records in an internal table. The ECCR can then perform a lookup on the table with the SR3 database key to find the matching SR2 record that contains the original record ID.

Run the DTLUCSR2 utility before you start the ECCR for the first time and after events that tend to generate SR2 and SR3 records. For example, run the utility after the following events:

- An IDMS REORG operation
- An IDMS dictionary migration utility (RHDCMIG1 and RHDCMIG2) run
- An alter table operation that adds one or more columns, or any other schema change that can increase the record size
- The following PowerExchange program logic errors, which are issued for an after image (AFTR) or before image (BFOR):

```
PWX-00999 Program logic error. Prog="program". Line=line_number. P1="UOW - SR3 AFTR
hex_SR3_database_key, not found in hash table". P2=1

PWX-00999 Program logic error. Prog="program". Line=line_number. P1="UOW - SR3 BFOR
hex_SR3_database_key, not found in hash table". P2=1
```

After you run the utility, restart the ECCR so that it can detect the SR2 and SR3 pairs that the utility recorded.

Running the DTLUCSR2 Utility

Run the DTLUCSR2 utility before you run the IDMS log-based ECCR the first time and after any event that tends to create SR2 and SR3 records.

Before you start the utility, ensure that you added the SR2INPUT DD statement to the IDMS log-based ECCR JCL. This DD statement points to the utility result files that contain information for building the SR2-SR3 internal table. For more information, see the *PowerExchange CDC Guide for z/OS*.

1. Edit the DTLICSRI member in the RUNLIB library.

   For each database with source tables to be registered for change capture, customize the following sample statements:

   ```
   Read,
   DD_NAME=ddname
   PAGE_GROUP=n
   RADIX=x
   ```

   The following table describes these statements:

Statement	Description
DD_NAME	The DDNAME to be added to the DTLUCSR2 JCL. This name does not have to match a DD name from an IDMS region, but it must exactly match the DD name in the DTLUCSR2 JCL. Format: DD_NAME=STUDENT
PAGE_GROUP	If the database file is normally accessed with a page group other than zero, you must specify the PAGE_GROUP number.
RADIX	If you want to use a RADIX value other than the default of 8, enter a value from 2 to 12.

   Note: DTLUCSR2 writes control information to the SR2TOTAL file and SR2/SR3 link information to the SR2OUT file. These files are created with default information at installation time. You might need to change the file sizes, depending on the number of SR3 records.

2. Add DD cards to the DTLUCSR2 JCL that match the DD names in the DTLICSRI parameter file.

   The DD cards point to the relevant IDMS data set names.

3. Run the JCL in RUNLIB member DTLUCSR2.

CHAPTER 11

DTLUCUDB - DB2 for Linux, UNIX, and Windows CDC Utility

This chapter includes the following topics:

DTLUCUDB Utility Overview

The DTLUCUDB utility performs the following functions:

- Creates a DB2 catalog snapshot to initialize the PowerExchange capture catalog table.
- Generates diagnostic information.

For more information about this utility, see the *PowerExchange CDC Guide for Linux, UNIX, and Windows*.

Running the DTLUCUDB Utility

You can run the DTLUCUDB utility in either of the following ways:

- Issue the command directly from the command line, for example:

  ```
  DTLUCUDB HELP
  ```

- Create a file that contains the commands you want to run and then call that file from the command line, for example:

  ```
  DTLUCUDB mycommands.txt
  ```

  Tip: Use a file when you run a number of different commands at the same time. You can include comments in the file by prefixing the comment line with a slash and asterisk (/*).

DTLUCUDB Utility Syntax

The DTLUCUDB syntax optionally includes database keywords for all of the command options except HELP. The database keywords provide information for connecting to a DB2 database. Although these keywords are optional, you should specify them if you do not want to use the defaults.

The following table describes the database keywords:

Keyword	Syntax	Description
DB	[DB=*database_name*]	Name of the DB2 database to which you want to connect. Default is SAMPLE.
UID	[UID=*user_id*]	User ID to use for connecting to the database. Default is logon user ID. For a database on a supported Linux, UNIX, or Windows system, if you have enabled PowerExchange LDAP user authentication and disabled relational pass-through authentication, the user ID is the enterprise user name. For more information, see the *PowerExchange Reference Manual*.
{PWD|EPWD}	[{PWD=*password*| EPWD=*encrypted_password*}]	Password or encrypted password for the specified user ID. Do not specify both.

In the DTLUCUDB syntax, the database keywords are represented by the italicized phrase *database keywords*.

The DTLUCUDB utility has the following syntax:

```
CCATDMP [database keywords]
  [CCATALOG=table_name]
  [FILE=file_name]
  [REPLACE={N|Y}]
;
DBINFO [database keywords]
;
DUMPDIAG [database keywords]
  [CCATALOG=table_name]
  BVTS=begin_VTS
  [EVTS=end_VTS]
  DIR=dump_directory
  [REPLACE={N|Y}]
;
HELP
;
LOGPRT [database keywords]
  [CCATALOG=table_name]
  [PART=DB partition_number]
  [FILE=file_name]
  [REPLACE={N|Y}]
  [RECSPERFILE=records_per_output_file]
  {BLSN=begin_LSN|BVTS=begin_VTS}
  [ELSN=end_LSN]
  [EVTS=end_VTS]
  [RECS=records_to_select]
  [TRANID=transaction_ID]
  [LOGICAL={Y|N}]
  [UDB={N|MIN|FMT|MAX}]
;
SETDEF [database keywords]
  [CCATALOG=table_name]
;
SNAPSHOT [database keywords]
  [CCATALOG=table_name]
  [REPLACE={N|Y}]
;
SNAPUPDT [database keywords]
  [CCATALOG=table_name]
  [REPLACE={N|Y}]
  [ARCHIVEOLDPOSITIONING={N|Y}]
;
```

```
SQUISH [database keywords]
  [CCATALOG=table_name]
  VTSDT=VTS_date_time
  REPLACE={Y|N}
;
UPDTDRP [database keywords]
  [CCATALOG=table_name]
  VTSDT={EOC|NOW|VTS date_time}
;
```

Command Options for the DTLUCUDB Utility

The DTLUCUDB utility has the following command options:

CCATDMP Command

The CCATDMP command produces a dump file that contains SQL insert statements corresponding to the contents of the capture catalog table.

The default file name is `ccatdmp.database_name.capture_catalog_name.sql`. The file is saved to the current working directory when the command is executed.

```
CCATDMP [database keywords]
  [CCATALOG=table_name]
  [FILE=file_name]
  [REPLACE={N|Y}]
;
```

The following table describes the parameters in the CCATDMP command:

Parameter	Description
CCATALOG	Name of the capture catalog table. Default is *current_user*.DTLCCATALOG.
FILE	Name of the dump file. This name overrides the default file name: ccatdmp.*database_name*.*capture_catalog_name*.sql.
REPLACE	REPLACE=Y overwrites an existing data in the file. Default is N.

DBINFO Command

The DBINFO command prints out environmental information.

```
DBINFO [database keywords];
```

An example of this type of information is:

```
PWX-20526 UDB capture DB/DBMS Info:
PWX-20527         SQL_DATABASE_NAME: CAPTURE
PWX-20527           SQL_SERVER_NAME: DB2
PWX-20527             SQL_USER_NAME: PWXUSER
PWX-20527             SQL_DBMS_NAME: DB2/NT
PWX-20527              SQL_DBMS_VER: 08.02.0004
PWX-20527 SQL_IDENTIFIER_QUOTE_CHAR: "
PWX-20527      SQL_CONNECT_CODEPAGE: 1252
PWX-20527     SQL_DATABASE_CODEPAGE: 1252
PWX-20527  SQL_APPLICATION_CODEPAGE: 1252
PWX-20527                INST_NAME: DB2
PWX-20527     IS_INST_PARTITIONABLE: 1
PWX-20527          NUM_DBPARTITIONS: 5
PWX-20527             INST_PTR_SIZE: 32
PWX-20527               RELEASE_NUM: 03050106
PWX-20527             SERVICE_LEVEL: DB2 v8.1.11.973
PWX-20527                 BLD_LEVEL: s060120
PWX-20527                       PTF: WR21365
PWX-20527               FIXPACK_NUM: 11
PWX-20527                   OS_NAME: WIN32_NT
PWX-20527                OS_VERSION: 5.2
PWX-20527                OS_RELEASE: Service Pack 1
PWX-20527                 HOST_NAME: S160019
PWX-20527                TOTAL_CPUS: 2
PWX-20527           CONFIGURED_CPUS: 4
PWX-20527              TOTAL_MEMORY: 3072
PWX-20527         CATALOG_PARTITION: 0
PWX-20528 Partition[  0]: S160019.informatica.com, 0, S160019
PWX-20541 LSN at first DB connect: 00003921000C0000
PWX-20541 LSN at End of Log: 00003921000C0000
PWX-20528 Partition[  1]: S160019.informatica.com, 1, S160019.informatica.com
PWX-20541 LSN at first DB connect: 0000088B800C0000
PWX-20541 LSN at End of Log: 0000088B800C0000
PWX-20506 Command DBINFO complete
```

DUMPDIAG Command

The DUMPDIAG command produces files for the capture catalog, general database information, and the DB2 log records for each partition in the directory that is specified by the DIR parameter.

```
DUMPDIAG [database keywords]
  [CCATALOG=table_name]
  BVTS=begin_VTS [EVTS=end_VTS]
  DIR=dump_directory [REPLACE={N|Y}]
;
```

The following table describes the parameters in the DUMPDIAG command:

Parameter	Description
CCATALOG	Name of the capture catalog table.
BVTS	Required. The starting timestamp for the diagnostics file in the format *YYYY-MM-DD-HH.MI.SS.SSSSSS*. The year, month and day are required. BVTS values are always specified in GMT (Greenwich Mean Time).

Parameter	Description
EVTS	The ending timestamp for the diagnostics file in the format *YYYY-MM-DD-HH.MI.SS.SSSSSS*. The year, month and day are required. EVTS values are always specified in GMT (Greenwich Mean Time).
DIR	Required. The directory where the diagnostics file is written. The file name is "ccatdmp.*database_name*.*capture_catalog_name*.sql" and cannot be changed.
REPLACE	Indicates whether to overwrite existing files. Specify Y to overwrite existing files.

An example of this type of information is:

```
PWX-20512 Producing file 'dtst20061221\ccatdmp.cap14.partcaptst.sql'
PWX-20512 Producing file 'dtst20061221\dbconfig.txt'
PWX-20512 Producing file 'dtst20061221\p0.logdmp'
PWX-20540 Begin LSN 0000042B3EBC0000 selected for BVTS value
PWX-20519 End of UDB log file reached
PWX-20512 Producing file 'dtst20061221\p1.logdmp'
PWX-20540 Begin LSN 00000768C1040000 selected for BVTS value
PWX-20519 End of UDB log file reached
PWX-20512 Producing file 'dtst20061221\p20.logdmp'
PWX-20540 Begin LSN 0000046B76C10000 selected for BVTS value
PWX-20519 End of UDB log file reached
PWX-20506 Command DUMPDIAG complete
```

HELP Command

The HELP command prints the full syntax of the DTLUCUDB command.

LOGPRT Command

The LOGPRT command produces a file that formats the contents of the DB2 log. By default, the command creates a file named "*database_name*.logprt" in the current working directory.

The command syntax is:

```
LOGPRT [database keywords]
  [CCATALOG=table_name]
  [PART=DB partition_number]
  [FILE=file_name]
  [REPLACE={N|Y}]
  [RECSPERFILE=records_per_output_file]
  BLSN={begin_LSN|BVTS=begin_VTS}
  [ELSN=end_LSN]
  [EVTS=end_VTS]
  [RECS=records_to_select]
  [TRANID=transaction_ID]
  [LOGICAL={Y|N}]
  [UDB={N|MIN|FMT|MAX}]
;
```

The following table describes the parameters in the LOGPRT command:

Parameter	Description
CCATALOG	Name of the capture catalog table.
PART	Database partition number.

Parameter	Description
FILE	Name of the formatted log file. This overrides the default “<database name>.logprt“ file.
REPLACE	Indicates whether to overwrite an existing file. Specify Y to overwrite an existing file.
RECSPERFILE	An option that can be used to divide a large amount of output into multiple files. The generated file names have the format: *database_name.first_lsn_value_in_file*.logprt. If the FILE keyword has also been specified, the generated file names have the format: *file_name*. *first_lsn_value_in_file*.
BLSN	A 6-byte DB2 Log Sequence Number (LSN), in hexadecimal digits, that indicates where the command is to start reading in the log. This value must represent an actual LSN. If fewer than 12 hexadecimal digits are specified, zeros are logically added to the left. BLSN defaults to the beginning of the active log. You must specify either BLSN or BVTS.
BVTS	Starting timestamp that indicates where the command is to start reading in the log. You must specify either BLSN or BVTS.
ELSN	A 6-byte DB2 Log Sequence Number (LSN), in hexadecimal digits, that specifies where the command is to stop. This value is not required to correspond to an actual LSN. If fewer than 12 hexadecimal digits are specified, zeros are logically added to the left. You can use this option to filter the output. ELSN defaults to the end of the log. You must specify either BLSN or BVTS.
EVTS	Ending timestamp that indicates where the command is to stop. You can use it to filter the output.
RECS	Number of records that indicates where the command is to stop. You can use this option to filter the output.
TRANID	Criteria for filtering output. This option does not stop the reading of log records when transaction-end log records are processed.
LOGICAL	DB2 log reading consists of reading actual DB2 log records and interpreting them into logical events (known as logical log records). The LOGICAL keyword can be used to force these log records to be printed in the file. Default is Y.
UDB	Controls how “real” DB2 log records are formatted in the file. Valid options are: - N. Does not print at all (default). - MIN. Prints a minimum of information. - FMT. Formats what is known about the record. - MAX. Dumps the record in hex and formats it.

SETDEF Command

The SETDEF command sets default values for keywords on the other commands.

```
SETDEF [database keywords]
  [CCATALOG=table_name]
;
```

The following table describes the parameter in the SETDEF command:

Parameter	Description
CCATALOG	Name of the capture catalog table. Default is DTLCCATALOG.

SNAPSHOT Command

The SNAPSHOT command is used to initialize capture catalog table. Note that restart points cannot precede the point in the log where a snapshot is taken. Therefore, use this command carefully.

```
SNAPSHOT [database keywords]
  [CCATALOG=table_name]
  [REPLACE={N|Y}]
;
```

The following table describes the parameters in the SNAPSHOT command:

Parameter	Description
CCATALOG	Name of the capture catalog table to be initialized. Default is DTLCCATALOG.
REPLACE	Indicates whether to overwrite any existing rows of data in the capture catalog table. If rows of data exist, you must specify Y. Default is N.

SNAPUPDT Command

Use the SNAPUPDT command after partitions are added to or dropped from the database instance. For each new partition, the command adds a new partition positioning entry in the capture catalog. For each partition that is dropped, the command removes a positioning entry from the capture catalog.

```
SNAPUPDT [database keywords]
  [CCATALOG=table_name]
  [REPLACE={N|Y}]
  [ARCHIVEOLDPOSITIONING={N|Y}]
;
```

The following table describes the parameters in the SNAPUPDT command:

Parameter	Description
CCATALOG	Name of the capture catalog table.
REPLACE	REPLACE=Y must be specified to update the capture catalog. If REPLACE is not set to Y, the command shows what changes would be made, but does not make them.
ARCHIVEOLDPOSITIONING	If you specify ARCHIVEOLDPOSITIONING=Y, the positioning entries remain in the capture catalog, but are not accessible.

SQUISH Command

Use the SQUISH command to advance the base of the capture catalog to a new VTS date and time by collapsing catalog entries (table or column alters) and removing positioning entries. Catalog (any DDL activity) and positioning (VTS, LSN, or partition set) entries are added to an active capture catalog during extraction processing.

```
SQUISH [database keywords]
  [CCATALOG=table_name]
```

```
  VTSDT=VTS date_time
  REPLACE={Y|N}
;
```

The following table describes the parameters in the SQUISH command:

Parameter	Description
CCATALOG	Name of the capture catalog table.
REPLACE	Specify Y to be able to update the capture catalog. If N is specified, the command shows the changes but does not make them.
VTSDT	A virtual timestamp (date and time). This timestamp value must be within the bounds of the capture catalog.

Note: Do not run the SQUISH command while extractions are active. Perform a backup before running SQUISH.

UPDTDRP Command

Use the UPDTDRP command to update the default restart point.

```
UPDTDRP [database keywords]
  [CCATALOG=table_name]
  VTSDT={EOC|NOW|VTS date_time}
;
```

The following table describes the parameters in the UPDTDRP command:

Parameter	Description
CCATALOG	Name of the capture catalog table.
VTSDT	Required. The value must be greater than lowest VTS value in the capture catalog and less than the current end-of-log VTS value. The value is one of the following: - EOC. End of catalog. - NOW. Current date and time. - VTS *date_time*. The virtual timestamp that has the specified date and time, for example, 2007-09-07.18.40.47

Gathering Diagnostic Information to Resolve a DB2 Capture Problem

Informatica Global Customer Support might request diagnostic information to use in resolving a DB2 capture problem. The following commands are example diagnostic commands that are entered at a Windows command line:

```
mkdir prob1234
cd /prob1234
dtlucudb dumpdiag db=mydb ccatalog=my.capturecat bvts=<start time> evts=<end time>
```

The directory, prob1234, contains several files. You would zip these files and send them to Informatica Global Customer Support for analysis.

Note: If you specify the EVTS option for the DUMPDIAG command, verify that the problem section of the log is captured.

CHAPTER 12

DTLULCAT and DTLULOGC - IDMS Log Catalog Utilities

This chapter includes the following topics:

DTLULCAT and DTLULOGC Utilities Overview

The Log Catalog holds information about the IDMS logs which are available for the use of PowerExchange log-based capture. During the initial installation of PowerExchange, a Log Catalog VSAM file will be created (default naming will be &HLQ..LOGSCAT) and a dummy record will be added.

For IDMS log-based capture to work effectively, it is vital to ensure that the log catalog is updated in a timely fashion and that log information is both secure and available. If the logs are not in the catalog, the records they hold will be unknown to PowerExchange. The correct way to add information to the catalog is to use utility DTLULCAT to format the input, then run DTLULOGC to amend the Log Catalog with that prepared input.

RUNLIB member DTLULCAU is supplied to run the two utilities one after the other. It is expected that this be scheduled to run as soon as the latest IDMS log had been spooled off. There may, however, be times when DTLULOGC is run in isolation, involving manual coding of the input file.

Correct scheduling of the addition logs to the Log Catalog is vital to obtaining timely data from the log-based IDMS capture environment.

Related Topics:

- "Guidelines for Adding Logs to the Catalog with the DTLULCAT and DTLULOGC Utilities" on page 145

Running the DTLULCAT Utility

This utility program is used to take the supplied journal name and use it to prepare the input required by the catalog utility program DTLULOGC. The utility is delivered as an executable on Windows and member DTLULCAT in RUNLIB on MVS.

Sample statements follow:

```
IDMS_VERSION=15
FILE_TYPE=C
MEDIA_TYPE=D
MEDIA_CONTENT=BI
SERVICE=IDMSE150
INSTANCE_IDENTIFIER=XYLOGSID
```

The following table describes the sample statements:

Parameter	Description
IDMS_VERSION	The version in which the journal record structure last changed. The value of 15 is the only valid value for the supported IDMS versions, all of which are later than IDMS Version 14.
FILE_TYPE	File type. Specifies one of the following: - C. Central version. - L. Local mode.
MEDIA_TYPE	Specifies one of the following: - T. Tape. - D. Disk.
MEDIA_CONTENT	Determines the images of changed records delivered: - BI. Before images. - AI. After images. - BA. Both before and after images.
SERVICE	IDMS CV name or Local Job name.
INSTANCE_IDENTIFIER	Chosen LOGSID identifier.

The utility DTLULCAT writes to DDCARD SYSPUNCH. This file is then the input to utility DTLULOGC.

Running the DTLULOGC Utility

The utility DTLULOGC populates the log catalog with information about the logs to process. The example below shows sample JCL DTLULCAU to run both DTLULCAT followed by DTLULOGC. Running DTLULCAU JCL is the recommended method of adding to the Log Catalog.

This example adds log DTLUSR.IDMS.E15SP0.OFF.LOADED.JOURNAL1 for an IDMS Version 18 environment with CV Name IDMSE150, where the log resides on disk storage and will be accessed using a LOGSID value of XYLOGSID. Here the SYSIN data is shown as instream for clarity, but the sample JCL is delivered pointing to member DTLIDLC when running against a CV (DTLIDLL for Local Job mode) in which these statements would normally be placed.

```
//*****************************************************************/
//*                                                               */
//* SAMPLE JCL TO:-                                               */
//*                                                               */
//* CAPTURE IDMS JOURNAL FILE INFORMATION AND INPUT STREAM        */
//* INTO FOR DTLULOGC LOG FILE CATALOG ROUTINE.                   */
//*                                                               */
//* NORMALLY THE SYSIN INPUT STREAM WOULD BE A PDS MEMBER.        */
//*                                                               */
//* THIS NEEDS TO BE INTEGRATED INTO THE END USERS JOURNAL        */
//* ARCHIVING PROCEDURE, WHICH MAY BE DIFFERENT FROM SITE TO SITE.*/
//*                                                               */
//* A MECHANISM WILL NEED TO BE ESTABLISHED TO REPLACE THE DATASET*/
//* SPECIFIED VIA THE LOGFILE DD STATEMENT WITH THE LOGFILE       */
//* WHICH IS CURRENTLY THE OBJECT OF THE USERS ARCHIVING PROCEDURE*/
//* AND OUR CATALOG OPERATION                                     */
//*                                                               */
/*****************************************************************/
//INCS1 INCLUDE MEMBER=GENBULK
//DTLULCAT EXEC PGM=DTLULCAT
//STEPLIB  DD DISP=SHR,DSN=DTLUSR.V800B14.LOADLIB
//DTLCFG   DD DISP=SHR,DSN=DTLUSR.V800B14.RUNLIB(DBMOVER)
//DTLKEY   DD DISP=SHR,DSN=DTLUSR.V800B14.RUNLIB(LICENSE)
//DTLMSG   DD DISP=SHR,DSN=&HLQ..DTLMSG,FREE=CLOSE
//DTLLOG   DD SYSOUT=*
//LOGFILE  DD DISP=SHR,DSN=DTLUSR.IDMS.E15SP0.OFF.LOADED.JOURNAL1
//SYSPRINT DD SYSOUT=*
//SYSPUNCH DD DSN=&&LOGDATA,
//        DISP=(,PASS),
//        SPACE=(CYL,(2,1),RLSE),
//        DCB=(RECFM=FB,LRECL=80,BLKSIZE=3120)
//SYSIN    DD *
IDMS_VERSION=15
FILE_TYPE=C
MEDIA_TYPE=D
MEDIA_CONTENT=BI
SERVICE=IDMSE150
INSTANCE_IDENTIFIER=XYLOGSID
/*
//DTLULOGC EXEC PGM=DTLULOGC
//STEPLIB   DD  DISP=SHR,DSN=DTLUSR.V800B14.LOADLIB
//DTLCFG    DD  DISP=SHR,DSN=DTLUSR.V800B14.RUNLIB(DBMOVER)
//DTLKEY    DD  DISP=SHR,DSN=DTLUSR.V800B14.RUNLIB(LICENSE)
//DTLSGN    DD  DISP=SHR,DSN=DTLUSR.V800B14.RUNLIB(SIGNON)
//DTLMSG    DD  DISP=SHR,DSN=&HLQ..DTLMSG
//LOGSCAT   DD  DISP=SHR,DSN=DTLUSR.V800B14.V1.LOGSCAT
//DTLLOG    DD  SYSOUT=*
//SYSUDUMP  DD  SYSOUT=*
//SYSPRINT  DD  SYSOUT=*
//REPORT    DD  SYSOUT=*
//EXPORT    DD  SYSOUT=*
//SYSIN     DD  DISP=SHR,DSN=&&LOGDATA
```

Note: For IDMS_VERSION, specify 15 for IDMS versions later than version 14.

Manually Manipulating the Log Catalog

During the normal course of IDMS log processing, the Log Catalog will be updated using the combination of DTLULCAT and DTLULOGC to add the next available log. You might need to alter details for log entries or remove logs from the catalog. To do this, DTLULOGC (DTLULOGC JCL in RUNLIB) will be run stand-alone with hand-coded input.

The utility allows the user to:

- Add an instance
- Add a log
- Update a log entry
- Delete an entry
- Export an entry to another data set for offload

The following list shows the keywords and parameters available to code in an 80 byte file, which you specify as input in the SYSIN DD card. See the sample JCL.

ADD_INSTANCE *parameters*

Add a LOGSID instance to the catalog. Each LOGSID used requires an instance to be added to the log catalog.

The following table shows the parameters available for the ADD_INSTANCE keyword:

Parameter	Description
INSTANCE_IDENTIFIER	LOGSID value
VERSION	Version number of the entry

ADD_ENTRY *parameters*

Adds a specific log to the log catalog.

The following table shows the parameters available for the ADD_ENTRY keyword:

Parameter	Description
BLOCK_SIZE	Block size of the log. Required if the logs are to be shipped to another platform.
ENTRY_NUMBER	Sequential number, which should be incremented by 1 for each new log added to the log catalog.
FILE_TYPE	- C. Central or Shared Service Log or Journal. - L. Local Mode or Unshared Service Log or Journal.
FIRST_RECORD_SEQUENCE_NUMBER	Sequence number of the first record in the block.
FIRST_RECORD_TIME_STAMP	Timestamp of the first record in the block.
IDMS_VERSION	Version number of IDMS. Specified as an integer.
INSTANCE_IDENTIFIER	LOGSID value

Parameter	Description
LAST_RECORD_IDENTIFIER	Record ID of the last record in the block or zeros if a non-data record.
LAST_RECORD_OFFSET	Offset of last valid offset in the block.
LOG_DATA_TYPE	IDL for MVS IDMS log data.
LOG_FILE_NAME	Name of IDMS log file.
MEDIA_CONTENT	- AI. Only contains After images. - BI. Only contains Before images. - BA. Contains both Before and After images.
MEDIA_TYPE	- D. Disk. - T. Tape.
NUMBER_OF_BLOCKS	Number of blocks in the log.
SERVICE	CV name or Local Mode job name.
STATUS	- A. Active. - S. Skip. - T. Terminate.
ENTRY_TYPE	- 1. File entry. - 2. Reserved for future use.
VERSION	Version number of the entry.

UPDATE_ENTRY *parameters*

Updates a log entry. The entry is identified by the value of INSTANCE_IDENTIFIER and ENTRY_NUMBER.

Valid parameters are those listed for ADD_ENTRY.

DELETE_ENTRY INSTANCE_IDENTIFIER=*instance_identifier*

Deletes the oldest log for the specified INSTANCE_IDENTIFIER.

REPORT_INSTANCE INSTANCE_IDENTIFIER=*instance_identifier*

Lists catalog entries for the specified INSTANCE_IDENTIFIER.

EXPORT_INSTANCE INSTANCE_IDENTIFIER=*instance_identifier*

Used to export all information for a specified INSTANCE_IDENTIFIER to a file.

Note: Keyword commands are separated by a semi-colon (;), parameters by a comma (,).

The following sample input adds two instances (LOGSIDs), adds entries (log files), deletes an entry, reports instance LOGSIDA, exports instance LOGSIDA to a file (dtlulgce.txt), and finally deletes instance LOGSIDA:

```
ADD_INSTANCE INSTANCE_IDENTIFIER=LOGSIDA, VERSION=224;
ADD_ENTRY INSTANCE_IDENTIFIER=LOGSIDA, ENTRY_NUMBER=777, VERSION=0, ENTRY_TYPE=1,
STATUS=A, LOG_DATA_TYPE=IDL, IDMS_VERSION=15, FILE_TYPE=C, MEDIA_TYPE=D,
MEDIA_CONTENT=BI, SERVICE=IDMSE150, LOG_FILE_NAME=XXXXXXXXXXXXXXXXXXXXXXXXXXXXX,
BLOCK_SIZE=29000, NUMBER_OF_BLOCKS=445, LAST_RECORD_OFFSET=1119,
LAST_RECORD_IDENTIFIER=3, FIRST_RECORD_SEQUENCE_NUMBER=4,
FIRST_RECORD_TIME_STAMP="05/03/03 10:55:01";
ADD_ENTRY INSTANCE_IDENTIFIER=LOGSIDA, ENTRY_NUMBER=778, VERSION=0, ENTRY_TYPE=1,
STATUS=A, LOG_DATA_TYPE=IDL, IDMS_VERSION=15, FILE_TYPE=C, MEDIA_TYPE=D,
```

```
MEDIA_CONTENT=BI, SERVICE=IDMSE150, LOG_FILE_NAME=MMMMMMMMMMMMMMMMMMMMMMMM,
BLOCK_SIZE=29000, NUMBER_OF_BLOCKS=445, LAST_RECORD_OFFSET=1119,
LAST_RECORD_IDENTIFIER=3, FIRST_RECORD_SEQUENCE_NUMBER=4,
FIRST_RECORD_TIME_STAMP="05/03/03 12:55:01";
ADD_ENTRY INSTANCE_IDENTIFIER=LOGSIDA, ENTRY_NUMBER=779, VERSION=0, ENTRY_TYPE=1,
STATUS=A, LOG_DATA_TYPE=IDL, IDMS_VERSION=15, FILE_TYPE=C, MEDIA_TYPE=D,
MEDIA_CONTENT=BI, SERVICE=IDMSE150, LOG_FILE_NAME=ZZZZZZZZZZZZZZZZZZZCCCCCCCCCCCC,
BLOCK_SIZE=29000, NUMBER_OF_BLOCKS=333, LAST_RECORD_OFFSET=1119,
LAST_RECORD_IDENTIFIER=3, FIRST_RECORD_SEQUENCE_NUMBER=4,
FIRST_RECORD_TIME_STAMP="05/03/03 14:55:01";
ADD_INSTANCE INSTANCE_IDENTIFIER=ABCDE, VERSION=0;
ADD_ENTRY INSTANCE_IDENTIFIER=ABCDE, ENTRY_NUMBER=1, VERSION=0, ENTRY_TYPE=1, STATUS=A,
LOG_DATA_TYPE=IDL, IDMS_VERSION=15, FILE_TYPE=C, MEDIA_TYPE=D, MEDIA_CONTENT=BI,
SERVICE=IDMSE15P, LOG_FILE_NAME=BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB, BLOCK_SIZE=29000,
NUMBER_OF_BLOCKS=444, LAST_RECORD_OFFSET=1112, LAST_RECORD_IDENTIFIER=2,
FIRST_RECORD_SEQUENCE_NUMBER=3, FIRST_RECORD_TIME_STAMP="05/04/03 08:55:01";
ADD_ENTRY INSTANCE_IDENTIFIER=ABCDE, ENTRY_NUMBER=2, VERSION=0, ENTRY_TYPE=1, STATUS=A,
LOG_DATA_TYPE=IDL, IDMS_VERSION=15, FILE_TYPE=C, MEDIA_TYPE=D, MEDIA_CONTENT=BI,
SERVICE=IDMSE15P, LOG_FILE_NAME=CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC, BLOCK_SIZE=29000,
NUMBER_OF_BLOCKS=445, LAST_RECORD_OFFSET=1119, LAST_RECORD_IDENTIFIER=3,
FIRST_RECORD_SEQUENCE_NUMBER=4, FIRST_RECORD_TIME_STAMP="05/04/03 10:55:01";
UPDATE_ENTRY INSTANCE_IDENTIFIER=LOGSIDA, ENTRY_NUMBER=779, VERSION=0, ENTRY_TYPE=1,
STATUS=A, LOG_DATA_TYPE=IDL, IDMS_VERSION=15, FILE_TYPE=C, MEDIA_TYPE=D,
MEDIA_CONTENT=BI, SERVICE=DTLXXXXX, LOG_FILE_NAME=AAAAAAAAAAAAAAAKKKKKKKKKKKKKK,
BLOCK_SIZE=29000, NUMBER_OF_BLOCKS=111, LAST_RECORD_OFFSET=1119,
LAST_RECORD_IDENTIFIER=3, FIRST_RECORD_SEQUENCE_NUMBER=4,
FIRST_RECORD_TIME_STAMP="05/04/03 12:55:01";
DELETE_ENTRY INSTANCE_IDENTIFIER=LOGSIDA;
REPORT_INSTANCE INSTANCE_IDENTIFIER=LOGSIDA;
EXPORT_INSTANCE INSTANCE_IDENTIFIER=LOGSIDA;
DELETE_INSTANCE INSTANCE_IDENTIFIER=LOGSIDA;
```

Guidelines for Adding Logs to the Catalog with the DTLULCAT and DTLULOGC Utilities

Care must be taken with the order in which the logs are added to the catalog. Operational procedures for the running of DTLULCAT and DTLULOGC must be developed to ensure that logs are added in the correct sequence.

The preferred method of operation is for DTLULCAT and DTLULOGC to be included in an archive log job (run together using DTLULCAU JCL) submitted by a WTOEXIT which intercepts a message written to the operator. The following steps need to be included in such a job:

1. Active journal will be off-loaded to an archive log.
2. The archive log should be written to a file, using a utility program like IEBGENER, which can be kept as a unique data set name (a GDG is a good data set structure). This copy of the log must be retained until all changes have been captured by PowerExchange. This file will be added to the PowerExchange Log Catalog by the following steps.
3. Execute DTLULCAT to generate input statements for DTLULOGC.
4. Execute DTLULOGC to update the PowerExchange Log Catalog.

When Central Versions are varied off-line to run in Local Mode, ensure Local Mode logs are added before any new Central Version logs. If a database, previously varied off-line, is subsequently varied back on-line and the Local Mode log is not added immediately, then if a later log is added to the catalog and a subsequent attempt made to add the Local Mode log, then this will fail.

The rule used for checking such log additions is:

- A local mode journal must not be added to the catalog if the last available timestamp within the journal is greater than the timestamp of the previously added CV mode journal.
- If logs are added in the incorrect sequence expect to see messages similar to the following:

```
51007 162240 MVS 1 PWX-19862 IDMS CATLG FILE: Add Entry Failure - Timestamp not
greater than previous for key
XYLOGSID00000000000000000000001AIDL15CDBAIDMSE150DTLUSR.IDMS.D15SP0.OFF.J4
```

CHAPTER 13

DTLURDMO - Data Map Utility

This chapter includes the following topics:

DTLURDMO Utility Overview

Use the DTLURDMO utility to copy the following types of definitions from one environment or location to another:

- PowerExchange data maps
- PowerExchange capture registrations
- PowerExchange extraction maps

When performing a copy, you can optionally change certain attributes of the new capture registration, data map, or extraction map, such as the schema name or table name.

Supported Operating Systems for the DTLURDMO Utility

You can run the DTLURDMO utility on the following platforms:

- i5/OS
- Linux, UNIX, and Windows

- z/OS

Control Statement Overview for the DTLURDMO Utility

DTLURDMO control statements are of the following types:

- Global statements control overall program execution or provide basic information, such as user name or password. Global statements remain active for the entire DTLURDMO execution. You can include them only once in the input file or stream.
- Copy statements specify the type of copy to be performed:
  - DM_COPY copies data maps.
  - REG_COPY copies capture registrations and, optionally, extraction maps.
  - XM_COPY copies extraction maps.

  Copy statements have no operands but can be followed by optional statements. Only a single type of copy statement can appear in the input file or stream, but it can appear multiple times.
- Optional statements follow a copy statement and are valid only for the scope of the execution of the copy statement. Optional statements become inactive when PowerExchange encounters a subsequent copy statement. Optional statements filter the objects selected, rename objects, change object attributes, and set optional functions for the copy.

Control Statement Syntax for the DTLURDMO Utility

The DTLURDMO definition file includes the following control statements:

```
[OUTPUT folder_name;]
USER user_ID;
[PWD password|EPWD encrypted_password];
[TARGETUSER target_user_ID;]
[TARGETPWD password|TARGETEPWD encrypted_password;]
SOURCE source_node;
TARGET target_node;
[REPLACE;]
[DETAIL;]
[VALIDATE;]
[DM_COPY;
[DM_COPY_optional_statements;]]
[REG_COPY;
[REG_COPY_optional_statements;]]
[XM_COPY;
[XM_COPY_optional_statements;]]
```

In the syntax, statements or parameters enclosed in square brackets ([]) are optional. The following rules and guidelines apply:

- All control statements must end with a semicolon (;).
- Statements and parameters are case-insensitive.
- Operands used for comparison, such as operands used to filter objects, are case-insensitive.
- Operands used to rename or modify object attributes are case-sensitive.

- Parameters that are enclosed in parentheses and comma-separated must be specified in that format.
- You must specify exactly one PWD or EPWD statement.
- You must specify exactly one type of copy statement: DM_COPY, REG_COPY, or XM_COPY. You can specify this statement once or multiple times.
- Optional statements follow a copy statement and are valid only for the scope of the execution of the copy statement.

Within the scope of a copy statement, DTLURDMO allows multiple occurrences of each of the EXCLUDE, MODIFY, RENAME, and SELECT statements. However, in most cases it is preferable to include multiple copy statements instead. For more information, see "Scope of Operands" on page 175.

Note: Before you use a DTLURDMO definition file that you used with a previous release of the product, verify that its syntax is consistent with the syntax described in this topic.

Related Topics:

- "Control Statements and Parameters for the DTLURDMO Utility " on page 149

Control Statements and Parameters for the DTLURDMO Utility

This section describes the control statements and their parameters. The section is organized as follows:

- Global statements
- DM_COPY statement
- REG_COPY statement
- XM_COPY statement

The discussion of each copy statement includes a description of its optional statements and their parameters.

Global Statements

Global statements remain active for the entire DTLURDMO execution. You can include them only once in the input file or stream.

The following DTLURDMO statements are global:

- DETAIL
- EPWD
- OUTPUT
- PWD
- REPLACE
- SOURCE
- TARGET
- TARGETEPWD
- TARGETPWD

- TARGETUSER
- USER
- VALIDATE

DETAIL Statement

The DETAIL statement causes DTLURDMO to print a detailed report containing information about the copying process including all changes made and renames performed.

The DETAIL statement has no operands.

This statement is optional.

EPWD Statement

The EPWD statement specifies an encrypted password for the user ID specified in the USER statement. The password and user ID are used to access maps and registrations on the source system.

If the maps and registrations are on an i5/OS or z/OS system, you can enter an encrypted PowerExchange passphrase instead of an encrypted password. Ensure that the passphrase you encrypt does not contain invalid characters, such as double-quotation marks, single quotation marks, or currency symbols.

If the TARGETEPWD statement is not specified, the EPWD statement also specifies the encrypted password or passphrase for the target to which maps and registrations are copied.

Use the following syntax:

```
EPWD encrypted_password_or_passphrase;
```

Note: To encrypt a password or passphrase, you can use the **File** > **Encrypt** command in the PowerExchange Navigator.

Do not also specify the PWD statement.

OUTPUT Statement

The OUTPUT statement specifies an alternative destination folder or data set for the output maps or registrations from the default destination that is specified in the DBMOVER configuration file.

Use the following syntax:

```
OUTPUT folder_name;
```

This statement is optional.

Caution: Do not use the OUTPUT statement with the REG_COPY statement if you also use the CREATEXMAPS statement.

PWD Statement

The PWD statement specifies a password for the user ID specified in the USER statement. The password and user ID are used to access maps and registrations on the source system.

If the maps and registrations are on an i5/OS or z/OS system, you can enter a valid PowerExchange passphrase instead of a password.

If the TARGETPWD statement is not specified, the PWD statement also specifies the password or passphrase for the target to which maps and registrations are copied.

Use the following syntax:

```
PWD password_or_passphrase;
```

Use a PowerExchange passphrase for enhanced security. An i5/OS passphrase can be from 9 to 31 characters in length. A z/OS passphrase can be from 9 to 128 characters in length. A passphrase can contain the following characters:

- Uppercase and lowercase letters
- The numbers 0 to 9
- Spaces
- The following special characters:

  ```
  ' - ; # \ , . / ! % & * ( ) _ + { } : @ | < > ?
  ```

  Note: The first character is an apostrophe.

Passphrases cannot include single quotation marks (‘), double quotation marks (“), or currency symbols.

If a passphrase contains spaces, you must enclose it with double-quotation marks ("), for example, `"This is a passphrase"`. If a passphrase contains special characters, you must enclose it with triple double-quotation characters (""), for example, `"""This passphrase contains special characters ! % & *."""`. If a passphrase contains only alphanumeric characters without spaces, you can enter it without delimiters.

To use passphrases, ensure that the PowerExchange Listener runs with a security setting of SECURITY=(1,N) or higher in the DBMOVER configuration member. On z/OS, the DTLCFG DD statement in the DTLURDMO JCL points to the DBMOVER member. On i5/OS, the DBMOVER member is the CFG file in the PowerExchange *datalib* library. For more information, see "SECURITY Statement" in the *PowerExchange Reference Manual*.

Note: On z/OS, a valid RACF passphrase can be up to 100 characters in length. PowerExchange truncates passphrases longer than 100 characters when passing them to RACF for validation.

Do not also specify the EPWD statement.

REPLACE Statement

The REPLACE statement causes any existing maps or registrations at the target destination to be overwritten by those copied. You can use this statement to change map or registration attributes by copying from and to the same location. To ensure that you do not inadvertently lose existing map or registration information, use this statement with caution.

The REPLACE statement has no operands.

This statement is optional.

SOURCE Statement

The SOURCE statement specifies the source node that contains the maps and registrations. This node must specified in a NODE statement in the DBMOVER configuration file.

Use the following syntax:

```
SOURCE source_node;
```

This statement is optional. The default value is local.

TARGET Statement

The TARGET statement specifies the target node to which you want to copy the maps and registrations. This node must be specified in a NODE statement in the DBMOVER configuration file.

Use the following syntax:

```
TARGET target_node;
```

This statement is optional on Linux, UNIX, and Windows. The default value is local. You can specify the same node name in the SOURCE and TARGET statements.

Caution: On z/OS systems, always specify a target node in the TARGET statement so that the PowerExchange Listener writes to the target files. If you use the default value of local, intermittent file contention might occur because the DTLURDMO job might try to write to the file when the PowerExchange Listener already has the file in write mode. If the PowerExchange Listener always writes to the target files, no such contention can occur.

TARGETEPWD Statement

The TARGETEPWD statement specifies an encrypted password for the user ID specified in the USER statement. The encrypted password and user ID are used to access maps and registrations on the target system.

If the maps and registrations are on an i5/OS or z/OS system, you can enter an encrypted PowerExchange passphrase instead of an encrypted password. Ensure that the passphrase you encrypt does not contain invalid characters, such as double-quotation marks, single quotation marks, or currency symbols.

Use the following syntax:

```
TARGETEPWD encrypted_password_or_passphrase;
```

Note: To encrypt a password or passphrase, you can use the **File** > **Encrypt** command in the PowerExchange Navigator.

If the DTLURDMO definition file includes the TARGETUSER statement but does not include either the TARGETEPWD or TARGETPWD statement, the EPWD or PWD statement specifies the password on the target system.

Do not also specify the TARGETPWD statement.

TARGETPWD Statement

The TARGETPWD statement specifies a password for the user ID in the TARGETUSER statement. The password and user ID are used to access maps and registrations on the target system.

If the maps and registrations are on an i5/OS or z/OS system, you can enter a valid PowerExchange passphrase instead of a password.

Use the following syntax:

```
TARGETPWD target_password_or_passphrase;
```

Use a PowerExchange passphrase for enhanced security. An i5/OS passphrase can be from 9 to 31 characters in length. A z/OS passphrase can be from 9 to 128 characters in length. A passphrase can contain the following characters:

- Uppercase and lowercase letters
- The numbers 0 to 9
- Spaces
- The following special characters:

  ```
  ' - ; # \ , . / ! % & * ( ) _ + { } : @ | < > ?
  ```

  Note: The first character is an apostrophe.

Passphrases cannot include single quotation marks (‘), double quotation marks (“), or currency symbols.

If a passphrase contains spaces, you must enclose it with double-quotation marks ("), for example, `"This is a passphrase"`. If a passphrase contains special characters, you must enclose it with triple double-quotation characters (`"""`), for example, `"""This passphrase contains special characters ! % & *."""`. If a passphrase contains only alphanumeric characters without spaces, you can enter it without delimiters.

To use passphrases, ensure that the PowerExchange Listener runs with a security setting of SECURITY=(1,N) or higher in the DBMOVER configuration member. On z/OS, the DTLCFG DD statement in the DTLURDMO JCL points to the DBMOVER member. On i5/OS, the DBMOVER member is in the CFG file in the PowerExchange *datalib* library. For more information, see "SECURITY Statement" in the *PowerExchange Reference Manual*.

Note: On z/OS, a valid RACF passphrase can be up to 100 characters in length. PowerExchange truncates passphrases longer than 100 characters when passing them to RACF for validation.

If the DTLURDMO definition file includes the TARGETUSER statement but does not include either the TARGETEPWD or TARGETPWD statement, the EPWD or PWD statement specifies the password on the target system.

Do not also specify the TARGETEPWD statement.

TARGETUSER Statement

The TARGETUSER statement specifies the user ID for accessing maps and registrations on the target system.

Use the following syntax:

```
TARGETUSER user_ID;
```

This statement is optional. If it is not specified, the USER statement specifies the user ID for both the source and target systems.

USER Statement

The USER statement specifies a user ID for accessing maps and registrations on the source system.

If you have enabled PowerExchange LDAP user authentication on a supported Linux, UNIX, or Windows source system, the user name is the enterprise user name. For more information, see the *PowerExchange Reference Manual*.

If the DTLURDMO definition file does not include a TARGETUSER statement, the USER statement also specifies the user ID on the target system.

Use the following syntax:

```
USER user_ID;
```

This statement is required. You must also specify either the PWD or EPWD statement.

VALIDATE Statement

The VALIDATE statement instructs DTLURDMO to run in test mode, without writing the copied maps or registrations to the target destination. Used in conjunction with the DETAIL statement, VALIDATE enables copy scenarios to be modeled without committing any changes.

The VALIDATE statement has no operands.

This statement is optional.

Note: When you run DTLURDMO in test mode, the utility does not validate target connection information. For example, DTLURDMO does not test for valid node names, version mismatches, or target object authority.

DM_COPY Statement

The DM_COPY statement copies data maps from the source to the target system. This statement has no operands but can be followed by other statements that filter the selected objects, rename objects, or change object attributes.

The following table summarizes the optional statements that can follow the DM_COPY statement:

Optional Statement	Parameters
[EXCLUDE]	[AM=*access_method*] [MAP=*map_name*] [SCHEMA=*schema_name*]
[MODIFY]	AM=*access_method* [DB2INSTANCE=*db2_instance*] [DB2TABLE=*db2_table_name*] [DBD=*dbd_name*] [DBID=*database_ID*] [DBNAME=*database_name*] [DDSNODENAME=*ddsnode_name*] [DICTNAME=*dictionary_name*] [FILEID=*file_ID*] [FN=*file_name*] [IMSID=*ims_ID*] [IMSTYPE=*imstype*] [MUFNAME=*muf_name*] [PCB=*pcb_name*] [PCBNUM=pcb_*number*] [PROGNAME=*program_name*] [PSB=*psb_name*] [SUBSCHEMA=*subschema_name*]
[RENAME]	[DBD=(*old_dbd_name*,*new_dbd_name*)] [IDMS_PAGEGROUP_RADIX=(*old_page_group*,*new_page_group*,*old_radix*,*new_radix*)] [MAP=(*old_map_name*,*new_map_name*)] [SCHEMA=(*old_schema_name*,*new_schema_name*)] [TABLE=(*old_table_name*,*new_table_name*)]
[SELECT]	[AM=*access_method*] [MAP=*map_name*] [SCHEMA=*schema_name*]

DM_COPY EXCLUDE Statement

The EXCLUDE statement specifies filter criteria for data maps to be explicitly excluded from the copying process.

The EXCLUDE statement is optional. By default, no items are excluded.

The EXCLUDE statement has the following parameters:

AM=*access_method*

Specifies the access method of the data maps to exclude.

MAP=*map_name*

Specifies a map name to exclude.

SCHEMA=*schema_name*

Specifies a schema name to exclude.

The parameters for the EXCLUDE statement are the same as those for the SELECT statement.

Related Topics:

- "DM_COPY SELECT Statement" on page 158

DM_COPY MODIFY Statement

The MODIFY statement modifies attributes of the copied data maps.

The AM= parameter is required with the MODIFY statement. Depending on the value that you specify for the AM= parameter, you can include additional parameters.

The MODIFY statement has the following parameters:

AM=*access_method*

Required. Specifies the access method to which the MODIFY statement applies.

The following table shows the additional parameters that are available for each access method:

Access Method	Available Parameters
ADABAS	DBID, FILEID
DB2	DB2TABLE, DB2INSTANCE
DB2UDB	DB2TABLE, DB2INSTANCE
DB2UNLD	FN, DB2TABLE, DB2INSTANCE
DCOM	DBID, DBNAME, MUFNAME
DL1	DBD, IMSID, IMSTYPE, PCBNUM
ESDS	FN
IDMS	SUBSCHEMA, DBNAME, PROGNAME, DICTNAME, DDSNODENAME
IMS	None
KSDS	FN
MSSQL	None
ODBA	DBD, IMSID, IMSTYPE, PSB, PCB
Oracle	None
RRDS	FN
SEQ	FN

DB2TABLE=*db2_schema.db2_table_name*

Modifies the DB2 schema or table name for the table mapped by the data map. For example:

```
DB2TABLE=DSN8910.EMP
```

DB2INSTANCE=*db2_instance*

Modifies the DB2 database name or subsystem ID for the data mapped by the data map. For example:

```
DB2INSTANCE=sample
```

DBD=*dbd_name*

Modifies the DBD name for the data mapped by the data map. For example:

```
DBD=PROD001
```

The DBD parameter of the DM_COPY MODIFY statement is deprecated but supported for backward compatibility. Instead, use the DM_COPY RENAME statement with the DBD parameter to rename a DBD in a data map.

DBID=*database_ID*

Specifies a new database ID to be used when reading the database.

DBNAME=*database_name*

Specifies a new database name to be used when reading the database.

DDSNODENAME=*ddsnode_name*

Specifies a new DDSNODE name to be used when reading the IDMS database.

DICTNAME=*dictionary_name*

Specifies a new dictionary name to be used when reading the IDMS database.

FILEID=*file_ID*

Specifies a new ADABAS file ID for the data mapped by the data map.

FN=*filename*

Specifies a new name for the data file associated with the data map. The file name must be a valid file name on the target system. The following examples specify a new data file name for z/OS, VSAM, and Windows, respectively:

```
FN='DATA01.SEQ.FILE'
FN='SYS01.KSDS.DATA.FILE'
FN=c:\myfolder\myfile.txt
```

IMSID=*ims_ID*

Specifies a new IMS system ID for the data mapped by the data map. For example:

```
IMSID=IMS7
```

IMSTYPE=*ims_type*

Specifies a new IMS type for the data mapped by the data map. Specify one of the following values:

- DEDB
- GSAM
- HDAM
- HIDAM
- HSAM

- HISAM
- MSDB
- PHDAM
- PHIDAM
- PSINDEX
- SHISAM

MUFNAME=*muf_name*

Specifies a new MUF name to be used when reading the Datacom database.

PCB=*pcb_name*

Specifies a new PCB name for the data mapped by the data map. For example:

```
PCB=PCB020
```

PCBNUM=*pcb_number*

Specifies a new PCB number for the data mapped by the data map.

PROGNAME=*program_name*

Specifies a new program name to be used when reading the IDMS database.

PSB=*psb_name*

Specifies a new PSB for the data mapped by the data map. For example:

```
PSB=DTL003
```

SUBSCHEMA=*subschema_name*

Specifies a new subschema name to be used when reading the IDMS database.

DM_COPY RENAME Statement

The RENAME statement specifies which elements of the data map name are renamed on the target system. For most parameters, the first operand represents the item or items being renamed, and the second operand represents the new name. For example:

```
MAP=(map01,map02)
```

The first operand can be any of the following:

- The full name of the item being renamed
- A partial name with a wildcard (*)
- A wildcard (*) only

In each case, all items matching the specified name or pattern are renamed to the value of the second operand.

DBD=(*old_dbd_name*,*new_dbd_name*)

For IMS DL1 or ODBA data maps, renames the DBDs in the data map that match the name or pattern in the first operand to the specified name.

In the following example, all DBDs named **dbd01** are renamed to **dbd02**:

```
DBD=(dbd01,dbd02)
```

IDMS_PAGEGROUP_RADIX=(*old_page_group*,*new_page_group*,*old_radix*,*new_radix*)

For IDMS data maps, changes the page group or radix value. This parameter enables you to migrate registrations and extraction maps to environments with different values for the page group and radix database settings.

For *old_page_group* and *new_page_group*, specify a value from 0 to 32767.

For *old_radix* and *new_radix*, specify a value are 2 to 12.

MAP=(*old_map_name*,*new_map_name*)

Renames the data maps that match the name or pattern in the first operand to the specified name.

In the following example, all data maps named **map01** are renamed to **map02**:

```
MAP=(map01,map02)
```

In the following example, all data maps are renamed to **newmap**:

```
MAP=(*,newmap)
```

In the following example, all data map names ending in **tmp** are renamed to **fixed**:

```
MAP=(*tmp,fixed)
```

SCHEMA=(*old_schema_name*,*new_schema_name*)

Renames the data map schemas that match the name or pattern in the first operand to the specified name.

In the following example, all schemas named **test** are renamed to **prod**:

```
SCHEMA=(test,prod)
```

In the following example, all schemas are renamed to **newprod**:

```
SCHEMA=(*,newprod)
```

In the following example, all schemas ending in **tmp** are renamed to **fixed**:

```
SCHEMA=(*tmp,fixed)
```

TABLE=(*old_table_name*,*new_table_name*)

Renames the tables that match the name or pattern in the first operand to the specified name.

In the following example, all tables named **testtab01** are renamed to **prodtab01**:

```
TABLE=(testtab01,prodtab01)
```

In the following example, all tables are renamed to **newtable**:

```
TABLE=(*,newtable)
```

In the following example, all tables ending in **01** are renamed to **fixed**:

```
TABLE=(*01,fixed)
```

DM_COPY SELECT Statement

The SELECT statement specifies filter criteria for the data maps to be copied.

The SELECT statement has the following parameters:

AM=*access_method*

Specifies the access method of the data map.

The following table lists and describes the values for *access_method*:

Access Method	Data Source
ADABAS	Adabas
DB2	DB2 for z/OS
DB2UDB	DB2 for Linux, UNIX, and Windows
DB2UNLD	DB2 Unload
DCOM	Datacom
DL1	DL/1 batch for IMS
ESDS	VSAM ESDS
IDMS	IDMS
IMS	IMS
KSDS	VSAM KSDS
MSSQL	Microsoft SQL Server
ODBA	IMS ODBA
Oracle	Oracle
RRDS	VSAM RRDS
SEQ	Sequential data set

MAP=*map_name*

Specifies a map name to select, which is any of the following:

- The full name of a data map
- A partial name with a wildcard (*)
- A wildcard (*) only

The following example specifies the map named **sample**:

```
MAP=sample
```

The following example specifies maps beginning with **sam**:

```
MAP=sam*
```

The following example specifies all data maps:

```
MAP=*
```

The default is *.

SCHEMA=*schema_name*

Specifies a schema name to select, which is one of the following:

- The full name of a data map schema
- A partial name with a wildcard (*)
- A wildcard (*) only

The following example specifies the schema **db2map**:

```
SCHEMA=db2map
```

The following example specifies schemas beginning with **prod**:

```
SCHEMA=prod*
```

The following example specifies all schemas:

```
SCHEMA=*
```

The default is *.

REG_COPY Statement

Use the REG_COPY statement to copy capture registrations. This statement has no operands but can be followed by other statements that filter the selected objects, rename objects, change object attributes, or enable optional functions.

The following table summarizes the optional statements that can follow the REG_COPY statement:

Optional Statement	Parameters
[CHECKXREF]	None
[CREATEXMAPS]	[LOC={SOURCE|TARGET} [OUTPUT=*alternative_pathname/data_set*]
[EXCLUDE]	[DBID=*database_instance*] [DBTYPE=*database_type*] [REG_NAME=*registration_name*]
[KEEPREGTAG]	None
[MODIFY]	[CONDENSE={FULL|PART|NONE}] [DBID=*database_ID*] [DBNAME=*database_name*] [FILEID=*file_ID*] [FN=*file_name*] [MSSOPTS=(DBSERVER=*db_server*,DBNAME=*database_name*)] [MUFNAME=*muf_name*] [NEW_DBID=*database_instance*] [SUBSCHEMA=*subschema_name*]
[RELATED]	BULK

Optional Statement	Parameters
[RENAME]	[BULKMAP=(*old_map_name*,*new_map_name*)] [BULKSCHEMA=(*old_schema_name*,*new_schema_name*)] [BULKTABLE=(*old_table_name*,*new_table_name*)] [DBD=*(old_dbd_name*,*new_dbd_name)* [IDMS_PAGEGROUP_RADIX=*(old_page_group*,*new_page_group*,*old_radix*,*new_radix*)] [IMSMAP=*(old_map_name*,*new_map_name)]* [IMSSCHEMA=*(old_schema_name*, *new_schema_name)]* [NRDB_DM_TABLE=(*old_map_name*,*new_map_name*,*old_table_name*,*new_table_name*)] [REG_NAME=*old_registration_name*,*new_registration_name)]* [SCHEMA=(*old_schema_name*,*new_schema_name*)] [TABLE=(*old_table_name*,*new_table_name*)]
[SELECT]	[DBID=*database_instance*] [DBTYPE=*database_type*] [REG_NAME=*registration_name*]

Notes:

- Do not use the OUTPUT statement with the REG_COPY statement if you also use the CREATEXMAPS statement.
- To use REG_COPY to copy registrations to an alternative CCT data set and create extraction maps, you must connect to the target by using a PowerExchange Listener, rather than using the default of local in the TARGET statement.
- If you include any of the following statements or combinations of statements and options after the REG_COPY statement, DTLURDMO reads the IMS data map from the target system:
  - RENAME IMSSCHEMA
  - RENAME IMSMAP
  - CHECKXREF
- If you copy a registration that has been suspended with the PWXUCREG utility, the following processing occurs:
  - If the current registration status is Suspended, the status is reset to Active.
  - The activation and suspension timestamps that define the suspension window are cleared.

REG_COPY CHECKXREF Statement

For IMS data sources, the REG_COPY CHECKXREF statement forces the DTLURDMO utility to load the corresponding data map on the target system and to update the registrations with the database organization information from the DBD that is specified in the data map when copying registrations. Use the CHECKKXREF parameter if you change the DBD to specify a different database organization and then need to the change the registrations.

This statement has no parameters.

Important: When you run the DTLURDMO utility with a REG_COPY CHECKXREF statement in the SYSIN and a TARGET value of LOCAL, ensure that the IMS RESLIB data set, or whichever library contains module DFSVC000 in your installation, is specified in the STEPLIB concatenation in the DTLURDMO JCL. If the TARGET value specifies a PowerExchange Listener node, ensure that the IMS RESLIB data set, or whichever library contains module DFSVC000, is specified in the STEPLIB concatenation in the target Listener JCL.

REG_COPY CREATEXMAPS Statement

The CREATEXMAPS statement creates an extraction map on the target system.

Use the following syntax:

```
CREATEXMAPS
[LOC={SOURCE|TARGET}]
[OUTPUT=alternative_pathname|data_set]
;
```

The REG_COPY statement has the following optional parameters:

LOC={SOURCE|TARGET}

Specifies whether the data map used to create the extraction map is loaded from the source or target location. Default is SOURCE.

OUTPUT=*alternative_pathname*|*data_set*

The extraction map is written to the alternative location. This function is analogous to that provided by the OUTPUT statement.

REG_COPY EXCLUDE Statement

The EXCLUDE statement specifies filter criteria for registrations to be explicitly excluded from the copying process.

The EXCLUDE statement is optional. By default, no items are excluded.

The EXCLUDE statement has the following parameters:

REG_NAME=*registration_name*

Specifies a registration name to select. This name is the one that was entered when the registration was created.

DBID=*database_instance*

Specifies the database instance of the registration. For example, depending on the source, the database instance might represent the subsystem ID or database name.

DBTYPE=*database_type*

Specifies the data type being captured.

The parameters for the EXCLUDE statement are the same as those for the SELECT statement.

Related Topics:

- "REG_COPY SELECT Statement" on page 167

REG_COPY KEEPREGTAG Statement

The KEEPREGTAG statement retains the original registration tag from the registration being copied when generating the extraction map. This statement is valid only with the CREATEXMAPS statement.

Do not use the KEEPREGTAG statement with the following REG_COPY statements:

- CHECKXREF
- MODIFY MUF_NAME
- MODIFY NEW_DBID
- RENAME DBD

- RENAME IMSMAP
- RENAME IMSSCHEMA
- RENAME REG_NAME

These statements might cause a different registration tag to be generated for the copied registration and the extraction map.

REG_COPY MODIFY Statement

The MODIFY statement modifies the specified attributes of the copied registrations.

This statement can include the following parameters:

CONDENSE={FULL|PART|NONE}

Specifies condense options for the captured data on the target system.

DBID=*database_ID*

For Adabas and Datacom databases, specifies a new database identifier.

DBNAME=*database_name*

For IDMS databases, specifies a new database name for the registration.

FN=*file_name*

Specifies the file name associated with the registration. For example:

```
FN=NEW.KSDS.FILE001
```

MSSOPTS=(DBSERVER=*db_server*,DBNAME=*database_name*)

Specifies the following SQL Server options for the registration:

- DBSERVER=*db_server* is the database server ID for the registration.
- DBNAME=*database_name* is the database name for the registration.

MUFNAME=*muf_name*

For Datacom databases, specifies a new MUF name for the registration. Use this parameter to specify a new database instance for the registration in the same way you can use the NEW_DBID parameter for other database types.

NEW_DBID=*database_ID*

Specifies a new database instance for the registration. For most data sources, this database instance value is a subsystem ID or database name.

For Microsoft SQL Server sources, optionally specifies a unique user-defined instance identifier for the database server and database name combination defined in the MSSOPTS parameter. Maximum length is seven characters. This instance identifier identifies a set of registrations for the publication database on the target system. If you also define the CREATEXMAPS statement in the registration copy syntax, the instance identifier is incorporated into the names of the extraction maps that are generated for the copied registrations on the target. If you use the PowerExchange Logger for Linux, UNIX, and Windows, ensure that the instance identifier matches the DBID parameter value in the Logger configuration file on the target. If you do not enter a NEW_DBID value, PowerExchange generates a unique instance identifier that is composed of all or part of the publication database name followed by a 3-digit number if a number is required to make the identifier unique.

Tip: If you are migrating from one SQL Server environment to another, you can enter an instance identifier that matches the instance identifier in the source system. In this manner, you can avoid using a

generated instance identifier and having to update the extraction map names in PowerCenter workflows and edit the PowerExchange Logger DBID parameter value on the target. In this case, ensure that the DBMOVER configuration files in the source and target environments define unique paths in the CAP_PATH and CAPT_XTRA statements.

SUBSCHEMA=*subschema_name*

For IDMS databases, specifies a new subschema for the registration.

REG_COPY RELATED BULK Statement

The RELATED BULK statement merges the extraction map with an existing bulk data map on the target system.

This statement is valid only in conjunction with the CREATEXMAPS statement for DB2 for z/OS or DB2 for Linux, UNIX, and Windows registrations.

Use the following syntax:

```
RELATED BULK;
```

The name of the bulk data map that DTLURDMO looks for on the target system depends on whether the original extraction map has been merged with a data map on the source system:

- If the original extraction map associated with the registration on the source system was merged with a bulk data map, DTLURDMO uses the same data map and table name when merging the generated extraction map on the target system.
- If a bulk data map was not merged with the original extraction map on the source system, or an extraction map does not exist, a bulk data map name of the following form is generated for the merge:

  table_name.registration_name_table_name

If DTLURDMO does not find the bulk data map on the target system, DTLURDMO reports the error and continues.

You can use the RENAME statement to modify the name of the generated extraction map or the name of the bulk data map to be merged with it.

REG_COPY RENAME Statement

The RENAME statement renames elements of the copied capture registration on the target system or identifies an existing bulk data map on the target system that is named differently from the default.

- For the DBD, IMSMAP, IMSSCHEMA, REG_NAME, SCHEMA, and TABLE parameters, the RENAME statement specifies the new name of the registration element on the target system.
- For the BULKSCHEMA, BULKMAP, and BULKTABLE parameters, the RENAME statement identifies the bulk data map on the target system to be merged with the newly generated extraction map. Use these parameters if the bulk data map on the target system is named differently from the default.

  These parameters are available only on DB2 for z/OS or DB2 for Linux, UNIX, and Windows systems.

For most RENAME parameters, the first operand represents the item or items being renamed, and the second operand represents the new name. The first operand can be any of the following:

- The full name of an item
- A partial name with a wildcard (*)
- A wildcard (*) only

The RENAME statement has the following parameters:

BULKMAP=(*old_map_name*,*new_map_name*)

Specifies a new map name to use in locating the bulk data map on the target system to merge with the copied extraction map. The BULKMAP parameter of the RENAME statement is valid only in conjunction with the RELATED BULK statement.

In the following example, all maps named **capture1** are renamed to **capture2**:

```
BULKMAP=(capture1,capture2)
```

In the following example, all maps are renamed to **newmap**:

```
BULKMAP=(*,newmap)
```

In the following example, all map names ending in **01** are renamed to fixed:

```
BULKMAP=(*01,fixed)
```

BULKSCHEMA=(*old_schema_name*,*new_schema_name*)

Specifies a new schema name to use in locating the bulk data map on the target system to merge with the copied extraction map. The BULKSCHEMA parameter of the RENAME statement is valid only in conjunction with the RELATED BULK statement.

In the following example, all bulk schemas named **test** are renamed to **prod**:

```
BULKSCHEMA=(test,prod)
```

In the following example, all bulk schemas are renamed to **newprod:**

```
BULKSCHEMA=(*,newprod)
```

In the following example, all schemas ending in **tmp** are renamed to **fixed**:

```
BULKSCHEMA=(*tmp,fixed)
```

BULKTABLE=(*old_table_name*,*new_table_name*)

Specifies a new table name to use in locating the bulk data map on the target system to merge with the copied extraction map. The BULKTABLE parameter of the RENAME statement is valid only in conjunction with the RELATED BULK statement.

In this example, all table names **testtab01** are renamed to **prodtab01**:

```
BULKTABLE=(testtab01t,prodtab01)
```

In this example, all tables are renamed to **newtable**:

```
BULKTABLE=(*,newtable)
```

In this example, all tables ending in **01** are renamed to **fixed**:

```
BULKTABLE=(*01,fixed)
```

DBD=(*old_dbd_name*,*new_dbd_name*)

For IMS data sources, specifies a new DBD name for the capture registration. DTLURDMO loads the corresponding DBD on the target system and updates the registration with the database organization.

The DBD specified in a RENAME statement takes precedence over the DBD name derived from a map specified by IMSSCHEMA or IMSMAP.

IDMS_PAGEGROUP_RADIX=(*old_page_group*,*new_page_group*,*old_radix*,*new_radix*)

For IDMS data sources, changes the page group or radix value. This parameter enables you to migrate registrations and extraction maps to environments with different values for the page group and radix database settings.

For *old_page_group* and *new_page_group*, specify a value from 0 to 32767.

For *old_radix* and *new_radix*, specify a value from 2 to 12.

Note: You can create an extraction map on the target system with renamed page group and radix values by using the REG_COPY, CREATEXMAPS LOC=TARGET, and RENAME IDMS_PAGEGROUP_RADIX statements. Before you issue these statements, you must copy the data map to the target platform by using the DM_COPY statement. If the data map is not correct on the target platform, the results of the REG_COPY and accompanying optional statements are unpredictable.

IMSMAP=(*old_map_name,new_map_name*)

Renames IMS data maps that match the name or the pattern in the old data map name to the new data map name. The utility loads the specified data map from the target system and updates the registration with database organization using the DBD named in the data map.

In the following example, all schemas named **test** are renamed to **prod**:

```
IMSMAP=(test,prod)
```

IMSSCHEMA=(*old_schema_name,new_schema_name*)

Renames IMS schemas that match the name or pattern in the old schema name to the new schema name. The utility loads the specified data map from the target system and uses the DBD named in the data map to update the database organization information in the registration.

The new schema name must match the name of the schema for an existing bulk data map on the target system.

In the following example, all schemas named **test** are renamed to **prod**:

```
IMSSCHEMA=(test,prod)
```

NRDB_DM_TABLE=(*old_map_name*,*new_map_name*,*old_table_name*,*new_table_name*)

For IDMS data sources, identifies the old and new data map and table names. This parameter enables the registration to link to a data map that has a new identity in the new environment.

NRDB_DM_TABLE is required only if the data map contains information that is necessary for CDC extractions. Specifically, it is required to propagate changes to the page group or radix value to the extraction map.

The following rules apply:

- Use the following format for *old_map_name* and *new_map_name*:

  ```
  schema_name.data_map_name
  ```

  schema_name, *data_map_name*, or both can consist in part or full of wildcards, as in the following examples:

  ```
  *.test_map
  test_schema.*
  test*.*
  ```

- If you omit *old_map_name* and *new_map_name*, include commas as placeholders, as follows:

  ```
  NRDB_DM_TABLE=(,,old_table_name,new_table_name)
  ```

- You can omit *old_table_name* and *new_table_name*. If you include them, specify the complete table name, without wildcards.

REG_NAME=(*old_registration_name,new_registration_name*)

Renames the registration. Because the registration name is used to create the registration tag, this option overrides the KEEPREGTAG option.

SCHEMA=*old_schema_name,new_schema_name*

Renames schemas that match the name or the pattern in the old schema name to the new schema name.

In the following example, all schemas named **test** are renamed to **prod**:

```
SCHEMA=(test,prod)
```

In the following example, all schemas are renamed to **newprod**:

```
SCHEMA=(*,newprod)
```

In the following example, all schemas ending in **tmp** are renamed to **fixed**:

```
SCHEMA=(*tmp,fixed)
```

TABLE=*old_table_name,old_table_name*

Renames tables that match the name or the pattern in the old table name to the new table name.

In the following example, all tables named **testtab01** are renamed to **prodtab01**:

```
TABLE=(testtab01,prodtab01)
```

In the following example, all tables are renamed to **newtable**:

```
TABLE=(*,newtable)
```

In the following example, all tables ending in **01** are renamed to **fixed**:

```
TABLE=(*01,fixed)
```

Related Topics:

- "REG_COPY RELATED BULK Statement" on page 164

REG_COPY SELECT Statement

The SELECT statement specifies filter criteria for the registrations to be copied.

The SELECT statement has the following parameters:

DBID=*database_instance*

Specifies the database instance of the registration. For example, depending on the source, the database instance might represent the subsystem ID or database name.

You can specify any of the following:

- The full name of a data map schema
- A partial name with a wildcard (*)
- A wildcard (*) only

The following example specifies the database instance **sample**:

```
DBID=sample
```

The following example specifies database instances beginning with **sam**:

```
DBID=sam*
```

The following example specifies all database instances:

```
DBID=*
```

The default is *.

DBTYPE=*database_type*

Specifies the data type being captured.

The following table lists and describes the values for *database_type*:

Database Type	Description
ADA	Adabas
AS4	DB2 i5/OS
DB2	DB2 for z/OS
DCM	Datacom
IDM	IDMS
IMS	IMS
MSS	Microsoft SQL Server
ORA	Oracle
UDB	DB2 for Linux, UNIX, and Windows
VSM	VSAM

REG_NAME=*registration_name*

Specifies a registration name to select. This name is the one that was entered when the registration was created.

You can specify any of the following:

- The full name of a data map schema
- A partial name with a wildcard (*)
- A wildcard (*) only

The following example specifies the registration **capture1**:

```
REG_NAME=capture1
```

The following example specifies registrations beginning with **prod**:

```
REG_NAME=prod*
```

The following example specifies all schemas:

```
REG_NAME=*
```

The default is *.

XM_COPY Statement

The XM_COPY statement copies extraction maps from the source system to the target system. This statement has no operands but can be followed by other statements that filter the selected objects, rename objects, or change object attributes.

The XM_COPY statement enables you to copy extraction maps without copying capture registrations. Typically, when you copy a capture registration, you copy the extraction map at the same time by including the CREATEXMAPS statement after the REG_COPY statement.

The following table summarizes the optional statements that can follow the XM_COPY statement:

Optional Statement	Parameters
[EXCLUDE]	[AM=*access_method*] [MAP=*map_name*] [SCHEMA=*schema_name*]
[MODIFY]	AM=*access_method* [DB2INSTANCE=*db2_instance*] [DB2TABLE=*db2_table_name*] [DBD=*dbd_name*] [DBID=*database_ID*] [DBNAME=*database_name*] [DDSNODENAME=*ddsnode_name*] [DICTNAME=*dictionary_name*] [FILEID=*file_ID*] [FN=*file_name*] [IMSID=*ims_ID*] [MUFNAME=*muf_name*] [PCB=*pcb_name*] [PROGNAME=*program_name*] [PSB=*psb_name*] [SUBSCHEMA=*subschema_name*]
[RENAME]	[IDMS_PAGEGROUP_RADIX=(*old_page_group*,*new_page_group*,*old_radix*,*new_radix*)] [MAP=(*old_map_name*,*new_map_name*)] [NRDB_DM_TABLE=(*old_map_name*,*new_map_name*,*old_table_name*,*new_table_name*)] [REG_NAME=(*old_registration_name*,*new_registration_name*,*new_version*)] [REGTAG=(*old_regtag*,*new_regtag*)] [SCHEMA=(*old_schema_name*,*new_schema_name*)] [TABLE=(*old_table_name*,*new_table_name*)]
[SELECT]	[AM=*access_method*] [MAP=*map_name*] [SCHEMA=*schema_name*]

XM_COPY EXCLUDE Statement

The EXCLUDE statement specifies filter criteria for extraction maps to be explicitly excluded from the copying process.

The EXCLUDE statement is optional. By default, no items are excluded.

The EXCLUDE statement has the following parameters:

AM=*access_method*

Specifies the access method of the extraction maps to exclude.

MAP=*map_name*

Specifies a map name to exclude.

SCHEMA=*schema_name*

Specifies a schema name to exclude.

The parameters for the EXCLUDE statement are the same as those for the SELECT statement.

XM_COPY MODIFY Statement

The MODIFY statement modifies various attributes of the copied extraction maps.

The AM= parameter is required with the MODIFY statement. Depending on the value that you specify for the AM= parameter, you can include additional parameters.

The MODIFY statement has the following parameters:

AM=*access_method*

Required. Specifies the access method to which the MODIFY statement applies.

The following table shows the additional parameters that are available for each access method:

Access Method	Available Parameters
ADABAS	DBID, FILEID
DB2	DB2TABLE, DB2INSTANCE
DB2UDB	DB2TABLE, DB2INSTANCE
DCOM	DBID, DBNAME, MUFNAME
DL1	DBD, IMSID
ESDS	FN
IDMS	SUBSCHEMA, DBNAME, PROGNAME, DICTNAME, DDSNODENAME
IMS	None
KSDS	FN
MSSQL	None
ODBA	DBD, IMSID, PSB, PCB
Oracle	None
RRDS	FN
SEQ	FN

DB2TABLE=*db2_schema.db2_table_name*

Modifies the DB2 schema or table name for the table mapped by the extraction map. For example:

```
DB2TABLE=DSN8910.EMP
```

DB2INSTANCE=*db2_instance*

Modifies the DB2 database name or subsystem ID for the data mapped by the extraction map. For example:

```
DB2INSTANCE=sample
```

DBD=*dbd_name*

Modifies the DBD name for the data mapped by the extraction map. For example:

```
DBD=PROD001
```

DBID=*database_ID*

Specifies a new database ID to be used when reading the database.

DBNAME=*database_name*

Specifies a new database name to be used when reading the database.

DDSNODENAME=*ddsnode_name*

Specifies a new DDSNODE name to be used when reading the IDMS database.

DICTNAME=*dictionary_name*

Specifies a new dictionary name to be used when reading the IDMS database.

FILEID=*file_ID*

Specifies a new ADABAS file ID for the data mapped by the extraction map.

FN=*filename*

Specifies a new name for the data file associated with the extraction map. The file name must be a valid file name on the target system. The following examples specify a new data file name for z/OS, VSAM, and Windows, respectively:

```
FN='DATA01.SEQ.FILE'
FN='SYS01.KSDS.DATA.FILE'
FN=c:\myfolder\myfile.txt
```

IMSID=*ims_ID*

Specifies a new IMS system ID for the data mapped by the extraction map. For example:

```
IMSID=IMS7
```

MUFNAME=*muf_name*

Specifies a new MUF name to be used when reading the Datacom database.

PROGNAME=*program_name*

Specifies a new program name to be used when reading the IDMS database.

PCB=*pcb_name*

Specifies a new PCB name for the data mapped by the extraction map. For example:

```
PCB=PCB020
```

PSB=*psb_name*

Specifies a new PSB for the data mapped by the extraction map. For example:

```
PSB=DTL003
```

SUBSCHEMA=*subschema_name*

Specifies a new subschema name to be used when reading the IDMS database.

XM_COPY RENAME Statement

The RENAME statement specifies which elements of the extraction map name are renamed on the target system. For most parameters, the first operand represents the item or items being renamed, and the second operand represents the new name. For example:

```
MAP=(map01,map02)
```

The first operand can be any of the following:

- The full name of a schema, extraction map, or table
- A partial name with a wildcard (*)
- A wildcard (*) only

In each case, all items matching the specified name or pattern are renamed to the value of the second operand.

IDMS_PAGEGROUP_RADIX=(*old_page_group*,*new_page_group*,*old_radix*,*new_radix*)

For IDMS data sources, changes the page group or radix value. This parameter enables you to migrate registrations and extraction maps to environments with different values for the page group and radix database settings.

For *old_page_group* and *new_page_group*, specify a value from 0 to 32767.

For *old_radix* and *new_radix*, specify a value from 2 to 12.

Note: You can create an extraction map on the target system with renamed page group and radix values by using the XM_COPY and RENAME IDMS_PAGEGROUP_RADIX statements. Before you issue these statements, you must copy the data map to the target platform by using the DM_COPY statement. If the data map is not correct on the target platform, the results of the XM_COPY and RENAME IDMS_PAGEGROUP_RADIX statements are unpredictable.

MAP=(*old_map_name*,*new_map_name*)

Renames the extraction maps that match the name or pattern in the first operand to the specified name.

In the following example, all extraction maps named **map01** are renamed to **map02**:

```
MAP=(map01,map02)
```

In the following example, all extraction maps are renamed to **newmap**:

```
MAP=(*,newmap)
```

In the following example, all extraction map names ending in **tmp** are renamed to **fixed**:

```
MAP=(*tmp,fixed)
```

NRDB_DM_TABLE=(*old_map_name*,*new_map_name*,*old_table_name*,*new_table_name*)

For IDMS data sources, identifies the old and new data map and table names. This parameter enables the registration to link to a data map that has a new identity in the new environment.

NRDB_DM_TABLE is required only if the data map contains information that is necessary for CDC extractions. Specifically, it is required to propagate changes to the page group or radix value to the extraction map.

The following rules apply:

- Use the following format for *old_map_name* and *new_map_name*:

  ```
  schema_name.data_map_name
  ```

  schema_name, *data_map_name*, or both can consist in part or full of wildcards, as in the following examples:

  ```
  *.test_map
  test_schema.*
  test*.*
  ```

- If you omit *old_map_name* and *new_map_name*, include commas as placeholders, as follows:

  ```
  NRDB_DM_TABLE=(,,old_table_name,new_table_name)
  ```

- You can omit *old_table_name* and *new_table_name*. If you include them, specify the complete table name, without wildcards.

REG_NAME=(*old_registration_name*,*new_registration_name*,*new_version*)

Renames the registration and, optionally, the version. Because the registration name is used to create the registration tag, this option overrides the KEEPREGTAG option. This parameter provides a documented link to the new registration.

REGTAG=(*old_regtag*,*new_regtag*)

Renames the registration tags that match the name or pattern in the first operand to the specified name.

SCHEMA=(*old_schema_name*,*new_schema_name*)

Renames the schemas that match the name or pattern in the first operand to the specified name.

In the following example, all schemas named **test** are renamed to **prod**:

```
SCHEMA=(test,prod)
```

In the following example, all schemas are renamed to **newprod**:

```
SCHEMA=(*,newprod)
```

In the following example, all schemas ending in **tmp** are renamed to **fixed**:

```
SCHEMA=(*tmp,fixed)
```

TABLE=(*old_table_name*,*old_table_name*)

Renames the tables that match the name or pattern in the first operand to the specified name.

In the following example, all tables named **testtab01** are renamed to **prodtab01**:

```
TABLE=(testtab01,prodtab01)
```

In the following example, all tables are renamed to **newtable**:

```
TABLE=(*,newtable)
```

In the following example, all tables ending in **01** are renamed to **fixed**:

```
TABLE=(*01,fixed)
```

XM_COPY SELECT Statement

The SELECT statement specifies filter criteria for the data maps to be copied.

The SELECT statement has the following parameters:

AM=*access_method*

Specifies the access method of the extraction map.

The following table lists and describes the values for *access_method*:

Access Method	Data Source
ADABAS	Adabas
DB2	DB2 for z/OS
DB2UDB	DB2 for Linux, UNIX, and Windows

Access Method	Data Source
DCOM	Datacom
DL1	DL/1 batch for IMS
ESDS	VSAM ESDS
IDMS	IDMS
IMS	IMS
KSDS	VSAM KSDS
MSSQL	Microsoft SQL Server
ODBA	IMS ODBA
Oracle	Oracle
RRDS	VSAM RRDS
SEQ	Sequential data set

MAP=*map_name*

Specifies a map name to select, which is any of the following:

- The full name of a extraction map
- A partial name with a wildcard (*)
- A wildcard (*) only

The following example specifies the map named **sample**:

```
MAP=sample
```

The following example specifies maps beginning with **sam**:

```
MAP=sam*
```

The following example specifies all extraction maps:

```
MAP=*
```

The default is *.

SCHEMA=*schema_name*

Specifies a schema name to select, which is one of the following:

- The full name of a extraction map schema
- A partial name with a wildcard (*)
- A wildcard (*) only

The following example specifies the schema **db2map**:

```
SCHEMA=db2map
```

The following example specifies schemas beginning with **prod**:

```
SCHEMA=prod*
```

The following example specifies all schemas:

```
SCHEMA=*
```

The default is *.

Scope of Operands

Within the scope of a copy statement (DM_COPY, REG_COPY, or XM_COPY), DTLURDMO allows multiple occurrences of each of the following statements:

- EXCLUDE
- MODIFY
- RENAME
- SELECT

Multiple occurrences of these statements are supported to maintain backward compatbility and to provide flexibility when migrating a large number of objects. However, in most cases it is simpler and clearer to include multiple copy statements instead. This way, each copy statement is followed by at most one EXCLUDE, MODIFY, RENAME, and SELECT statement.

If you include multiple occurrences of the EXCLUDE MODIFY, RENAME, or SELECT statements, include a VALIDATE statement to first run DTLURDMO in test mode and verify the operation of these statements. In addition, if you include multiple occurrences of the MODIFY or SELECT statements within the scope of a copy statement, PowerExchange issues a warning message to indicate that the outcome of these multiple statements might be unpredictable.

Behavior of EXCLUDE, MODIFY, RENAME, and SELECT Statements

The following rules summarize the behavior of the EXCLUDE, MODIFY, RENAME, and SELECT statements within the scope of a copy statement:

- To determine which objects to exclude, DTLURDMO performs a logical OR operation on all the EXCLUDE statements.
- To determine which objects to select, DTLURDMO performs a logical OR operation on all the SELECT statements.
- The EXCLUDE and SELECT statements determine the scope of each RENAME statement. Each RENAME statement renames the specified objects that are selected by the SELECT statements and not excluded by the EXCLUDE statements.
- The EXCLUDE and SELECT statements determine the scope of each MODIFY statement. The scope of each MODIFY statement is further restricted by the specified access method. Multiple MODIFY statements that specify different access methods are logically independent (OR'ed).
- If multiple MODIFY statements specify the same access method, DTLURDMO ignores all but the first statement.
- Except in the case of multiple MODIFY statements that specify the same access method, the order of statements does not matter.

Example Using Multiple SELECT and MODIFY Statements

Suppose you need to copy several data maps that were defined with an access method of SEQ, and you need to modify the file name attribute in each data map.

The following statements might appear to be correct, but they do *not* produce the desired result:

```
DM_COPY;
SELECT MAP=fbti SCHEMA=vsam ;
MODIFY AM=KSDS FN=FPRSV.PAS.PSCODV1;
SELECT MAP=scpd SCHEMA= flatfile ;
MODIFY AM=SEQ FN=FPRS.PAS.D416.PSW.PLNXTR;
SELECT MAP=sczp SCHEMA= flatfile ;
MODIFY AM=SEQ FN=FPRS.PAS.D416.PSW.PLNSTAT;
```

In this example, the scope of the MODIFY statements is determined by all the SELECT statements within the scope of the DM_COPY statement, not just the immediately preceding SELECT statement. The following statement is disregarded, because it applies to the same access method and set of selected data maps (as determined by the SELECT statements) as a previous MODIFY statement within the scope of a single DM_COPY command:

```
 MODIFY AM=SEQ FN=FPRS.PAS.D416.PSW.PLNSTAT
```

Instead, include multiple DM_COPY commands, with each DM_COPY command followed by a single SELECT command and a single MODIFY command:

```
DM_COPY;
SELECT MAP=fbti SCHEMA=vsam ;
MODIFY AM=KSDS FN=FPRSV.PAS.PSCODV1;

DM_COPY;
SELECT MAP=scpd SCHEMA= flatfile ;
MODIFY AM=SEQ FN=FPRS.PAS.D416.PSW.PLNXTR;

DM_COPY;
SELECT MAP=sczp SCHEMA= flatfile ;
MODIFY AM=SEQ FN=FPRS.PAS.D416.PSW.PLNSTAT;
```

Running the DTLURDMO Utility on i5/OS

To run the DTLURDMO utility on i5/OS, enter the following command:

```
CALL PGM(DTLURDMO_executable_file_name)
```

For example:

```
CALL PGM(dtllib/DTLURDMO)
```

By default, the DTLURDMO utility looks for the DTLURDMO definition file in the CFG(DTLURDMO) member in the current *datalib* library. The DTLURDMO definition file contains the DTLURDMO control statements.

To specify an alternative location for the DTLURDMO definition file, enter the library name and file name of the definition file in the PARM option. For example:

```
CALL PGM(dtllib/DTLURDMO) PARM ('datalib/definition_file(DTLURDMO)')
```

Running the DTLURDMO Utility on Linux, UNIX, and Windows

On Linux, UNIX, or Windows, run the utility by navigating to the Informatica PowerExchange directory and entering `dtlurdmo` on the command line as follows:

```
dtlurdmo DTLURDMO_definition_file
```

For example:

```
dtlurdmo e:\powerexchange\bin\dtlurdmo.ini
```

The DTLURDMO definition file contains the DTLURDMO control statements.

If no definition file is specified, PowerExchange looks for the dtlurdmo.ini file in the current path.

Running the DTLURDMO Utility on z/OS

You run the utility by submitting the DTLURDMO job. The input control statements for this utility are read from SYSIN.

The following is an example of JCL to use when you run this utility on z/OS.

```
//DTLUSR01 JOB 'ADA',MSGLEVEL=1,
//          CLASS=A,MSGCLASS=X,NOTIFY=&SYSUID
//*
//DTL JCLLIB ORDER=(DTLUSR.V951.RUNLIB)
//*
//         SET HLQ=DTLUSR.V951
//*
//URDMO   PROC HLQ=&HLQ
//*
//STEP1   EXEC PGM=DTLURDMO,
//             REGION=0M,TIME=NOLIMIT
//STEPLIB  DD  DSN=CEE.SCEERUN,DISP=SHR
//         DD  DSN=&HLQ..LOADLIB,DISP=SHR
//*DTLCAMAP  DD DSN=&HLQ..DTLCAMAP,
//*             DISP=SHR
//*
//DTLMSG   DD  DSN=&HLQ..DTLMSG,DISP=SHR
//DTLCFG   DD  DSN=&HLQ..RUNLIB(DBMOVER),DISP=SHR
//DTLKEY   DD  DSN=&HLQ..RUNLIB(LICENSE),DISP=SHR
//DTLSGN   DD  DSN=&HLQ..RUNLIB(SIGNON),DISP=SHR
//DTLLOG   DD  SYSOUT=*
//DATAMAP  DD  DSN=&HLQ..V1.DATAMAPS,DISP=SHR
//DTLCAMAP DD  DSN=&HLQ..V1.DTLCAMAP,DISP=SHR
//DTLREPOS DD  DSN=&HLQ..V1.REPOS,DISP=SHR
//DTLAMCPR DD  DSN=&HLQ..V1.CCT,DISP=SHR
//SYSUDUMP DD  SYSOUT=*
//SYSOUT   DD  SYSOUT=*
//SYSPRINT DD  SYSOUT=*
//CEEDUMP  DD  SYSOUT=*
//        PEND
//*
//*
//* EXECUTE THE PROCEDURE
//*
//DTLLSTN EXEC URDMO
//*
//SYSIN    DD *
USER DTLUSR;
EPWD 095E463AC1C5D5B8;
TARGET DTLUSR;
SOURCE NODE1;
OUTPUT DTLUSR.V951.V1.DATAMAPS.TESTMIGR;
DETAIL;
DM_COPY;
    SELECT AM=ADABAS;
//*
```

DTLURDMO Utility Examples

The following examples show the control statements for example DTLURDMO jobs.

Copying Selected Data Maps

The following example uses the DM_COPY statement to copy data maps from systema to systemb. The following conditions apply:

- Only data maps with the test01 schema and the DB2 access method are copied.
- The data map test01.map01 is excluded from the copy.

The schema name of the copied data map is changed from test01 to test04.

```
global statements
SOURCE systema;
TARGET systemb;
more global statements
DM_COPY;
SELECT SCHEMA=test01 AM=DB2;
EXCLUDE MAP=map01;
RENAME SCHEMA=(test01,test04);
```

Copying All Data Maps

The following example uses the DM_COPY statement to copy all data maps from systema to systemb.

```
global statements
SOURCE systema;
TARGET systemb;
more global statements
DM_COPY;
```

Copying and Modifying Data Maps

The following example uses DM_COPY to copy all data maps from systema to systemb. All data maps are modified to use the DSN6 subsystem ID.

```
global statements
SOURCE systema;
TARGET systemb;
more global statements
DM_COPY;
MODIFY AM=DB2 DB2INSTANCE=DSN6;
```

Copying Registrations and Generating Extraction Maps

The following example uses REG_COPY to copy registrations from systema to systemb and generate extraction maps on systemb. This example illustrates how to migrate registrations from a test system to a production system.

Because a SELECT statement is not included, all registrations are selected.

The schema name of the registered table is changed from test01 to prod01 on the target system, and the database instance is changed to DSNP.

```
global statements
SOURCE systema;
TARGET systemb;
more global statements
```

```
REG_COPY;
CREATEXMAPS;
RENAME SCHEMA=(test01,prod01);
MODIFY NEW_DBID=DSNP;
```

Copying Registrations, Generating Extraction Maps, and Merging Extraction Maps with Bulk Data Maps

The following example uses REG_COPY to copy a specific registration from systema to systemb and generate an extraction map on systemb.

Also, the RELATED BULK statement merges the created extraction map with a bulk data map on the target system. The RENAME statements identify the bulk data map on the target system.

```
global statements
SOURCE systema;
TARGET systemb;
more global statements
REG_COPY;
CREATEXMAPS;
RELATED BULK;
SELECT REG_NAME=capture01;
RENAME BULKSCHEMA=(*,test) BULKMAP=(*,map01) BULKTABLE=(*,table01);
```

These statements copy registration capture01, generate extraction map dbtestdb.capture01 and merge the extraction map with data map test.map01_table01. Because only one registration is selected, you can use wildcards in the RENAME statement to explicitly force DTLURDMO to merge with the required bulk map.

You can use subsequent input REG_COPY statements to repeat the process for other registrations.

Copying Microsoft SQL Server Registrations and Generating Extraction Maps with a User-Defined Instance ID

This example copies Microsoft SQL Server registrations from systema to the local target system and generates corresponding extraction maps on target. The extraction map names include the unique instance identifier that you define in the NEW_DBID parameter for the database server and database name combination.

The example uses the following syntax:

```
global statements
SOURCE systema;
TARGET local;
DETAIL;
REPLACE;
REG_COPY;
KEEPREGTAG;
CREATEXMAPS;
SELECT DBTYPE=MSS DBID=CAPT123;
MODIFY NEW_DBID=CAPTUR2,MSSOPTS=(DBSERVER=ABC188888\server2,DBNAME=capture2);
```

In this syntax:

- The REG_COPY statement copies the registrations.
- The CREATEXMAPS statement generates the extraction maps.
- The SELECT statement must contain DBTYPE=MSS to identify the source type as Microsoft SQL Server.

- The MODIFY statement contains the NEW_DBID and MSSOPTS parameters for this example:
  - The NEW_DBID parameter specifies the optional user-defined instance identifier for the database server and database name that are defined in the MSSOPTS parameter. This instance identifier has a maximum length of seven characters. You can enter only one unique instance identifier for a database server and database name combination. If you are migrating from one environment to another, you can enter an instance identifier that matches the instance identifier in the source system. In this manner, you can avoid using a generated instance identifier and having to update the extraction map names in PowerCenter workflows and edit the PowerExchange Logger DBID parameter value on the target.
  - The MSSOPTS parameter identifies the SQL Server database server and database name.

Copying IMS Data Maps and Copying and Modifying Registrations

The following example runs DTLURDMO twice: first with a DM_COPY statement to copy data maps, and then with a REG_COPY statement to copy registrations and extraction maps.

The first execution of DTLURDMO uses the following input statements:

```
global statements
SOURCE systema;
TARGET systemb;
more global statements
DETAIL;
DM_COPY;
SELECT AM=DL1 SCHEMA=REGRESS MAP=FDPVF2;
MODIFY AM=DL1 PCBNUM=6;
RENAME SCHEMA=(REGRESS,IMSSRB);
```

These statements achieves the following results:

- The DM_COPY and SELECT statements copy the IMS DL/1 data map named REGRESS.FDPVF2 from the source system (PowerExchange Listener systsema) to the target system (PowerExchange Listener systemb).
- The RENAME SCHEMA statement changes the schema name from REGRESS to IMSSRB. The new data map name on the target is thus IMSSRB.FDPVF2.
- The MODIFY PCBNUM statement changes the PCB number in the data map on the target system to 6.

After DTLURDMO copies the data map from the source system to the target, a second exeuction of DTLURDMO copies the registration. This execution uses the following input statements:

```
global statements
SOURCE systema;
TARGET systemb;
more global statements
DETAIL;
REG_COPY;
CREATEXMAPS LOC=TARGET;
CHECKXREF;
SELECT REG_NAME=DEPT DBID=SYNC DBTYPE=IMS;
RENAME IMSSCHEMA=(REGRESS,IMSSRB);
```

These statements achieve the following results:

- The REG_COPY and SELECT statements copy the IMS registration named DEPT that has the DBID and RECON ID of SYNC from the source system (PowerExchange Listener systema) to the target system (PowerExchange Listener systemb).
- The RENAME IMSSCHEMA statement changes the schema name from REGRESS to IMSSRB. The new data map name on the target is thus IMSSRB.FDPVF2.
- The CHECKXREF statement forces the utility to load the corresponding data map on the target system and to update the registration with database organisation from the DBD specified in the data map.

- CREATEXMAPS generates the new extraction map. This statement eliminate the need to run the utility again with the XM_COPY statement.

  LOC=TARGET specifies that the data map used to create the extraction map is loaded from the target destination.

Copying IMS Data Maps and Registrations and Modifyng the IMSID Data Map Property

The following example runs DTLURDMO twice: first with a DM_COPY statement to copy data maps, and then with a REG_COPY statement to copy registrations and extraction maps. The REG_COPY statement also modifies the IMSID data map property.

The DM_COPY and REG_COPY statements also modify the schema name and IMSID data map property. In this example, the source system uses IM95 for the schema name for IMS data maps and IM95 for the RECON ID on registration groups and for the IMS system ID. The target system uses IM91 for the schema name for IMS data maps and IM91 for the RECON ID on registration groups and for the IMS system ID.

The first execution of DTLURDMO uses the following input statements:

```
global statements
SOURCE systema;
TARGET systemb;
more global statements
DETAIL;
DM_COPY;
SELECT AM=DL1 SCHEMA=IM95;
MODIFY AM=DL1 IMSID=IM91;
RENAME SCHEMA=(IM95,IM91);
```

These statements achieve the following results:

- The DM_COPY and SELECT statements copy IMS data maps with schema name IM95 and access method DL1 from the source system (PowerExchange Listener systema) to the target system (PowerExchange Listener systemb).
- The RENAME SCHEMA statement changes the schema name from IM95 to IM91.
- The MODIFY statement modifies the IMSID data map property for all selected data maps for which AM=DL1 from IM95 to IM91.

After DTLURDMO copies the data map from the source system to the target, a second exeuction of DTLURDMO copies the registration. This execution uses the following input statements:

```
global statements
SOURCE systema;
TARGET systemb;
more global statements
DETAIL;
REG_COPY;
CREATEXMAPS LOC=TARGET;
SELECT DBID=IM95 DBTYPE=IMS;
MODIFY NEW_DBID=IM91;
RENAME IMSSCHEMA=(IM95,IM91);
```

These statements achieve the following results:

- The REG_COPY and SELECT statements copy IMS registrations with DBID of IM95 from the source system (PowerExchange Listener systema) to the target system (PowerExchange Listener systemb).
- CREATEXMAPS generates the new extraction map. This statement eliminate the need to run the utility with XM_COPY statement.

  LOC=TARGET specifies that the data map used to create the extraction map is loaded from the target destination.

- The RENAME IMSSCHEMA statement changes the schema name from IM95 to IM91.
- The MODIFY NEW_DBID statement changes the IMSID data map property to IM91.

Note: When executing a REG_COPY statement, the DTLURDMO utility must load the IMS DBD if any of the following statements are also included:

- CHECKXREF
- RENAME DBD
- RENAME IMSMAP
- RENAME IMSSCHEMA

DTLURDMO retrieves the location of the IMS DBD library from the appropriate IMSID statement in the DBMOVER configuration file on the target system. The IMSID statement has the following syntax:

```
IMSID=(ims_ssid
      ,dbdlib
     [,RECON=(recon1
             [,recon2]
             [,recon3])]
)
```

DTLURDMO uses the IMSID statement that has a value for IMS_SSID that matches one of the following values, listed in order of priority:

1. If a MODIFY NEW_DBID statement is present, DTLURDMO uses this value, which represents the new data map property value.
2. If no MODIFY NEW_DBID statement is present, DTLURDMO uses the IMSID data map property. This value can be different from the RECON ID value.
3. If no MODIFY NEW_DBID statement is present, and the data map does not have a value for the IMSID property, DTLURDMO uses the value specified in the SELECT DBID= statement. This value is the value of the RECON ID of the registration group.
4. If none of the previous conditions apply, DTLURDMO issues a message indicating that the registration cannot be copied and will be skipped.

CHAPTER 14

DTLUTSK - Task Control Utility

This chapter includes the following topics:

DTLUTSK Utility Overview

This utility enables you to list active tasks, current locations, or allocated data sets. Additionally, you can use this utility to stop active tasks for PowerExchange applications that read data for remote requests running in the PowerExchange Listener.

You can use the following methods to run this utility:

- i5/OS command line
- Linux, UNIX, and Windows command line
- MVS job
- MVS command
- PowerExchange Navigator database row test

  Note: To run the DTLUTSK utility to run the LISTTASK, STOPTASK, or LISTLOCATIONS command, select **TASK_CNTL** from the **DB_Type** list in the **Database Row Test** dialog box.

DTLUTSK Command Line Utility on i5/OS

Syntax:

```
CALL PGM(library/DTLUTSK)PARM
  ('LOC=location
  CMD=command_name
  [TASKID=task_id]|[APPL=task_name]
```

```
[NODETYPE={N|A|S}
[UID=<user_id]
[PWD=<pwd_or_passphrase>]')
```

Parameters:

The following table describes the parameters:

Parameter	Description
LOC	The remote location where the task is running. Locations must be specified in NODE statements in the DBMOVER configuration file. If you enter LOCAL, the utility returns an error message.
CMD	The command name: - **LISTTASK**. Lists all current tasks. - **STOPTASK**. Stops the task specified by TASKID parameter. - **LISTLOCATIONS**. Lists all current locations - **LISTALLOC**. Lists all allocated data sets.
TASKID	When CMD=STOPTASK, the task ID of the task that you want to stop. You can determine the task ID by using the LISTTASK command.
APPL	When CMD=STOPTASK, the task name of the task that you want to stop. You can determine the task ID by using the LISTTASK command.
NODETYPE	When CMD=LISTLOCATIONS, specify one of the following node types: - N. Default. List locations that are defined in NODE statements in the DBMOVER configuration file. - A. List locations that are defined in NODE or SVCNODE statements in the DBMOVER configuration file. - S. List locations that are defined in SVCNODE statements in the DBMOVER configuration file.
UID	A user ID that has the authority to access the location, if required by your security settings.
PWD	A password or encrypted password for the specified user. If a password contains nonalphanumeric characters, you must enclose it in double quotation marks (“). Do not include double quotation marks within a password string, even if you enclose it in double quotation marks. For access to a remote i5/OS or z/OS location, you can enter a valid PowerExchange passphrase instead of a password. An i5/OS passphrase can be from 9 to 31 characters in length. A z/OS passphrase can be from 9 to 128 characters in length. Passphrases can contain the following characters: - Uppercase and lowercase letters - The numbers 0 to 9 - Spaces - The following special characters: `' - ; # \ , . / ! % & * ( ) _ + { } : @ | < > ?` **Note:** The first character is an apostrophe. Passphrases cannot include single quotation marks (‘), double quotation marks (“), or currency symbols. If a passphrase contains spaces, enclose it in double quotation marks ("). If a passphrase contains special characters, enclose it in triple double-quotation marks ("""). **Note:** On z/OS, a valid RACF passphrase can be up to 100 characters in length. PowerExchange truncates passphrases longer than 100 characters when passing them to RACF for validation. To use passphrases, ensure that the PowerExchange Listener runs with a security setting of SECURITY=(1,N) or higher in the DBMOVER member. For more information, see "SECURITY Statement" in the *PowerExchange Reference Manual*.

Note: Be sure to add the *dtllb* and *datalib* libraries to your library list before executing this command.

DTLUTSK Command Line Utility on Linux, UNIX, and Windows

Syntax:

```
DTLUTSK
  CMD=<command>
  [TASKID=<task_id>]
  [APPL=<task_name>]
  [NODETYPE={N|A|S}
   LOC=<location>
  [UID=<user_id>]
  [PWD=<pwd_or_passphrase>]
```

Parameters:

The following table lists and describes the parameters for DTLUTSK:

Parameter	Description
CMD	- **LISTTASK**. Lists all current tasks. - **STOPTASK**. Stops the task specified by TASKID parameter. - **LISTLOCATIONS**. Lists all current locations - **LISTALLOC**. Lists all allocated data sets.
TASKID	When CMD=STOPTASK, the task ID of the task that you want to stop. You can determine the task ID by using the LISTTASK command.
APPL	When CMD=STOPTASK, the task name of the task that you want to stop. You can determine the task ID by using the LISTTASK command.
NODETYPE	When CMD=LISTLOCATIONS, specify one of the following node types: - N. Default. List locations that are defined in NODE statements in the DBMOVER configuration file. - A. List locations that are defined in NODE or SVCNODE statements in the DBMOVER configuration file. - S. List locations that are defined in SVCNODE statements in the DBMOVER configuration file.
LOC	The remote location where the task is running. Locations must be specified in NODE statements in the DBMOVER configuration file. If you enter LOCAL, the utility returns an error message.

Parameter	Description
UID	A user ID that has the authority to access the location, if required by your security settings. For a location on a supported Linux, UNIX, or Windows system, if you have enabled PowerExchange LDAP user authentication, the user name is the enterprise user name. For more information, see the *PowerExchange Reference Manual*.
PWD	A password or encrypted password for the specified user. If a password contains nonalphanumeric characters, you must enclose it in double quotation marks (“). Do not include double quotation marks within a password string, even if you enclose it in double quotation marks. For access to a remote i5/OS or z/OS location, you can enter a valid PowerExchange passphrase instead of a password. An i5/OS passphrase can be from 9 to 31 characters in length. A z/OS passphrase can be from 9 to 128 characters in length. Passphrases can contain the following characters: - Uppercase and lowercase letters - The numbers 0 to 9 - Spaces - The following special characters: ’ - ; # \ , . / ! % & * ( ) _ + { } : @ | < > ? **Note:** The first character is an apostrophe. Passphrases cannot include single quotation marks (‘), double quotation marks (“), or currency symbols. **Note:** On z/OS, a valid RACF passphrase can be up to 100 characters in length. PowerExchange truncates passphrases longer than 100 characters when passing them to RACF for validation. If a passphrase contains spaces, enclose it in double quotation marks ("). If a passphrase contains special characters, enclose it in triple double-quotation marks ("""). To use passphrases, ensure that the PowerExchange Listener runs with a security setting of SECURITY=(1,N) or higher in the DBMOVER member. For more information, see "SECURITY Statement" in the *PowerExchange Reference Manual*.

Example Output:

```
2003-06-27 14:20:25                                TASK LIST

Name               Taskid          Partner       Port Status        Acc_Method
================================================================================
                     0740                              Active        RPX
                     1688                              Active        TASK_CNTL
```

DTLUTSK Utility Help on Linux, UNIX, and Windows

If you do not provide any arguments (such as DTLUTSK) or if you use a question mark (such as DTLUTSK ?) the utility will display the following assistance.

```
DTLUTSK Help: DTLUTSK CMD=LISTTASK/STOPTASK/LISTLOCATIONS/LISTALLOC LOC=location UID=uid
PWD=pwd/EPWD=encryptpwd
```

The following example shows the DTLUTSK help:

```
DTLUTSK Help: Examples:
DTLUTSK Help:    DTLUTSK CMD=LISTTASK LOC=NODE1 UID=uid PWD=pwd
DTLUTSK Help:    DTLUTSK CMD=STOPTASK TASKID=taskid LOC=NODE1 UID=uid PWD=pwd
DTLUTSK Help:    DTLUTSK CMD=STOPTASK APPL=taskname LOC=NODE1 UID=uid PWD=pwd
DTLUTSK Help:    DTLUTSK CMD=LISTLOCATIONS
DTLUTSK Help:    DTLUTSK CMD=LISTLOCATIONS LOC=NODE1
DTLUTSK Help:    DTLUTSK CMD=LISTALLOC LOC=NODE1 UID=uid PWD=pwd
```

DTLUTSK Job on MVS

This section provides information about submitting the DTLUTSK job.

DTLUTSK Job on MVS - Example JCL

The following example shows JCL for a DTLUTSK job. To supply the JOB card to run this JCL, copy the JOBCARD member into the DTLUTSK member.

```
//*
//* MEMBER DTLUTSK
//*
//INCS1 INCLUDE MEMBER=GENBULK
//*
//RUN EXEC PGM=DTLUTSK,
// PARM=('CMD=LISTTASK LOC=location UID=userid PWD=password')
/*
//* SAMPLE PARMS FOLLOW:
//* REMOVE COMMENT BEFORE CMD TO RUN
//* DTLUTSK Help: Examples:
//* PARM=('CMD=LISTTASK LOC=NODE1 UID=uid PWD=pwd')
//* PARM=('CMD=STOPTASK TASKID=taskid LOC=NODE1 UID=uid PWD=pwd')
//* PARM=('CMD=STOPTASK APPL=taskname LOC=NODE1 UID=uid PWD=pwd')
//* PARM=('CMD=LISTTASK TASKID=taskid LOC=location',
//* 'UID=uid EPWD=encryptpwd')
//STEPLIB DD DISP=SHR,DSN=&HLQ..LOADLIB
// DD DISP=SHR,DSN=&SCERUN
//*
//SYSIN DD DUMMY
/*
//*
//DTLMSG DD DSN=&HLQ..DTLMSG,DISP=SHR
//* IF USING MESSAGE OVERRIDE THEN CUSTOMIZE BELOW
//*DTLMSGO DD DISP=SHR,DSN=&RUNLIB(DTLMSGO)
//DTLCFG DD DSN=&RUNLIB(DBMOVER),DISP=SHR
//DTLKEY DD DSN=&RUNLIB(LICENSE),DISP=SHR
//DTLSGN DD DSN=&RUNLIB(SIGNON),DISP=SHR
//DTLLOG DD SYSOUT=*
//DTLLOG01 DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//SYSOUT DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//CEEDUMP DD SYSOUT=*
```

In this JCL, the PARM statements specify the utility commands. These statements can include the following common parameters:

LOC

A node name for the remote location where the task or tasks are running. This node name must be specified in a NODE statement in the DBMOVER configuration file.

UID

A user ID that can be used to access the remote location. You must also specify either the PWD or EPWD parameter.

PWD

A password for the specified user or a valid PowerExchange passphrase.

A passphrase for z/OS access can be from 9 to 128 characters in length and can contain the following characters:

- Uppercase and lowercase letters

- The numbers 0 to 9
- Spaces
- The following special characters:

  ```
  ' - ; # \ , . / ! % & * ( ) _ + { } : @ | < > ?
  ```

  Note: The first character is an apostrophe.

Passphrases cannot include single quotation marks (‘), double quotation marks (“), or currency symbols.

If a passphrase contains spaces, you must enclose it with double-quotation marks ("), for example, `"This is a passphrase"`. If a passphrase contains special characters, you must enclose it with triple double-quotation characters (`"""`), for example, `"""This passphrase contains special characters ! % & *."""`. If a passphrase contains only alphanumeric characters without spaces, you can enter it without delimiters.

To use passphrases, ensure that the PowerExchange Listener runs with a security setting of SECURITY=(1,N) or higher in the DBMOVER member. For more information, see "SECURITY Statement" in the *PowerExchange Reference Manual*.

Note: On z/OS, a valid RACF passphrase can be up to 100 characters in length. PowerExchange truncates passphrases longer than 100 characters when passing them to RACF for validation.

Do not also specify the EPWD parameter.

EPWD

An encrypted password for the specified user.

For a location on z/OS, you can enter an encrypted PowerExchange passphrase instead of an encrypted password. Do not encrypt a passphrase that contains invalid characters, such as double-quotation marks, single quotation marks, or currency symbols.

DTLUTSK Job on MVS - Example Output

```
******************************** TOP OF DATA ***********************************

2007-10-25 13:28:45                              TASK LIST
0Name               Taskid          Partner       Port Status
Acc_Method             Sessid

================================================================================================
========
                    x 0001        10.3.4.57       6900 Active        CAPXRT
                      0002        127.0.0.1       6900 Active        TASK_CNTL

******************************** BOTTOM OF DATA ********************************
```

DTLUTSK Command Line Utility on MVS

The following command line commands are available:

- **LISTTASK.** Lists all current tasks.
- **STOPTASK.** Stops the task specified by the TASKID parameter.
- **LISTLOCATIONS.** Lists all current locations.
- **LISTALLOC.** Lists all allocated data sets.
- **FREEALLOC.** Frees the allocated data sets specified by the DDNAME and data set name.

LISTTASK Command on MVS

Lists all current tasks.

Syntax:

MODIFY < *listener name* >,LISTTASK

Example output from the utility:

```
DTL-00711 Active tasks:
DTL-00712 taskid=0, partner=10.7.16.71, port=16634, name=, am=DB2, status=
DTL-00713 1 active tasks
```

STOPTASK Command on MVS

Stops the task specified by TASKID parameter or by application name.

Syntax for stopping by TASKID:

```
MODIFY <listener name>,STOPTASK TASKID=<taskid>
```

Syntax for stopping by application name:

```
MODIFY <listener name>,STOPTASK <application name>
```

Syntax for the MVS modify command to stop a task by application name does not use the same syntax as DTLUTSK.

Note: When you stop CDC sessions, STOPTASK waits for a commit boundary before terminating the task. For more information about commit boundaries and processing, see *PowerExchange CDC Guide for z/OS*.

LISTLOCATIONS Command on MVS

Lists all current locations.

Syntax:

```
MODIFY <listener name>,LISTLOCATIONS [NODETYPE={N|A|S}]
```

LISTALLOC Command on MVS

Lists all allocated data sets.

Syntax:

```
MODIFY <listener name>,LISTALLOC
```

Example output:

```
Alloc: DDN=<STEPLIB > DSN=<CEE.SCEERUN                                >
Alloc: DDN=<        > DSN=<DTLUSR.DEVBLD.LOADLIB                      >
Alloc: DDN=<        > DSN=<DTLUSR.DEVBLD.LOAD                         >
Alloc: DDN=<        > DSN=<DTLUSR.DEVBLD.NIML.USERLIB                 >
Alloc: DDN=<DTLAMCPR> DSN=<DTLUSR.DEVBLD.V1.CCT                       >
Alloc: DDN=<DTLCACDE> DSN=<DTLUSR.DEVBLD.V1.CDEP                      >
Alloc: DDN=<DTLCACDC> DSN=<DTLUSR.DEVBLD.V1.CDCT                      >
Alloc: DDN=<DTLCAMAP> DSN=<DTLUSR.DEVBLD.V1.DTLCAMAP                  >
Alloc: DDN=<DTLMSG  > DSN=<DTLUSR.DEVBLD.DTLMSG                       >
Alloc: DDN=<DTLCFG  > DSN=<DTLUSR.V811.RUNLIB                         >
Alloc: DDN=<DTLKEY  > DSN=<DTLUSR.V811.RUNLIB                         >
Alloc: DDN=<DTLSGN  > DSN=<DTLUSR.V811.RUNLIB                         >
Alloc: DDN=<DTLLOG  > DSN=<DTLUSR.DTLLOG.LOG                          >
Alloc: DDN=<DATAMAP > DSN=<DTLUSR.V811.V1.DATAMAPS                    >
Alloc: DDN=<SYSUDUMP> DSN=<DTLUSR.DTLUSR2.JOB05761.D0000101.?         >
Alloc: DDN=<SYSOUT  > DSN=<DTLUSR.DTLUSR2.JOB05761.D0000102.?         >
Alloc: DDN=<URLEOUT > DSN=<DTLUSR.DTLUSR2.JOB05761.D0000103.?         >
```

```
Alloc: DDN=<SYSPRINT> DSN=<DTLUSR.DTLUSR2.JOB05761.D0000104.?           >
Alloc: DDN=<CEEDUMP > DSN=<DTLUSR.DTLUSR2.JOB05761.D0000105.?           >
Alloc: DDN=<CXX     > DSN=<DCOM.V10.CXX                                 >
Alloc: DDN=<DTLOUT  > DSN=<DTLUSR.DTLUSR2.JOB05761.D0000106.?           >
Alloc: DDN=<DTLERR  > DSN=<DTLUSR.DTLUSR2.JOB05761.D0000107.?           >
Command < LISTALLOC> succeeded
```

FREEALLOC Command on MVS

Use FREEALLOC to close and deallocate a data set that has already been dynamically allocated by a listener.

It should only be used in situations such as when a task abends and the resource managers fail to close all the dynamically allocated data sets.

Syntax:

```
MODIFY <listener name>,FREEALLOC DDN=<ddname> FN=<data set name>
```

The DDN and FN parameters are mandatory.

Note: The data set is not deallocated if any of the following apply:

- The request is not from the listener.
- The request is for a file that has not been dynamically allocated by the listener, and does not have a DD name starting with SYS0.
- The request is issued without the file name being specified.

Running the DTLUTSK Utility in the PowerExchange Navigator

The TASK_CNTL data access method is available so that you can perform a database row test to retrieve the results of the LISTTASK, STOPTASK, or LISTLOCATIONS command from the PowerExchange Navigator.

Note: STOPTASK works only with the CAPXRT access method.

To run the DTLUTSK utility in the PowerExchange Navigator:

1. In the **Resource Explorer**, double-click a data map that is defined for the location where the PowerExchange Listener is running to open the data map.

   Note: Alternatively, you can open an extraction map or personal metadata profile that is defined for the PowerExchange Listener location.
2. On the **Data Map** tab, select a table view, and then click **File** > **Database Row Test** on the menu bar.

   A message might prompt you to sent the data map to a remote location.
3. In the **Data Map Remote Node** dialog box, enter connection information for the location where the PowerExchange Listener is running, and click **OK**.

   The **Database Row Test** dialog box appears.
4. In the **DB Type** list, select **TASK_CNTL**.

5. In the **Fetch** list, select one of the following commands:
   - **List Locations**. Displays information about the locations that are defined in NODE or SVCNODE statements in the DBMOVER configuration file on the system where the PowerExchange Listener is running. The output includes the node name, IP address, port number, send and receive buffer sizes and lengths, receive timeout, and SSL use.
   - **List Task**. Displays information about each active task that is running under the PowerExchange Listener. The output includes the task ID, TCP/IP address, port number, application name, access type, and status.
   - **Stop Task**. Stops a specific PowerExchange Listener task. To identify the task, you must enter a task ID or application name.
6. If you are issuing a STOPTASK command, enter a task ID or application name in the **SQL Statement** box. Use the following syntax:

   ```
   stoptask {taskid=task_id|appname=application_name}
   ```

   Do not include the curly brackets. These brackets indicate a choice of either taskid or appname is required.

   Note: Do not enter the application name in the **Application** field. The **Application** field is ignored for TASK_CNTL commands.
7. If you are issuing a LISTLOCATIONS command, optionally enter the node type in the **SQL Statement** box. Use the following syntax:

   ```
   listlocations nodetype={N|A|S}
   ```

   Specify one of the following values for nodetype:
   - N. Default. List locations that are defined in NODE statements in the DBMOVER file.
   - A. List locations that are defined in NODE or SVCNODE statements in the DBMOVER file.
   - S. List locations that are defined in SVCNODE statements in the DBMOVER file.
8. Click **Go**.

   The **Database Row Test Output** window displays the output for the command.

For more information, see the "Database Row Test" chapter in the *PowerExchange Navigator User Guide*.

DTLUTSK Utility Security Requirements

The following security requirements apply to the DTLUTSK utility.

DTLUTSK Utility Security Requirements on MVS

If the SECURITY configuration parameter is set to (2,*x*), where *x* is N or Y, then the following RACF (or similar security package) resources must be defined to MVS, using the RACF_CLASS configuration parameter, and access granted to the required users:

```
DTL.TASKCTRL.DISPLAY
DTL.TASKCTRL.STOPTASK
```

These enables users to display the active tasks then stop them respectively.

DTLUTSK Utility Security Requirements on i5/OS

On i5/OS, if the SECURITY parameter is set to (2,*x*), where *x* is N or Y, you must define security statements as follows, replacing DATALIB with the required data library:

```
GRTOBJAUT OBJ(DATALIB/AUTHTSKLST) OBJTYPE(*FILE) USER(USERID) AUT(*USE)
GRTOBJAUT OBJ(DATALIB/AUTHTSKSTP) OBJTYPE(*FILE) USER(USERID) AUT(*USE)
```

Using Signon.txt to Authorize Users to Display or Stop Tasks

If running with a configuration setting of SECURITY=(*n*,Y) where *n* is 0 to 2, an additional parameter is available for allowing the use of list and stop tasks:

```
/*   4.    TASKCNTRL= is an optional function allowed
/*          Format is D or S
/*          If it is supplied, then the user can use Task Control to
/*          Display or Stop tasks.
/*          This signon list will only be used if Security=(n,Y) is used
/*          in the config.
```

CHAPTER 15

EDMLUCTR - Log Scan and Print Utility

This chapter includes the following topics:

EDMLUCTR Utility Overview

Use the EDMLUCTR utility to perform the following tasks:

- Produce summary information about each log record.
- Produce detailed information about change records and units of work (UOWs) records.
- Produce summary information, by registration tag name, about all sources for which changes are captured.
- List UOWs that have not yet ended.

For more information about the PowerExchange Logger and Post Log Merge, see *PowerExchange CDC Guide for z/OS*.

Supported Operating Systems for the EDMLUCTR Utility

The EDMLUCTR utility can run on z/OS only.

Control Statement Syntax for the EDMLUCTR Utility

Use the following syntax for the EDMLUCTR utility control statements:

```
[-SEL
   [CHANGE-DETAIL]
   [LOGRBA=logrba]
   [ENDRBA=endrba]
   [PACKET-DETAIL]
   [RECORDS={nnnnnnnn|EOF}]
   [SUMM] ]

[-MASK mask]
```

The following rules and guidelines apply:

- Use the SYSIN DD JCL statement to enter the utility control statements.
- All of the control statements are optional and begin in column 1.
- Control statements must end with a blank and must not exceed 80 characters in length.
- Use one or more spaces as a delimiter between parameters for a control statement.
- No continuation syntax exists.
- If more than a single line is required for a -SEL control statement, code -SEL at the beginning of each subsequent line that includes additional parameters.
- The value for a parameter cannot continue from line to line.
- If you code multiple -MASK statements, only the last one is used.

Control Statement Parameters for the EDMLUCTR Utility

Review the parameter descriptions to determine which parameters to use in the EDMLUCTR control statements.

-SEL Statement

-SEL has the following parameters:

CHANGE-DETAIL

Optional. Prints summary and detailed information, in hexadecimal format, about change records. If not specified, only summary information for change records prints.

LOGRBA

Optional. Specifies an RBA in the log data sets to use as the starting point for the EDMLUCTR utility. In a Post-Log Merge environment, LOGRBA specifies a timestamp value in the log data sets as an unstructured TOD-clock value.

As the starting point, EDMLUCTR uses the first log record that has an RBA or a timestamp that is equal to or higher than the specified value.

Specify up to 12 hexadecimal digits for the LOGRBA value. You can omit leading zeroes.

Note: With Post-Log Merge configurations, LOGRBA must be specified and the LOGRBA value must be 16 hexadecimal digits. LOGRBA values represent the timestamp of the requested data when using Post-Log Merge.

If no parameter is specified, LOGRBA is the default. Its default value is the RBA that is recorded in the emergency restart data set (ERDS) from the latest checkpoint.

ENDRBA

Optional. Specifies an RBA in the log data sets to use as the ending point for the EDMLUCTR utility. In a Post-Log Merge environment, ENDRBA specifies a timestamp value in the log data sets as an unstructured TOD-clock value.

EDMLUCTR prints or scans log records until it finds a log record that has an RBA or a timestamp that is equal to or greater than the specified ENDRBA value. When EDMLUCTR reaches that point, it ends.

Specify up to 16 hexadecimal digits for the ENDRBA value. You can omit leading zeroes.

PACKET-DETAIL

Optional. Prints summary and detailed information, in hexadecimal format, about UOW records. If not specified, only summary information for UOW records prints.

RECORDS

Optional. Prints or scans the specified number of log records.

When you specify RECORDS=EOF, EDMLUCTR prints all records from the specified or default start location to the current end of the log data.

If you specify -SEL RECORDS and the -MASK statement, EDMLUCTR uses the RECORDS value as the number of records to scan for the mask value rather than as the number of records to print.

Minimum is 1. Maximum is 99999999. Default is 5,000.

SUMM

Optional. Prints only change summary information.

Change summary information includes the total number of inserts, updates, and deletes found in the log data scanned, ordered by source registration tag name.

-MASK Statement

-MASK has the following parameter:

mask

Required. Specify a filter in one of the following formats:

- A character value, such as a DB2 table name, without embedded blanks. Use the hexadecimal format for character strings with embedded blanks.
- A hexadecimal value, such as a UOW number. Enclose hexadecimal character strings in single quotes and proceed the string with the letter X.

If you specify both -SEL RECORDS and -MASK, EDMLUCTR uses the RECORDS value as the number of records to scan for the mask value rather than as the number of records to print.

Maximum length is 70 characters.

Running the EDMLUCTR Utility

PowerExchange provides sample JCL for the EDMLUCTR utility in the LOGPRINT member of the SAMPLIB library.

The following JCL statements are required to run the utility:

```
//         JOB
//READER   EXEC PGM=EDMLUCTR
//STEPLIB  DD   DISP=SHR,DSN=hlq.LOAD
//ERDS01   DD   DISP=SHR,DSN=your.ERDS01
//EDMPARMS DD   DISP=SHR,DSN=your.USERLIB
//SYSIN    DD   *
```

JOB

Initiates the job.

EXEC PGM=EDMLUCTR

Invokes the utility.

STEPLIB DD

Defines the PowerExchange LOAD library that contains the utility.

ERDS01 DD

Defines the PowerExchange Logger emergency restart data set (ERDS) that contains the inventory of log data sets containing the log records to be displayed. Specify only one ERDS data set.

EDMPARMS DD

Defines the data set that contains the EDMSDIR options module.

SYSIN DD

Defines the utility control statements.

EDMLUCTR Utility Usage Notes

Consider the following points before using the EDMLUCTR utility:

- If you specify old LOGRBA values, the utility might read archive log data sets that have been migrated by the storage management system. Verify that you have sufficient DASD to recall any migrated archive log data sets.
- You can use the EDMLUCTR utility in either a single PowerExchange Logger environment or Post-Log Merge environment.
- You can run the EDMLUCTR utility whether or not the PowerExchange Logger is running.

EDMLUCTR Utility Examples

The following are examples of the EDMLUCTR utility.

EDMLUCTR Utility - Example 1

The following JCL statements print summary data for all log records, starting with the RBA recorded in the ERDS at the time the latest PowerExchange Logger checkpoint was taken:

```
//         JOB
//READER   EXEC PGM=EDMLUCTR
//STEPLIB  DD   DISP=SHR,DSN=hlq.LOAD
//ERDS01   DD   DISP=SHR,DSN=your.ERDS01
//EDMPARMS DD   DISP=SHR,DSN=your.USERLIB
//SYSIN    DD   *
//
```

The resulting output is:

```
 18:53:31.86  L O G   S T A R T
 18:53:31.83  PWXEDM172502I Log Scan/Print Utility Initialization in-progress product level V2.4.05 04/22/2014
 18:53:31.83  Echo of input from SYSIN.............
 18:53:31.85  End of input from SYSIN.............
 18:53:42.04  PWXEDM172191I EDMLRDS: LMF will begin transferring data for Log Scan/Print Utility at X'0000000050000000'
 18:53:42.14  PWXEDM172146I EDMLRDP: LMF now processing EDMTEST.DEV.V1.PRILOG.DS01 for Log Scan/Print Utility
 18:53:42.37               Log-rec  EDP-UOW=LOGGER00000000500000000000 LogRBA=0000000050000000
 18:53:42.37               Log-rec  EDP-UOW=LOGGER0000000050B400000001 LogRBA=0000000050B40000
 18:53:42.37               Log-rec  EDP-UOW=LOGGER00000000523400000003 LogRBA=0000000052340000
 18:53:42.37               Log-rec  EDP-UOW=AUSL  0000000052E800000001 LogRBA=0000000052E80000
 18:53:42.37               Beg-pkt  EDP-UOW=AUSL  0000000054680000000 LogRBA=0000000054680000
 18:53:42.37                        ECCR-UOW=AUSPRT01 AUSPRT01 C1E4E2D7D9E3F0F1 008F803000000001
 18:53:42.37                        Timestamp-18:35:40:11 Date-04/29/2014
 18:53:42.37               Chg-rec  EDP-UOW=AUSL  0000000054680000000 LogRBA=0000000054F00000
 18:53:42.37                        ECCR-UOW=AUSPRT01 C1E4E2D7D9E3F0F1 008F803000000001
 18:53:42.37                        Source=VSM Func=ISRT Srcname=VSAMEDMTEST.VSAM.KSDS01
 18:53:42.37                        Timestamp-18:35:40:18 Date-04/29/2014
 18:53:42.37               Chg-rec  EDP-UOW=AUSL  0000000054680000000 LogRBA=0000000056550000
 18:53:42.37                        ECCR-UOW=AUSPRT01 C1E4E2D7D9E3F0F1 008F803000000001
 18:53:42.37                        Source=VSM Func=ISRT Srcname=VSAMEDMTEST.VSAM.KSDS01
 18:53:42.37                        Timestamp-18:35:40:36 Date-04/29/2014
  . . . . .
  . . . . .
  . . . . .
 18:53:43.63               Srv-rec  EDP-UOW=ECCRCTF5F24040404040404040 LogRBA=0000001902F90000
 18:53:43.63               Srv-rec  EDP-UOW=ECCRCTF5F24040404040404040 LogRBA=0000001906090000
 18:53:43.63               Log-rec  EDP-UOW=AUSL  0000001909350000001 LogRBA=0000001909350000
 18:53:43.63               Srv-rec  EDP-UOW=ECCRCTF5F24040404040404040 LogRBA=000000190AB50000
 18:53:43.63               Srv-rec  EDP-UOW=ECCRCTF5F24040404040404040 LogRBA=000000190DC50000
 18:53:43.63               Log-rec  EDP-UOW=AUSL  0000019110500000001 LogRBA=0000001911050000
 18:53:43.63               Srv-rec  EDP-UOW=AUSDB2F0F10000000000004040 LogRBA=0000001912850000
 18:53:43.63               Srv-rec  EDP-UOW=AUSDB2F0F10000000000004040 LogRBA=0000001916890000
 18:53:43.63               Beg-pkt  EDP-UOW=AUSL  000000191A8D00000000 LogRBA=000000191A8D0000
 18:53:43.63                        ECCR-UOW=AUSDB201 01       F0F1000000000001 605FE82B00000000
 18:53:43.63                        Timestamp-18:53:19:68 Date-05/07/2014
 18:53:43.63               Chg-rec  EDP-UOW=AUSL  000000191A8D00000000 LogRBA=000000191D410000
 18:53:43.63                        ECCR-UOW=01       F0F1000000000001 605FE82B00000000
 18:53:43.63                        Source=DB2 Func=DLET Srcname=DB2DSNBtenchar1
 18:53:43.63                        Timestamp-18:53:19:68 Date-05/07/2014
 18:53:43.63               Chg-rec  EDP-UOW=AUSL  000000191A8D00000000 LogRBA=00000019219F0000
 18:53:43.63                        ECCR-UOW=01       F0F1000000000001 605FE82B00000000
 18:53:43.63                        Source=DB2 Func=DLET Srcname=DB2DSNBtenchar1
 18:53:43.63                        Timestamp-18:53:19:68 Date-05/07/2014
 18:53:43.63               Com-pkt  EDP-UOW=AUSL  000000191A8D00000000 LogRBA=0000001925E90000
 18:53:43.63                        ECCR-UOW=AUSDB201 01       F0F1000000000001 605FE82B00000000
 18:53:43.63                        Timestamp-18:53:19:68 Date-05/07/2014
 18:53:48.67  PWXEDM172198I EDMLPOPU: LMF table populate tasks are terminating due to an operator stop or PAC termination
 18:53:48.70  PWXEDM172195I EDMLUPLU: LMF task for Log Scan/Print Utility is terminating due to an operator stop or PAC
termination
 18:53:48.73  Totals by Source.............
 18:53:48.73               VSAMEDMTEST.VSAM.KSDS01             Isrt= 5 Repl= 0 Dlet= 0 Unk= 0
 18:53:48.73               VSAMEDM.DEV.EDMAB123                Isrt= 0 Repl= 1000 Dlet= 0 Unk= 0
 18:53:48.73               VSAMEDM.DEV.EDMAB123                Isrt= 1000 Repl= 0 Dlet= 1000 Unk= 0
 18:53:48.73               DB2DSNBtenchar1                     Isrt= 2 Repl= 2 Dlet= 2 Unk= 0
 18:53:48.73  Open Uows.....................
 18:53:48.74  L O G   E N D
```

EDMLUCTR Utility - Example 2

The following JCL statements print summary data for all log records starting from the specified RBA and detailed information in hexadecimal format about the change records:

```
//         JOB
//READER   EXEC PGM=EDMLUCTR
//STEPLIB  DD   DISP=SHR,DSN=hlq.LOAD
```

```
//ERDS01   DD   DISP=SHR,DSN=your.ERDS01
//EDMPARMS DD   DISP=SHR,DSN=your.USERLIB
//SYSIN    DD   *
-SEL   LOGRBA=000000191A8D0000 CHANGE-DETAIL
-SEL   RECORDS=10
//
```

Note: The job prints the detailed information about the change records because the optional CHANGE-DETAIL parameter is included.

The resulting output is:

```
 19:08:00.74  L O G   S T A R T
 19:08:00.69  PWXEDM172502I Log Scan/Print Utility Initialization in-progress product level V2.4.05 04/22/2014
 19:08:00.70  Echo of input from SYSIN..............
 19:08:00.70  -SEL LOGRBA=000000191A8D0000 CHANGE-DETAIL
 19:08:00.70  -SEL RECORDS=10
 19:08:00.72  End of input from SYSIN..............
 19:08:00.72  PWXEDM172191I EDMLUCTR: LMF will begin transferring data for Log Scan/Print Utility at X'000000191A8D0000'
 19:08:11.01  PWXEDM172146I EDMLRDP: LMF now processing WBRUMB1.DEV.V1.PRILOG.DS01 for Log Scan/Print Utility
 19:08:11.67               Beg-pkt  EDP-UOW=AUSL  000000191A8D00000000 LogRBA=000000191A8D0000
 19:08:11.67                        ECCR-UOW=AUSDB201 01       F0F1000000000001 605FE82B00000000
 19:08:11.67                        Timestamp-18:53:19:68 Date-05/07/2014
 19:08:11.67               Chg-rec  EDP-UOW=AUSL  000000191A8D00000000 LogRBA=000000191D410000
 19:08:11.67                        ECCR-UOW=01       F0F1000000000001 605FE82B00000000
 19:08:11.67                        Source=DB2 Func=DLET Srcname=DB2DSNBtenchar1
 19:08:11.67                        Timestamp-18:53:19:68 Date-05/07/2014
 19:08:11.67                 0000  00000003 00000014 000002B4 000002B4            ©   ©
 19:08:11.67                 0010  0000044A 02A00101 C3C46008 00000002   œ¢ µ  CD--
 19:08:11.67                 0020  E2C40000 00000000 00000000 00040000 SD           œ
 19:08:11.67                 0030  00000000 00000000 0000CD1E 42982685          ò âq e
 19:08:11.67                 0040  8081C4C2 F2C4E2D5 C2A38595 83888199 ØaDB2DSNBtenchar
 19:08:11.67                 0050  F1404040 40404040 40404040 40404040 1
 19:08:11.67                 0060  40404040 4040C1E4 E2D34040 00000019       AUSL
 19:08:11.67                 0070  1A8D0000 00000000 00191D41 00000000 ’ý
  . . . . .
  . . . . .
 19:08:11.70  Default 5000 or RECORDS= threshold reached
 19:08:11.73  PWXEDM172198I EDMLPOPU: LMF table populate tasks are terminating due to an operator stop or PAC termination
 19:08:11.77  PWXEDM172195I EDMLUPLU: LMF task for Log Scan/Print Utility is terminating due to an operator stop or PAC
termination
 19:08:11.79  Totals by Source..............
 19:08:11.79               DB2DSNBtenchar1                    Isrt= 2 Repl= 1 Dlet= 2 Unk= 0
 19:08:11.79  Open Uows.....................
 19:08:11.79                Edp-UOW=AUSL  0000001936AD00000000 LogRBA=0000001936AD0000
 19:08:11.79                        ECCR-UOW=01       F0F1000000000001 605FF6E000000000
 19:08:11.80  L O G   E N D
```

EDMLUCTR Utility - Example 3

The following JCL statements filter records by using the -MASK value of DB2DSNB and then print the records starting from a specific RBA.

```
//         JOB
//READER   EXEC PGM=EDMLUCTR
//STEPLIB  DD   DISP=SHR,DSN=hlq.LOAD
//ERDS01   DD   DISP=SHR,DSN=your.ERDS01
//EDMPARMS DD   DISP=SHR,DSN=your.USERLIB
//SYSPRINT DD   SYSOUT=*
//SYSUDUMP DD   SYSOUT=*
//SYSIN    DD   *
-SEL   LOGRBA=000000191A8D0000 RECORDS=10
-MASK DB2DSNB
//
```

The inclusion of the optional RECORDS parameter limits the number of record scans for the character string DB2DSNB.

The resulting output is:

```
 19:12:40.93  L O G   S T A R T
 19:12:40.89  PWXEDM172502I Log Scan/Print Utility Initialization in-progress product level V2.4.05 04/22/2014
 19:12:40.89  Echo of input from SYSIN..............
 19:12:40.89  -SEL  LOGRBA=000000191A8D0000 RECORDS=10
 19:12:40.89  -MASK DB2DSNB
 19:12:40.91  End of input from SYSIN..............
 19:12:40.91  PWXEDM172191I EDMLUCTR: LMF will begin transferring data for Log Scan/Print Utility at X'000000191A8D0000'
 19:12:51.12  PWXEDM172146I EDMLRDP: LMF now processing WBRUMB1.DEV.V1.PRILOG.DS01 for Log Scan/Print Utility
```

```
 19:12:52.69                Chg-rec  EDP-UOW=AUSL  000000191A8D00000000 LogRBA=000000191D410000
 19:12:52.69                         ECCR-UOW=01       F0F1000000000001 605FE82B00000000
 19:12:52.69                         Source=DB2 Func=DLET Srcname=DB2DSNBtenchar1
 19:12:52.69                         Timestamp-18:53:19:68 Date-05/07/2014
 19:12:52.69                Chg-rec  EDP-UOW=AUSL  000000191A8D00000000 LogRBA=00000019219F0000
 19:12:52.69                         ECCR-UOW=01       F0F1000000000001 605FE82B00000000
 19:12:52.69                         Source=DB2 Func=DLET Srcname=DB2DSNBtenchar1
 19:12:52.69                         Timestamp-18:53:19:68 Date-05/07/2014
 19:12:52.69                Chg-rec  EDP-UOW=AUSL  000000019289D00000000 LogRBA=000000192B510000
 19:12:52.69                         ECCR-UOW=01       F0F1000000000001 605FF25800000000
 19:12:52.69                         Source=DB2 Func=ISRT Srcname=DB2DSNBtenchar1
 19:12:52.69                         Timestamp-18:53:19:68 Date-05/07/2014
 19:12:52.69                Chg-rec  EDP-UOW=AUSL  000000019289D00000000 LogRBA=000000192F9B0000
 19:12:52.69                         ECCR-UOW=01       F0F1000000000001 605FF25800000000
 19:12:52.69                         Source=DB2 Func=ISRT Srcname=DB2DSNBtenchar1
 19:12:52.69                         Timestamp-18:53:19:68 Date-05/07/2014
 19:12:52.69                Chg-rec  EDP-UOW=AUSL  0000001936AD00000000 LogRBA=0000001939610000
 19:12:52.69                         ECCR-UOW=01       F0F1000000000001 605FF6E000000000
 19:12:52.69                         Source=DB2 Func=UPDT Srcname=DB2DSNBtenchar1
 19:12:52.69                         Timestamp-18:53:19:68 Date-05/07/2014
 19:12:52.70  Default 5000 or RECORDS= threshold reached
 19:12:52.72  PWXEDM172198I EDMLPOPU: LMF table populate tasks are terminating due to an operator stop or PAC termination
 19:12:52.76  PWXEDM172195I EDMLUPLU: LMF task for Log Scan/Print Utility is terminating due to an operator stop or PAC
termination
 19:12:52.85  Totals by Source.............
 19:12:52.85                DB2DSNBtenchar1                    Isrt= 2 Repl= 1 Dlet= 2 Unk= 0
 19:12:52.85  Open Uows.....................
 19:12:52.89  L O G   E N D
```

EDMLUCTR Utility - Example 4

If you run the utility in a Post-Log Merge environment, the following JCL statements print summary data for all log records starting from a specific timestamp:

```
//          JOB
//READER    EXEC PGM=EDMLUCTR
//STEPLIB   DD   DISP=SHR,DSN=hlq.LOAD
//ERDS01    DD   DISP=SHR,DSN=your.ERDS01
//EDMPARMS  DD   DISP=SHR,DSN=your.USERLIB
//SYSIN     DD   *
-SEL   LOGRBA=CD1FCF19F4713301 RECORDS=EOF
//
```

The resulting output is:

```
 19:27:31.86  L O G   S T A R T
 19:27:31.83  DTLEDM172502I Log Scan/Print Utility Initialization in-progress product level V2.4.05 04/22/2014
 19:27:31.83  Echo of input from SYSIN.............
 19:27:31.84  -SEL    LOGRBA=CD1FCF19F4713301 RECORDS=EOF
 19:27:31.85  End of input from SYSIN.............
 19:27:42.04  DTLEDM172191I EDMLRDS: LMF will begin transferring data for Log Scan/Print Utility at X'0000000050000000'
 19:27:42.14  DTLEDM172146I EDMLRDP: LMF now processing AUSQA.DEV.V1.PRILOG.DS01 for Log Scan/Print Utility
 19:27:42.37               Log-rec  EDP-UOW=LOGGER00000000500000000000 LogRBA=CD1FCF19F4713301
 19:27:42.37               Log-rec  EDP-UOW=LOGGER0000000050B400000001 LogRBA=CD1FCF19F472B075
 19:27:42.37               Log-rec  EDP-UOW=LOGGER00000000523400000003 LogRBA=CD1FCF19F473A691
 19:27:42.37               Log-rec  EDP-UOW=AUSL  0000000052E800000001 LogRBA=CD1FCF19F4751B0A
 19:27:42.37               Beg-pkt  EDP-UOW=AUSL  00000000546800000000 LogRBA=CD1FCF19F47644B1
 19:27:42.37                        ECCR-UOW=AUSPRT01 AUSPRT01 E6D9C2D7D9E3F0F1 008F803000000001
 19:27:42.37                        Timestamp-19:25:40:11 Date-04/29/2014
 19:27:42.37               Chg-rec  EDP-UOW=AUSL  00000000546800000000 LogRBA=CD1FCF19F47815C3
 19:27:42.37                        ECCR-UOW=AUSPRT01 E6D9C2D7D9E3F0F1 008F803000000001
 19:27:42.37                        Source=VSM Func=ISRT Srcname=VSAMEDM.VSAM.KSDS01
 19:27:42.37                        Timestamp-19:25:40:18 Date-04/29/2014
 19:27:42.37               Chg-rec  EDP-UOW=AUSL  00000000546800000000 LogRBA=CD1FCF19F47AC50D
 19:27:42.37                        ECCR-UOW=AUSPRT01 E6D9C2D7D9E3F0F1 008F803000000001
 19:27:42.37                        Source=VSM Func=ISRT Srcname=VSAMEDM.VSAM.KSDS01
 19:27:42.37                        Timestamp-19:25:40:36 Date-04/29/2014
 19:27:42.37               Chg-rec  EDP-UOW=AUSL  00000000546800000000 LogRBA=CD1FCF19F47B2F04
 19:27:42.37                        ECCR-UOW=AUSPRT01 E6D9C2D7D9E3F0F1 008F803000000001
 19:27:42.37                        Source=VSM Func=ISRT Srcname=VSAMEDM.VSAM.KSDS01
 19:27:42.37                        Timestamp-19:25:40:55 Date-04/29/2014
 19:27:42.37               Chg-rec  EDP-UOW=AUSL  00000000546800000000 LogRBA=CD1FCF19F47B6D63
 19:27:42.37                        ECCR-UOW=AUSPRT01 E6D9C2D7D9E3F0F1 008F803000000001
 19:27:42.37                        Source=VSM Func=ISRT Srcname=VSAMEDM.VSAM.KSDS01
 19:27:42.37                        Timestamp-19:25:40:81 Date-04/29/2014
 19:27:42.37               Chg-rec  EDP-UOW=AUSL  00000000546800000000 LogRBA=CD1FCF19F47D3D94
 19:27:42.37                        ECCR-UOW=AUSPRT01 E6D9C2D7D9E3F0F1 008F803000000001
 19:27:42.37                        Source=VSM Func=ISRT Srcname=VSAMEDM.VSAM.KSDS01
 19:27:42.37                        Timestamp-19:25:41:03 Date-04/29/2014
 19:27:42.37               PH1-pkt  EDP-UOW=AUSL  00000000546800000000 LogRBA=CD1FCF19F47D4581
 19:27:42.37                        ECCR-UOW=AUSPRT01 AUSPRT01 E6D9C2D7D9E3F0F1 008F803000000001
 19:27:42.37                        Timestamp-19:25:41:39 Date-04/29/2014
 19:27:42.37               Com-pkt  EDP-UOW=AUSL  00000000546800000000 LogRBA=CD1FCF19F47F0362
```

```
 19:27:42.37                         ECCR-UOW=AUSPRT01 AUSPRT01 E6D9C2D7D9E3F0F1 008F803000000001
 19:27:42.37                         Timestamp-19:25:41:39 Date-04/29/2014
 19:27:42.37                Srv-rec  EDP-UOW=AUSPRTF0F140404040404040 LogRBA=CD1FCF19F47E2049
 19:27:42.37                Srv-rec  EDP-UOW=AUSPRTF0F140404040404040 LogRBA=CD1FCF19F48B60C0
 19:27:42.37                Log-rec  EDP-UOW=AUSL  00000000634900000001 LogRBA=CD1FCF19F48F240A
  . . . . .
  . . . . .
  . . . . .
 19:27:43.63                Beg-pkt  EDP-UOW=AUSL  0000001936AD00000000 LogRBA=CD1FCF19F48F8030
 19:27:43.63                         ECCR-UOW=AUSDB201 01       F0F1000000000001 605FF6E000000000
 19:27:43.63                         Timestamp-19:27:19:68 Date-05/07/2014
 19:27:43.63                Chg-rec  EDP-UOW=AUSL  0000001936AD00000000 LogRBA=CD1FCF19F493A0D2
 19:27:43.63                         ECCR-UOW=01       F0F1000000000001 605FF6E000000000
 19:27:43.63                         Source=DB2 Func=UPDT Srcname=DB2DSNBtenchar1
 19:27:43.63                         Timestamp-19:27:19:68 Date-05/07/2014
 19:27:43.63                Chg-rec  EDP-UOW=AUSL  0000001936AD00000000 LogRBA=CD1FCF19F4950D34
 19:27:43.63                         ECCR-UOW=01       F0F1000000000001 605FF6E000000000
 19:27:43.63                         Source=DB2 Func=UPDT Srcname=DB2DSNBtenchar1
 19:27:43.63                         Timestamp-19:27:19:68 Date-05/07/2014
 19:27:43.63                Com-pkt  EDP-UOW=AUSL  0000001936AD00000000 LogRBA=CD1FCF19F497F385
 19:27:43.63                         ECCR-UOW=AUSDB201 01       F0F1000000000001 605FF6E000000000
 19:27:43.63                         Timestamp-19:27:40:69 Date-05/07/2014
 19:27:48.67  DTLEDM172198I EDMLPOPU: LMF table populate tasks are terminating due to an operator stop or PAC termination
 19:27:48.70  DTLEDM172195I EDMLUPLU: LMF task for Log Scan/Print Utility is terminating due to an operator stop or PAC
termination
 19:27:48.73  Totals by Source.............
 19:27:48.73                VSAMEDM.VSAM.KSDS01                 Isrt= 5 Repl= 0 Dlet= 0 Unk= 0
 19:27:48.73                VSAMEDM.QA.EDMABC04                 Isrt= 0 Repl= 1000 Dlet= 0 Unk= 0
 19:27:48.73                VSAMEDM.QA.EDMABC07                 Isrt= 1000 Repl= 0 Dlet= 1000 Unk= 0
 19:27:48.73                DB2DSNBtenchar1                     Isrt= 2 Repl= 2 Dlet= 2 Unk= 0
 19:27:48.73  Open Uows.....................
 19:27:48.74  L O G   E N D
```

CHAPTER 16

EDMXLUTL - Event Marker Utility

This chapter includes the following topics:

EDMXLUTL Utility Overview

Use the EDMXLUTL utility to create an event marker in your PowerExchange Logger for z/OS.

Creating an Event Marker in Batch Mode

Use the following procedure to create an event marker in batch mode.

To create an event marker in batch mode:

1. Make a working copy of the #`EDMLUTB` sample JCL from the `HLQ.SAMPLIB` sample library, where `HLQ` is the high-level qualifier specified at installation, and edit the copy as required.
2. Run the job to create the event marker.

EDMXLUTL Utility JCL Statements

The following table describes the JCL statements for the EDMXLUTL utility:

Statement	Description
EXEC	Specify the EDMXLUTL program.
STEPLIB DD	Include the PowerExchange Change Capture load library. If you added the load library to your system's LNKLST concatenation, you do not need to add it to the STEPLIB.
EDMPARMS DD	Specify the name of the user library (YOUR.USERLIB) that contains the default options module (EDMSDIR) associated with the PowerExchange Logger you are using. If you do not include an EDMPARMS DD statement, or if you specify a library that does not contain the options modules, PowerExchange Change Capture uses the STEPLIB concatenation to obtain the configuration options.
EDMMSG DD	Specify the data set name to which you want to issue errors and warnings.
EDMSYSIN DD	Specify the appropriate EVENT command for the marker that you want to create.

EDMXLUTL Utility Control Statements

The following table lists the control statements for the event-marker utility:

Command
EVENT TYPE=BASEEDM NOTIFY={EDITION|ENDCOPY|COPY} OBJECT={IMS|VSAM|DB2} ACCESS=STRUCTURE {DBD={*dbd_name* DSN=*data_set_name*|SYSID=*ssid*}
EVENT TYPE=BASEEDM NOTIFY={EDITION|ENDCOPY|COPY} OBJECT={IMS|VSAM|DB2} ACCESS=OBJECT {EDMNAME=*edmname*|DBD=*dbd_name*} DSN=*data_set_name* SEGMENT=*segment_name* [SEGMENT=*segment_name* ...]|DBD=*dbd_name* DSN=*data_set_name*|SYSID=*ssid* CREATOR=*table_creator* TABNAME=*table_name* [TABNAME=*table_name* ...] }

Related Topics:

- "EVENT Command Syntax" on page 203
- "Keyword Sets for the BASEEDM Category" on page 203

EDMXLUTL Utility EVENT Command

Use the EVENT command to create event markers in batch mode.

EVENT Command Syntax

Use the following syntax for the EVENT command:

```
EVENT TYPE=category keyword1=value1 keyword2=value2
      keyword3=value3 ...
```

Subsequent sections discuss the parameters for this command, by category. Each category has one or more sets of keywords associated with it.

EVENT Command Usage

To use this command, include it as a control statement in a batch job. Then, run the job to create the event marker. The following rules apply to specifying this control statement:

- Your statement should be contained within columns 1 through 71.
- If your statement will not fit in this range, you must have a character in column 72 to indicate that your statement continues on more than one line.
- A statement that continues on more than one line must contain only a single command.
- Continued statements must begin in column 1, if column 71 on the previous line is blank.
- A statement can use up to a 38 lines.
- You can use a maximum of 255 blanks to separate commands and keywords.

The following additional information is listed for this command:

- Before you run a job to create an event marker, be sure that the PowerExchange Logger is active.
- A PowerExchange Logger failure could cause the logger to stop while running an event marker job. In this case, the control statements processed prior to the failure are still accepted. Conversely, the control statement that is in progress when the PowerExchange Logger fails, and the subsequent control statements, cause the event marker utility to abend.
- Take care if running this command while the PowerExchange active log is receiving other log records for the source object that the marker affects. This can mix the event marker in with the other records, producing unexpected results.
- When the utility successfully records the event marker record in the PowerExchange log, the utility displays message DTLEDM175016I. This message provides the RBA of the event marker record within the log. You may need the RBA to reference that record.
- This utility obtains the name of the PowerExchange Logger that it accesses from the default options module, `EDMSDIR`.

Keyword Sets for the BASEEDM Category

Use the BASEEDM category to create a special event record in the PowerExchange active log. This section describes the two keyword sets that you can use with the BASEEDM category:

- MARK
- NOTIFY

MARK Keyword Set

The MARK keyword set tells the event-marker utility to insert a special marker into the PowerExchange Logger for z/OS active logs. The marker returns a log address and passes a signal to a component that uses PowerExchange Logger data.

Note: Use the MARK keyword set only at the direction of Informatica Global Customer Support.

Syntax:

```
EVENT TYPE=BASEEDM MARK=type DATA=text
```

Example:

```
EVENT TYPE=BASEEDM MARK=EOL DATA='my text'
```

The following table describes the keywords that you can use in place of the variable for the MARK statement:

Variable	Keyword Description
type	Tells the utility the type of event marker to add to the log. The following keywords are valid: - EOD. Creates an event marker that indicates that the end of day has been reached.§ - SIGNAL. Creates an event marker that indicates a starting point within the log or that passes a signal to a component that uses PowerExchange logger data. - EOL. Creates an event marker that indicates the end of the log. The utility places the marker at the current end of the PowerExchange active log. For the utility to identify the precise end of the log, the PowerExchange Logger should not receive any other records.
text	Enter up to 30 characters of text that you want the utility to add to the event marker record. If you include embedded blanks, you must enclose the text in single quotation marks (').

NOTIFY Keyword Set

This set of keywords tells the utility to insert a special marker into the PowerExchange active log. The special marker notifies the component using the data of an event change, such as a change in the edition value.

This is used to generate a restart point in the PowerExchange Change Capture log.

Syntax:

```
For ACCESS=STRUCTURE:
EVENT TYPE=BASEEDM NOTIFY=type OBJECT=database_type
      ACCESS=STRUCTURE {DBD=database_name DSN=data_set_name |
      SYSID=ssid}
For ACCESS=OBJECT:
EVENT TYPE=BASEEDM NOTIFY=type OBJECT=db_type
      ACCESS=level_of_data_objects
      {EDMNAME=edmname |
        DBD=database_name DSN=data_set_name SEGMENT=segment_name
        [SEGMENT=segment_name ...] | DBD=database_name DSN=data_set_name |
        SYSID=ssid CREATOR=tbcreator TABNAME=table_name
        [TABNAME=table_name ...]}
```

The following table lists and describes the variables that you can use with the BASEEDM category:

Variables	Description
type	Tells the utility what type of notification the event marker signals. The following value is valid: - EDITION provides notification that a resource registration is changing.
db_type	Indicates the database type of the associated resource. The following values are valid: - IMS - VSAM - DB2
level_of_data_objects	Indicates the level of data objects to be associated with the notification. The following values are valid: - STRUCTURE indicates that all data objects within the database, data set, or subsystem are to be associated with the notification. When you specify ACCESS=STRUCTURE, you must specify either the DBD and data set name or the subsystem ID. For example, for OBJECT=IMS, you would specify DBD and DSN. - OBJECT indicates that only the specified object is to be associated with the notification. When you specify ACCESS=OBJECT, you can specify either the EDMNAME or the fully qualified data object name. For example, for OBJECT=IMS, you would specify DBD, DSN, and SEGMENT.
edmname	You can specify a particular registered source segment, record, or table by using its EDMNAME. This variable supports delimited strings, but you must enclose them in quotation marks.
dbdname	When used alone, allows you to specify the database description (DBD) name of a set of IMS segments or VSAM records. When you use the DBD name as part of a fully qualified name, this name allows you to specify a particular IMS segment or VSAM record.
data_set_name	Specifies the data set name of a particular IMS segment or VSAM record as part of a fully qualified name.
segment_name	Specifies a particular IMS segment as part of a fully qualified name. You can use this variable multiple times (up to 255) in a single statement to associate multiple segments with the notification.
ssid	You can specify the subsystem ID of a particular set of DB2 tables when used alone, or a particular DB2 table when used as part of a fully qualified name
tbcreator	Specifies the creator of a particular DB2 table as part of a fully qualified name. This variable supports delimited strings, but you must enclose them in quotation marks. **Note:** tbcreator cannot handle DB2 long names and is limited to 8 bytes.
table_name	Specifies a particular DB2 table as part of a fully qualified name. You can use this variable multiple times (up to 255) in a single statement to associate multiple tables with the notification. These tables must be in the same subsystem and have the same creator ID.This variable supports delimited strings, but you must enclose them in quotation marks. **Note:** table_name cannot handle DB2 long names and is limited to 18 bytes.

If the DB2 ECCR is active when you run the create-event-marker utility to update the edition level, you must refresh the ECCR. To do so, run the `MODIFY` job_name,`REFRESH` command (where job_name is the name of the MVS batch job or started task that runs the DB2 ECCR). This ensures that the DB2 ECCR reads the new edition level in the PowerExchange repository.

Note: Alternatively, you can stop and restart the DB2 ECCR with the `WARM START` keyword.

EDMXLUTL Utility Example

The following example JCL creates an event marker when the edition level changes. You can find this example in the `#EDMLUTB` member of the `HLQ.SAMPLIB` sample library (where `HLQ` is the high-level qualifier specified at installation).

```
//         JOB
//*--------------------------------------------------------------------*
//* DETAIL Change Capture - EVENT MARKER UTILITY TO CREATE SPECIAL EVENT
//*                  RECORD TO REFLECT A CHANGE IN EDITION LEVELS
//*--------------------------------------------------------------------*
//* REPLACE THE FOLLOWING ITEMS WITH PROPER INSTALLATION VALUES
//* 1. JCL DATA SET NAMES
//* 2. EDMSYSIN DD CONTROL CARD
//*--------------------------------------------------------------------*
//EDMUTIL  EXEC PGM=EDMXLUTL
//STEPLIB  DD DISP=SHR,DSN=HLQ.LOAD              <=== CDM LOADLIB
//EDMPARMS DD DISP=SHR,DSN=YOUR.USERLIB          <=== EDMSDIR,EDMUPARM
//EDMMSG   DD SYSOUT=*
//EDMSYSIN DD *
EVENT TYPE=BASEEDM NOTIFY=EDITION OBJECT=DB2 ACCESS=OBJECT             X
      EDMNAME=EDM.EDMNAME1
/*
```

The following lines show the messages that result after you run the create-event-marker utility.

```
Sample Messages for the Create an Event Marker Utility

DTLEDM175015I Control card read from EDMSYSIN
*
*  Do EVENT mark for EDMNAME=VSAM.API.SOURCE
*
EVENT                                                                  -
       TYPE=BASEEDM                                                    -
     NOTIFY=ENDCOPY                                                    -
     OBJECT=IMS                                                        -
     ACCESS=OBJECT                                                     -
    EDMNAME=VSAM.API.SOURCE
DTLEDM175015I Executing EVENT command; command messages may follow.
    Event type=BASEEDM
DTLEDM175025I Event Mark Notify=ENDCOPY Summary:
    Event Mark Logger RBA . . . . . :C4C7D2D340400000001E466400000000
    Event Sequence number . . . . . : 0000001E466400000000
    Event Edition number. . . . . : B42B13970E162802
    Event Source EDMNAME  . . . . . : VSAM.API.SOURCE
    Related Target EDMNAME  . . . . . : DB2.DEAG.RDADGK.APITARGET
```

CHAPTER 17

HOSTENT - TCP/IP Address Reporter Utility

This chapter includes the following topics:

HOSTENT Utility Overview

Use the HOSTENT utility to:

- Display the TCP/IP host name and address for a system.
- Diagnose problems with PowerExchange communication.

Supported Operating Systems for the HOSTENT Utility

The HOSTENT utility can run on the following operating systems:

- i5/OS
- Linux and UNIX
- z/OS

Running the HOSTENT Utility on i5/OS

To run the HOSTENT utility on i5/OS:

- Enter the following command:

```
CALL HOSTENT
```

Running the HOSTENT Utility on Linux and UNIX

To run the HOSTENT utility on Linux and UNIX:

- Enter the following command:

```
hostent
```

Running the HOSTENT Utility on z/OS

Use the version of the HOSTENT TCP/IP Address Reporter utility for your TCP/IP environment.

The following table lists the HOSTENT versions by type of TCP/IP environment:

HOSTENT Version	Environment
HOSTENT	Standard z/OS Communications Server
HOSTENT2	Computer Associates CA-TCPAccess Communications Server
HOSTENT3	Native MVS Sockets

You can run the HOSTENT utility from the TSO/E command line or by submitting a z/OS job.

To run the HOSTENT utility from the command line, use the following statement:

```
call 'hlq.LOADLIB(HOSTENT)'
```

Use the sample JCL in the HOSTENT member of the RUNLIB library to create a job to run the utility. The sample JCL contains the following statements:

```
//STEP1   EXEC PGM=HOSTENT
//STEPLIB  DD  DSN=&SCERUN,DISP=SHR
//         DD  DSN=&HLQ..LOADLIB,DISP=SHR
//SYSPRINT DD  SYSOUT=*
//
```

JOB

Initiates the job.

EXEC PGM=HOSTENT

Invokes the utility.

STEPLIB DD

Defines the PowerExchange LOAD library that contains the utility.

SYSPRINT DD

Defines the print location for the report.

HOSTENT Utility Usage Notes

Consider the following points before using the HOSTENT utility:

- PowerExchange uses the TCP/IP resolver to translate the host name of the TCP/IP stack into an IP address.
- On z/OS and OS/390 the resolver queries the local host table. On i5/OS, Linux, and UNIX, the resolver queries the name server before it queries the local host table.
- On i5/OS, z/OS, and OS/390, PowerExchange uses the primary interface address of the TCP/IP stack to verify the licence if the resolver cannot find the host name.
- Operating systems can run more than one TCP/IP stack. Ensure that the HOSTENT runs on the TCP/IP stack that is used by PowerExchange. You cannot specify a stack name in the HOSTENT parameters.

HOSTENT Utility Resolver Details

The resolver uses the local site tables to look up the official host name and address. The resolver does not use name servers.

For z/OS 1.2 or later, you can add the following DD statement to the HOSTENT JCL to get a resolver trace to assist in diagnosis:

```
//SYSTCPT  DD  SYSOUT=*
```

This reports the configuration data sets and methods of look-up that the resolver uses.

HOSTENT Utility Output

The following table describes the output messages generated by HOSTENT:

Operating System	Message	Description
i5/OS, Linux, UNIX, z/OS	gethostname() gives *host name*	Displays the host name of the TCP/IP stack. On z/OS and OS/390 systems, you can find the gethostname( ) details in the TCPIP.DATA file specified in the TCP/IP stack. On i5/OS, Linux, and UNIX, you can find the gethostname() details in the TCPIP.DATA file used by PowerExchange.
i5/OS, Linux, UNIX, z/OS	official hostname *host name.domain name*	Displays the host name returned by the resolver. The resolver looks up the given host name to find the fully qualified name including domain name. This also displays: - Alias names found by the resolver. - TCP/IP address as returned by the resolver. PowerExchange uses this address to validate the license.
i5/OS, Linux, UNIX, z/OS	reporting on hostname *host name*	Displays the host name. The TCP/IP resolver uses the following methods to find the host name: - Looks up the host name from a local hosts file. - Uses the gethostbyname() system call to look up host names from a name server. PowerExchange uses these details to validate the license.
i5/OS, z/OS	gethostid() gives: *nnn.nnn.nnn.nnn*	Displays the primary interface address of the TCP/IP stack. If the TCP/IP resolver cannot find the host name, PowerExchange uses this address to validate the license. On z/OS or OS/390 systems, the gethostid() details are specified in the TCP/IP stack in the PRIMARYINTERFACEADDRESS parameter of the PROFILE data set.
z/OS	resolver gives domainname: *domain name*	Displays the domain name as determined by the local resolver configuration data set. PowerExchange does not use this address to validate the license.
z/OS	resolver gives hostname : *host name*	Displays the host name as determined by the local resolver configuration data set. PowerExchange does not use this address to validate the license.

HOSTENT Utility on i5/OS Example

The following command displays the TCP/IP host address and host name of the system on which it was run:

```
CALL HOSTENT
```

The resulting output is:

```
gethostid() gives: nnn.nnn.nnn.nnn
  gethostname() gives host name
  reporting on hostname host name
  official hostname: host name
    address: nnn.nnn.nnn.nnn
```

HOSTENT Utility on Linux and UNIX Example

The following command displays the TCP/IP host address and host name of the system on which it was run:

```
hostent
```

The resulting output is:

```
gethostname() gives host name
reporting on hostname host name
official hostname: host name
  address: nnn.nnn.nnn.nnn
```

HOSTENT Utility on z/OS Example

The following statement displays the TCP/IP host address and host name of the system on which it was run:

```
//STEP1   EXEC PGM=HOSTENT,
//             PARM='/'
//STEPLIB  DD  DSN=&SCERUN,DISP=SHR
//         DD  DSN=&HLQ..LOADLIB,DISP=SHR
//SYSOUT   DD  SYSOUT=*
//SYSPRINT DD  SYSOUT=*
```

The resulting output is:

```
HOSTENT:
gethostid() gives: nnn.nnn.nnn.nnn
resolver gives hostname  : host name
resolver gives domainname: domain name
gethostname() gives host name
reporting on hostname host name
official hostname: host name.domain name
  alias: host name
  address: nnn.nnn.nnn.nnn
```

CHAPTER 18

PWXUCDCT - Logger for Linux, UNIX, and Windows Utility

This chapter includes the following topics:

PWXUCDCT Utility Overview

Use the PWXUCDCT utility to manage files and print reports for the PowerExchange Logger for Linux, UNIX, and Windows.

With the utility, you can perform the following tasks:

- Manually back up the CDCT file if the backups that are automatically generated at PowerExchange Logger initialization and termination are not available or not recent.
- Derive a CDCT file backup that can be used for a restore operation based on the PowerExchange Logger log files.
- Restore the CDCT file from a backup.
- Delete expired CDCT records and any PowerExchange Logger log files associated with those records.
- Delete orphaned PowerExchange Logger log files that are not referenced by any CDCT record.
- Print reports on PowerExchange Logger pwxccl configuration parameters, the CDCT file contents, and current, orphaned, and expired log files.

For more information about the PowerExchange Logger, see the *PowerExchange CDC Guide for Linux, UNIX, and Windows*.

Supported Operating Systems for the PWXUCDCT Utility

The PWXUCDCT utility runs on computers that have the following types of operating systems:

- Linux
- UNIX
- Windows

For more information about supported operating systems, see the *PowerExchange Installation and Upgrade Guide*.

Control Statement Syntax for PWXUCDCT Commands

Use the following general syntax to specify control statements for the PWXUCDCT utility:

```
PWXUCDCT
    CMD=command_name
    [CONFIG=override_dbmover.cfg]
    [CS=override_pwxccl.cfg]
    [LICENSE=override_license.key]
    command-specific parameters
```

The following syntax rules apply:

- You can specify the optional CONFIG, CS, and LICENSE parameters on any command. Informatica recommends that you specify the CS parameter because each command must run under a specific PowerExchange Logger configuration for a database instance. If you do not specify a PowerExchange Logger configuration file, the PowerExchange Logger uses the pwxccl file in the top-level installation directory.
- Other parameters are specific to the command that is being issued. You can enter these command-specific parameters in any order.
- You cannot define the parameters in a separate file and then reference that file in the command syntax.

Related Topics:

- "PWXUCDCT Commands and Parameters" on page 213
- "Usage Notes for the PWXUCDCT Utility" on page 220
- "Examples of PWXUCDCT Utility Commands" on page 220

PWXUCDCT Commands and Parameters

This section describes the commands that you can enter in the CMD statement of the PWXUCDCT syntax and the command-specific parameters.

It also describes the CONFIG, CS, LICENSE parameters that you specify for any command.

Commands

This topic summarizes the commands that you can issue to the PWXUCDCT utility, including any command-specific parameters.

The following table describes each command:

Command	Description	Command-specific Parameters[1]
CONVERT_CDCT	If you upgrade to 9.5.1 HotFix 1 or later from an earlier release, you can issue this command to manually perform a one-time conversion of the CDCT file to the new format. Alternatively, the first time the PowerExchange Logger is warm started, it automatically converts the CDCT file to the new format. The conversion creates a CDCT_*dbid* file instance from the original CDCT file. Ensure that the *dbid* value in the CDCT file name matches the DBID parameter value in the PowerExchange Logger pwxccl configuration file under which you run the command. **Note:** If the old CDCT file contains information for multiple database instances, you must run this command multiple times, once for each instance. Each time you run the command, ensure that the CS parameter points to the correct pwxccl configuration file for the instance.	None
CREATE_CDCT_BACKUP	Manually creates a backup of all records in a CDCT file instance for a source database based on the latest configuration incarnation. The *dbid* value in the CDCT file name must match the DBID parameter value in the pwxccl configuration file. **Note:** The PowerExchange Logger automatically generates a backup at initialization and at termination.	BACKUPFILE
DELETE_EXPIRED_CDCT	This command is deprecated but is still supported for backward compatibility. Use DELETE_EXPIRED_FILES instead.	None
DELETE_EXPIRED_FILES	Deletes the log files for which the retention period has expired and the CDCT records that reference those expired logs. For this command to work, you must set the LOGGER_DELETES_EXPIRED_CDCT_RECORDS parameter to N in the pwxccl configuration file. If you set the LOGGER_DELETES_EXPIRED_CDCT_RECORDS parameter to Y, or if you do not specify this parameter, the command does not work.	None
DELETE_ORPHAN_FILES	Deletes PowerExchange Logger log files that are not referenced by any record in the CDCT file.	None

Command	Description	Command-specific Parameters[1]
DERIVE_CDCT_BACKUP	If the CDCT file is corrupted or deleted, and if a recent CDCT backup is not available or the latest available backup would result in significant reprocessing of data, use this command to derive a backup file for recovery purposes. The command uses the EXTERNAL_CAPTURE_MASK parameter value from the PowerExchange Logger configuration file or the *external_capture_mask* positional parameter from the group definition file to generate a list of PowerExchange Logger log files. The command then uses the content of these log files to generate a text file that can be used as input to the RESTORE CDCT command. Do not use this command if the PowerExchange Logger log files were also corrupted or deleted. **Tip:** Use the PREVBACKUPFILE parameter to supply the name of the last available backup file. By using a previous backup file, you preserve more historic information in the CDCT file. The utility will add any log files that were created since the backup was taken to the derived backup file.	BACKUPFILE [PREVBACKUPFILE]

Command	Description	Command-specific Parameters[1]
REPORT_CDCT	Prints the contents of the CDCT file. This information is primarily for debugging purposes. For the current Logger configuration incarnation, the report shows: - Incarnation identifier, status, and reason (Rsn) for creation. The reason can be a cold start or a change in the configuration. - Source instance (or DBID) name and image type. - Number of groups defined in the group definition file. If no groups are defined, the default of 1 is used. - Type of AES encryption algorithm, if log file encryption is enabled. - Begin and end restart and sequence tokens. For each Logger group, the report shows: - Group number and name. - Incarnation to which the group belongs. - Path to the group log files. - Registration count. - Log file count, and the first and current log sequence numbers. - Status - Oldest log file timestamp. For each registration, the report shows: - Registration tag name and status. - Incarnation and group to which the registration belongs. - Activation date and inactivation date, if available. - Default schema name. For each PowerExchange Logger log file, the report shows: - Log file path and file name, and sequence number. - Configuration incarnation and group to which the log file belongs. - File open and close timestamps. - Record count, commit count, and whether the log file contains uncommitted data. - Whether the log file is encrypted, and a value used to test the encryption key. - Status and Fmt version. - Begin and end restart tokens.	[*report_file_name*]
REPORT_CDCT_FILES	Reports the following information for each PowerExchange Logger log file that is recorded in the CDCT file: - Log file path and file name, and sequence number. - Configuration incarnation and group to which the log file belongs. - File open and close timestamps. - Record count, commit count, and whether the log file contains uncommitted data. - Whether the log file is encrypted, and a value used to test the encryption key. - Status and Fmt version. - Begin and end restart tokens. This information is the same that reported in the CCL Files section of the REPORT_CDCT report.	[*report_file_name*]

Command	Description	Command-specific Parameters[1]
REPORT_CONFIG	Lists the parameter settings that are defined in the associated PowerExchange Logger pwxccl configuration file. If you created a group definition file and specified it in the GROUPDEFS parameter in the pwxccl file, the command also reports the group statements in the group definition file.	[*report_file_name*]
REPORT_EXPIRED_CDCT	This command is deprecated but is still supported for backward compatibility. Use REPORT_EXPIRED_FILES instead.	[*report_file_name*]
REPORT_EXPIRED_FILES	Lists the PowerExchange Logger log files for which the retention period has elapsed.	[*report_file_name*]
REPORT_FILES_BY_NAME	Lists PowerExchange Logger log files by file name. This information is based on directory information for the log files and not on the CDCT file. For each file, the command reports the following information: - Date and time when the file was written. - Sequence number - Path and file name. Also, the command reports the number of log files that match the default mask that is specified in the EXT_CAPT_MASK parameter of the pwxccl configuration file. If you specified a group definition file in the GROUPDEFS parameter of the pwxccl file, the command reports the number of log files that match any masks in the group definition file. **Note:** The PowerExchange Logger generates log file names that include the EXT_CAPT_MASK value, the date and time, and a sequential number. For example: MYMASK.CND.CP090813.T1748013, where 090813 is MMDDYY, 1748 is HHMM, and 013 is the generated sequential number.	[*report_file_name*]
REPORT_FILES_BY_TIME	Lists PowerExchange Logger log files in the order in which they were created, from earliest to latest. This information is based on directory information for the log files and not on the CDCT file. For each file, the command reports the following information: - Date and time when the file was written. - Sequence number of the file. - Path and file name. Also, the command reports the number of log files that match the default mask that is specified in the EXT_CAPT_MASK parameter of the pwxccl configuration file. If you specified a group definition file in the GROUPDEFS parameter of the pwxccl file, the command also reports the number of log files that match any masks in the group definition file.	[*report_file_name*]
REPORT_ORPHAN_FILES	Lists PowerExchange Logger log files that are not referenced by any record in the CDCT file.	[*report_file_name*]

Command	Description	Command-specific Parameters[1]
RESTORE_CDCT	Restores the CDCT file from a backup, up to a specific point in time. The PowerExchange Logger will reprocess any data that is later than this point in time. After the restore operation completes, run the DELETE_ORPHAN_FILES command.	BACKUPFILENAME [ENCRYPTEPWD] [PROGRESSFREQUENCY]
1. Optional parameters are enclosed in square brackets.		

Also, you can specify the following global parameters on any PWXUCDCT utility command: CONFIG, CS, and LICENSE. Informatica recommends that you include the CS parameter in each command statement because each command must run under a specific PowerExchange Logger configuration. Otherwise, the utility uses the default PowerExchange Logger configuration file in the installation directory.

Parameter Descriptions

The following the global and command-specific parameters are used with PWXUCDCT utility commands.

BACKUPFILE=*path\file_name*.txt

Specifies the full path and file name for a CDCT backup file that you are creating or using for a restore operation. The backup file is a delimited text file.

Required in the following commands: CREATE_CDCT_BACKUP, DERIVE_CDCT_BACKUP, and RESTORE_CDCT.

CONFIG=*path\file_name*

If you specified the CONFIG parameter in the pwxccl statement when starting the PowerExchange Logger process, specify the same parameter value in the PWXUCDCT utility command for that process. The parameter specifies the full path and file name for a DBMOVER configuration file that overrides the default dbmover configuration file in the installation directory. The full path is required only if the override file does not reside in the default location. The override file takes precedence over any other override configuration file that you optionally specify with the PWX_CONFIG environment variable.

For example, you might use an override DBMOVER configuration file to split PowerExchange Logger processing across multiple database instances but maintain a separate CDCT file for each instance.

Optional in any PWXUCDCT utility command.

CS=*path\file_name*

If you specified the CS parameter in the pwxccl statement when starting the PowerExchange Logger process, specify the same parameter value in the PWXUCDCT utility command for that process. The parameter specifies the full path and file name of the PowerExchange Logger configuration file. If you specify either the CONFIG or LICENSE parameter, the CS parameter is required. You can use the CS parameter to specify a PowerExchange Logger configuration file that overrides the default pwxccl configuration file in the installation directory. The full path is required only if the override file does not reside in the default location.

Recommended in all PWXUCDCT utility commands.

ENCRYPTEPWD=*encrypted_encryption_password*

If you cold started the PowerExchange Logger from the command line with a pwxccl command that included the encryptepwd parameter, you must specify that same parameter value in the ENCRYPTEPWD

parameter in the RESTORE_CDCT command. The parameter specifies an encryption password, in encrypted format, that enables the encryption of PowerExchange Logger log files. With this password, the command can restore the CDCT file, including the encryption password that is stored in the file in encrypted format.

Tip: After you run the RESTORE_CDCT command, perform a CAPX database row test from the PowerExchange Navigator to verify that the encryption password has been successfully restored.

LICENSE=*path\file_name*

If you specified the LICENSE parameter in the pwxccl command when starting the PowerExchange Logger process, specify the same parameter value in the PWXUCDCT utility command for that process. The parameter specifies the full path and file name for a license key file that overrides the default license.key file in the installation directory. The full path is required only if the override file does not reside in the default location. The override file takes precedence over any other override license key file that you optionally specify with the PWX_LICENSE environment variable.

Optional in all PWXUCDCT commands.

PREVBACKUPFILE=*file_name*

Specifies a previous backup file to use with the DERIVE_CDCT_BACKUP command to derive a backup that is suitable for a restore operation. Use this parameter when a recent backup is not available after a failure or the latest available backup would result in significant reprocessing of data. You can use a backup file that was automatically generated at PowerExchange Logger initialization or shutdown or that you manually created. By using a previous backup file, you preserve more historic information in the CDCT file. Also, the utility will roll the CDCT contents forward based on any log files it discovers that are later than the backup point in time.

PROGRESSFREQUENCY=*number_of_records*

Specifies the frequency with which the PWXUCDCT utility displays progress information for a RESTORE_CDCT operation. The frequency is expressed as the number of records read from the CDCT backup file. Each time the utility processes this number of records, it writes progress message PWX-25132 to the console screen and PowerExchange message log. Default is to display progress information each time the utility processes approximately 1 percent of the records in the backup file.

report_file_name

Specifies a path and file name that you can specify to send report output to a file instead of to the command line screen. In the command, precede this value with a greater than (>) sign, for example:

```
>C:\Informatica\PowerExchange9.5.1\reports\expiredcdct01.txt
```

Optional in any PWXUCDCT REPORT command.

For more information about the PWXUCDCT commands, see "Commands" on page 214.

Running the PWXUCDCT Utility

You can run a PWXUCDCT utility command from a command line on a Linux, UNIX, or Windows system where PowerExchange is installed.

Make sure that you run the utility under a user ID that has READ access to the PowerExchange Logger log files and control files.

To run the utility, navigate to the directory where the pwxucdct executable is located. By default, this directory is in the PowerExchange installation directory. Then enter `pwxucdct` followed by a command name and any relevant command parameters. Use the following syntax:

```
C:\Informatica\PowerExchangev.r.m pwxucdct [parameter1 parameter2...] cs=pwxccl_config
CMD=command_name
```

Usage Notes for the PWXUCDCT Utility

Before you use the PWXUCDCT utility, review the following usage notes:

- You can schedule PWXUCDCT utility commands to run during off-peak hours to avoid increasing the PowerExchange Logger workload when transaction activity is high.
- If you run the PowerExchange Logger in continuous mode, do not use the CREATE_CDCT_BACKUP command to back up the CDCT file while it is being updated.
- By default, the PWXUCDCT utility writes output from a REPORT command to stdout so that you see it on screen. To send the output to a file, you must specify a *report_file_name* preceded by a greater than (>) sign, for example, `>C:\Informatica\PowerExchange9.0.0\reports\myfile.txt`. Otherwise, the PWXUCDCT utility scrolls the report lines onto the screen. For some REPORT commands, the utility also writes report messages to the PowerExchange message log but does not include the detailed lines that are written to stdout.
- All PWXUCDCT utility commands other than REPORT_FILES_BY_TIME and REPORT_FILES_BY_NAME reference the CDCT file.
- The CDCT file name includes the source instance name that is specified in the DBID parameter of the pwxccl configuration file. The CDCT location is determined by the CAPT_PATH parameter in the dbmover configuration file. The location can be expressed as $(CAPT_PATH)/CDCT_$(DBID).

Examples of PWXUCDCT Utility Commands

This section provides example PWXUCDCT utility commands and shows sample output where appropriate. Enter the commands from a command line.

Example 1. Creating a Backup of the CDCT File

The backups that PowerExchange automatically generated during PowerExchange Logger initialization and shutdown are not available. To create a backup of all records in the CDCT file manually, you run the CREATE_CDCT_BACKUP command during daily batch processing.

Enter the following command:

```
pwxucdct cmd=create_cdct_backup backupfile=C:\Informatica\PowerExchangev.r.m\backups
\backup1.txt cs=C:\Informatica\PowerExchangev.r.m\resources\pwxccl_orcl.cfg
```

If the command is successful, the following messages are written to the message log file:

```
PWX-33314 TIMEOUTS configuration parameter is deprecated
PWX-33269 CCL configuration parameter <CHKPT_BASENAME> is deprecated.
PWX-33269 CCL configuration parameter <CHKPT_NUM> is deprecated.
PWX-25404 Processing console program. pwxucdct cmd=create_cdct_backup backupfile=C:\Informatica
\PowerExchangev.r.m\backups\backup1.txt cs=C:\Informatica\PowerExchangev.r.m\resources\pwxccl_orcl.cfg
```

The backup1.txt file is created in the backups directory.

Tip: You can use this backup file to restore the CDCT file, if necessary. In the RESTORE_CDCT command, use the backupfile parameter to specify the backup file name and path.

Example 2. Restoring the CDCT File from a Backup File

The CDCT file has become damaged. You want to restore it from its latest backup file. You created the backup file with the CREATE_CDCT_BACKUP command based on the latest Logger configuration incarnation.

From the command line, navigate to the PowerExchange installation directory and enter the following command:

```
pwxucdct cmd=restore_cdct backupfile=C:\Informatica\PowerExchangev.r.m\backups
\backup1.txt cs=C:\Informatica\PowerExchangev.r.m\resources\pwxccl_orcl.cfg
```

If the command is successful, the following messages are written to the message log file:

```
PWX-33314 TIMEOUTS configuration parameter is deprecated
PWX-33269 CCL configuration parameter <CHKPT_BASENAME> is deprecated.
PWX-33269 CCL configuration parameter <CHKPT_NUM> is deprecated.
PWX-25404 Processing console program. pwxucdct cmd=restore_cdct backupfile=C:Informatica\PowerExchangev.r.m
\backups\backup1.txt cs=C:\Informatica\PowerExchangev.r.m\resources\pwxccl_orcl.cfg
PWX-25200 Created CDCT file "C:\Informatica\PowerExchangev.r.m\resources\CDCT_orcl"
PWX-36937 Restore is using backup of CDCT file <C:\Informatica\PowerExchangev.r.m\resources\CDCT_ora1>
created <2013/02/21 19:03:14.000000>.
```

Example 3. Re-creating the CDCT File After a Failure

The CDCT file and all recent CDCT backup files were damaged or deleted. However, an older backup file based on a previous Logger configuration incarnation is available. To re-create the CDCT file, you first derive a backup based on the previous backup file and then restore that backup.

1. Derive a backup file from existing log files by entering the following command:

   ```
   pwxucdct cmd=derive_cdct_backup prevbackupfile=C:\Informaticav.r.m\pwx\backups
   \prev_backup0.txt backupfile=C:\Informaticav.r.m\pwx\backups\derived_backup1.txt
   cs=C:\Informaticav.r.m\pwx\resources\pwxccl_orcl.cfg
   ```

   Tip: Include the PREVBACKUPFILE parameter to use a previous backup file as a starting point to recover the CDCT. If the utility discovers additional log files that are not in the previous backup, it adds them to the derived backup.

   The following messages are written to the message log file:

   ```
   PWX-33314 TIMEOUTS configuration parameter is deprecated
   PWX-33269 CCL configuration parameter <CHKPT_BASENAME> is deprecated.
   PWX-33269 CCL configuration parameter <CHKPT_NUM> is deprecated.
   PWX-25404 Processing console program. pwxucdct cmd=derive_cdct_backup backupfile=C:\Informaticav.r.m\pwx
   \backups\derived_backup1.txt cs=\Informaticav.r.m\pwx\resources\pwxccl_orcl.cfg
   2 Logger file(s) found for flexible groups masks
   PWX-33261 Loaded "bonus.1". Table "MYORA1.BONUS". Tag "ORAORCLbonus1"
   PWX-33261 Loaded "dept.1". Table "MYORA1.DEPT". Tag "ORAORCLdept1"
   PWX-33261 Loaded "emp.1". Table "MYORA1.EMP". Tag "ORAORCLemp1"
   PWX-33261 Loaded "o015716k.1". Table "MYORA1.ORL157_SRC_16K". Tag "ORAORCLo015716k1"
   PWX-33261 Loaded "o01612k.1". Table "MYORA1.ORL161_SRC_2K". Tag "ORAORCLo01612k1"
   PWX-33261 Loaded "salgrade.1". Table "MYORA1.SALGRADE". Tag "ORAORCLsalgrade1"
   PWX-33263 12 registrations loaded
   PWX-33264 Start of registrations for group processing
   PWX-06264 Group <GROUP1> using registration <emp> reg_schema=<MYORA1> table=<EMP> with schema=<MYORA1>.
   PWX-06264 Group <GROUP2> using registration <bonus> reg_schema=<MYORA1> table=<BONUS> with
   schema=<MYORA1>.
   PWX-06264 Group <GROUP2> using registration <dept> reg_schema=<MYORA1> table=<DEPT> with schema=<MYORA1>.
   PWX-06264 Group <GROUP2> using registration <o015716k> reg_schema=<MYORA1> table=<ORL157_SRC_16K> with
   schema=<MYORA1>.
   PWX-06264 Group <GROUP2> using registration <o01612k> reg_schema=<MYORA1> table=<ORL161_SRC_2K> with
   schema=<MYORA1>.
   PWX-06264 Group <GROUP2> using registration <salgrade> reg_schema=<MYORA1> table=<SALGRADE> with
   schema=<MYORA1>.
   PWX-06119 Controller: added new registration tag ORAORCLbonus1
   PWX-06119 Controller: added new registration tag ORAORCLdept1
   PWX-06119 Controller: added new registration tag ORAORCLo015716k1
   PWX-06119 Controller: added new registration tag ORAORCLsalgrade1
   ```

2. Restore the derived backup by entering the following command:

```
pwxucdct cmd=restore_cdct backupfile=C:\Informaticav.r.m\pwx\backups
\derived_backup1.txt cs=C:\Informaticav.r.m\pwx\resources\pwxccl_orcl.cfg
```

The following messages are written to the message log file:

```
PWX-33314 TIMEOUTS configuration parameter is deprecated
PWX-33269 CCL configuration parameter <CHKPT_BASENAME> is deprecated.
PWX-33269 CCL configuration parameter <CHKPT_NUM> is deprecated.
PWX-25404 Processing console program. pwxucdct cmd=restore_cdct backupfile=C:Informatica
\PowerExchangev.r.m\backups\derived_backup1.txt cs=C:\Informatica\PowerExchangev.r.m\resources
\pwxccl_orcl.cfg
PWX-25200 Created CDCT file "C:\Informatica\PowerExchangev.r.m\resources\CDCT_orcl"
PWX-36937 Restore is using backup of CDCT file <C:\Informatica\PowerExchangev.r.m\resources\CDCT_ora1>
created <2013/02/21 19:03:14.000000>.
```

To verify that the restore operation was successful, check that the return code from the PWXUCDCT utility is zero and that messages PWX-25140 through PWX-25145 provide reasonable record counts for records read from the backup file and for records that were changed in the CDCT file.

Also, view the PWX-25132 messages that report the progress of the restore operation. PowerExchange tries to display progress information to the console approximately every 1 percent of the backup file processed. If you need to display progress information more frequently or less frequently, include the progressfrequency parameter in the restore_cdct statement to adjust the frequency.

3. Run the DELETE_ORPHAN_FILES command to delete log files that are no longer referenced by the restored CDCT file.
After you warm start the PowerExchange Logger, it re-creates the CDCT content for those files.

Example 4. Reporting and Deleting Orphan CDCT Records

You want to determine if orphaned PowerExchange Logger log files exist so that you can delete them. Orphaned log files are not referenced by any record in the CDCT file.

1. To determine if orphan log files exist, enter the following command:

```
pwxucdct cmd=report_orphan_files cs=C:\Informatica\PowerExchangev.r.m\resources
\pwxccl_orcl.cfg
```

The following messages are displayed on screen and written to the message log file:

```
PWX-25404 Processing console program. pwxucdct cmd=report_orphan_files
REPORT FOR COMMAND REPORT_ORPHAN_FILES
PWX-25229 Started initialization of the CDCT Retention Array
PWX-25230 Retention array initialized. Files 2. CDCTs read 0. Allocated 0. Memory 0
5 Logger file(s) found for mask C:\Informatica\PowerExchange9.0.0\capture\condenseO.CND.*
Total files found for masks 5
Date Time Seq File
---- ---- --- ----
091109 1447 007 C:\Informatica\PowerExchangev.r.m\capture\condenseO.CND.CP091109.T1447007
091109 1615 008 C:\Informatica\PowerExchangev.r.m\capture\condenseO.CND.CP091109.T1615008
091109 1615 009 C:\Informatica\PowerExchangev.r.m\capture\condenseO.CND.CP091109.T1615009
```

These messages indicate that three orphan log files exist.

2. To delete all orphan log files, enter the following command:

```
pwxucdct cmd=delete_orphan_files cs=C:\Informatica\PowerExchangev.r.m\resources
\pwxccl_orcl.cfg
```

The following messages are displayed on screen and written to the message log file:

```
PWX-25404 Processing console program. pwxucdct cmd=delete_orphan_files
REPORT FOR COMMAND DELETE_ORPHAN_FILES
PWX-25229 Started initialization of the CDCT Retention Array
PWX-25230 Retention array initialized. Files 2. CDCTs read 0. Allocated 0. Memory 0
5 Logger file(s) found for mask C:\Informatica\PowerExchange9.0.0\capture\condenseO.CND.*
Total files found for masks 5
Date Time Seq File
---- ---- --- ----
PWX-25163 Deleted orphan file C:\Informatica\PowerExchangev.r.m\capture\condenseO.CND.CP091109.T1447007
PWX-25163 Deleted orphan file C:\Informatica\PowerExchangev.r.m\capture\condenseO.CND.CP091109.T1615008
PWX-25163 Deleted orphan file C:\Informatica\PowerExchangev.r.m\capture\condenseO.CND.CP091109.T1615009
PWX-25162 Files not referenced in CDCT (orphans) 3
```

These messages indicate that the orphan log files were successfully deleted.

3. To verify that the orphan log files were deleted, enter the following command:

```
pwxucdct cmd=report_files_by_time
```

This command lists all current log files in the order in which they were created, for example:

```
PWX-25404 Processing console program. pwxucdct cmd=report_files_by_time
REPORT FOR COMMAND REPORT_FILES_BY_TIME
PWX-25229 Started initialization of the CDCT Retention Array
PWX-25230 Retention array initialized. Files 2. CDCTs read 0. Allocated 0. Memory 0
2 Logger file(s) found for mask C:\Informatica\PowerExchangev.r.m\capture\condenseO.CND.*
Date Time Seq File
---- ---- --- ----
091109 1443 006 C:\Informatica\PowerExchangev.r.m\capture\condenseO.CND.CP091109.T1443006
091109 1615 010 C:\Informatica\PowerExchangev.r.m\capture\condenseO.CND.CP091109.T1615010
```

The list does not include the deleted orphan log files.

Example 5. Reporting and Deleting Expired CDCT Records

To find out if the CDCT file contains records for expired PowerExchange Logger log files, run the REPORT_EXPIRED_CDCT command. If the report indicates that records for expired logs exist, run the DELETE_EXPIRED_CDCT command to delete them and free space on your system.

Note: To use the DELETE_EXPIRED_CDCT command, you must set the LOGGER_DELETES_EXPIRED_CDCT_RECORDS parameter to Y in the pwxccl configuration file. If this parameter is set to Y or is not specified, the PowerExchange Logger does not delete the expired CDCT records until a file switch occurs.

1. To determine if CDCT records for expired log files exist, enter the following command:

```
pwxucdct cmd=report_expired_cdct cs=C:\Informatica\PowerExchangev.r.m\resources
\pwxccl_orcl.cfg
```

If you do not direct the output to a report file, this command prints the following messages to the message log file:

```
PWX-33314 TIMEOUTS configuration parameter is deprecated
PWX-33269 CCL configuration parameter <CHKPT_BASENAME> is deprecated.
PWX-33269 CCL configuration parameter <CHKPT_NUM> is deprecated.
PWX-25404 Processing console program. pwxucdct cmd=report_expired_files cs=C:\Informatica\PowerExchangev.r.m\resources
\pwxccl_orcl.cfg
REPORT FOR COMMAND Report Expired Files
------------------------------------------------------------------------------
------------------------------      CCL FILES     ------------------------------
------------------------------------------------------------------------------
Incarnation: 20130221202420000000 Group #: 1 File Seq#: 1302212029001
             Name:                C:\Informatica\PowerExchangev.r.m\resources\ccl\group1\grp1.CND.CP130221.T2029001
             Open TimeStamp:      20130221202930000000
             Close TimeStamp:     20130221202948000000
             Record Count:        840
             Commit Count:        60
             Has Uncommited data: No
             Status:              Closed
             Fmt Version:         951
             Interest List flags: 1

Restart Information:   Sequence                                                                  Restart
Timestamp
Begin:                 D4000000CF3E790000000000000000000CF3E7700000000000002F500006E73001000020000 000000CF3E774ED056E4
20130221202929000000
End:                   D4000000CF3F4A000000000000FFFFFFFFFFFFFFFFFFFF000002F5000071850084FFFF0000 000000CF3F494ED056E4
20130221202936000000

Incarnation: 20130221202420000000 Group #: 1 File Seq#: 1302212029002
             Name:                C:\Informatica\PowerExchangev.r.m\resources\ccl\group1\grp1.CND.CP130221.T2029002
             Open TimeStamp:      20130221202952000000
             Close TimeStamp:     20130221202958000000
             Record Count:        280
             Commit Count:        20
             Has Uncommited data: No
             Status:              Closed
             Fmt Version:         951
             Interest List flags: 1

Restart Information:   Sequence                                                                  Restart
Timestamp
Begin:                 D4000000CF3F680000000000000000000CF3F6700000000000002F5000071BB001000020000 000000CF3F674ED056E4
```

```
20130221202950000000
End:                    D4000000CF3FA8000000000000FFFFFFFFFFFFFFFFFFFF000002F5000072B50090FFFF0000 000000CF3FA74ED056E4
20130221202950000000

Incarnation: 20130221202420000000 Group #: 1 File Seq#: 1302212030001
             Name:                C:\Informatica\PowerExchangev.r.m\resources\ccl\group1\grp1.CND.CP130221.T2030001
             Open TimeStamp:      20130221203005000000
             Close TimeStamp:     20130221203010000000
             Record Count:        280
             Commit Count:        20
             Has Uncommited data: No
             Status:              Closed
             Fmt Version:         951
             Interest List flags: 1

Restart Information:   Sequence                                                                  Restart
Timestamp
Begin:                  D4000000CF3FBB000000000000000000CF3FB900000000000002F5000072D0014400020000 000000CF3FB94ED056E4
20130221203004000000
End:                    D4000000CF3FFB000000000000FFFFFFFFFFFFFFFFFFFF000002F5000073CB017CFFFF0000 000000CF3FFA4ED056E4
20130221203004000000
```

This output indicates that the CDCT file contains records for three expired log files.

2. To delete the expired PowerExchange Logger log files and the associated CDCT records, enter the following command:

   ```
   pwxucdct cmd=delete_expired_files cs=C:\Informatica\PowerExchangev.r.m\resources
   \pwxccl_orcl.cfg
   ```

   The following messages, which are written to the message log file, confirm that the three log files were deleted:

   ```
   PWX-33314 TIMEOUTS configuration parameter is deprecated
   PWX-33269 CCL configuration parameter <CHKPT_BASENAME> is deprecated.
   PWX-33269 CCL configuration parameter <CHKPT_NUM> is deprecated.
   PWX-25404 Processing console program. pwxucdct cmd=DELETE_EXPIRED_FILES cs=C:\Informatica
   \PowerExchangev.r.m\resources\pwxccl_orcl.cfg
   REPORT FOR COMMAND DELETE_EXPIRED_CDCT
   PWX-25204 Deleted expired file "C:\Informatica\PowerExchangev.r.m\resources\ccl
   \group1\grp1.CND.CP130221.T2029001"
   PWX-25204 Deleted expired file "C:\Informatica\PowerExchangev.r.m\resources\ccl
   \group1\grp1.CND.CP130221.T2029002"
   PWX-25204 Deleted expired file "C:\Informatica\PowerExchangev.r.m\resources\ccl
   \group1\grp1.CND.CP130221.T2030001"
   ```

3. To verify that the expired log files and associated CDCT records were deleted, enter the following command:

   ```
   pwxucdct cmd=report_cdct cs=C:\Informatica\PowerExchangev.r.m\resources
   \pwxccl_orcl.cfg
   ```

   Ensure that the CCL Files section of the report does not list the deleted log files.

Example 6. Printing the CDCT File Contents

Informatica Global Customer Support requests that you print the CDCT file contents and send it to them in a text file for debugging use.

1. From the command line, navigate to the PowerExchange installation directory and enter the following command:

   ```
   pwxucdct cmd=report_cdct >C:\Informatica\PowerExchangev.r.m\reports\cdct_debug.txt
   cs=C:\Informatica\PowerExchangev.r.m\resources\pwxccl_orcl.cfg
   ```

   To print the information to a file, include a greater than (>) sign followed by the output path and file name. Otherwise, the information is displayed on screen.

2. Review the report contents.
   The following example report shows the CDCT contents for a PowerExchange Logger with only one group:

```
DTL-33314 TIMEOUTS configuration parameter is deprecated
DTL-25404 Processing console program. PWXUCDCT cs=C:\Informatica\PowerExchange\v.r.m\resources\capt\pwxccl_ORCL.cfg
CMD=REPORT_CDCT
------------------------------------------------------------------------------
------------------------------   CCL INCARNATION  ------------------------------
```

```
--------------------------------------------------------------------------------
Incarnation: 20141217080907000000
             Instance Name:       ORCL
             Image Type:          Before and After Images
             Group Count:         1
             Status:              Current
             Create Rsn:          Cold Start
             File Encryption:     AES256

Restart Information:   Sequence                                                                        Restart
Timestamp
Begin                  D40000007B0198000000000000000000000000000000000000000000000000000000000000000000 0000007B0198D35844C4
20141217080913000000
End                    D40000007CFC67000000000000FFFFFFFFFFFFFFFFFFFF0000016D000046300108FFFF0000 0000007CFC66D35844C4
20141217102535000000

--------------------------------------------------------------------------------
------------------------------      CCL GROUP     ------------------------------
--------------------------------------------------------------------------------
Incarnation: 20141217080907000000 Group #: 0
             Name:                DEFAULT
             File Prefix:         C:\Informatica\PowerExchangev.r.m\resources\ccl\group1\grp2
             Reg Count:           6
             File Count:          4
             Status:              Active
             First File Seq:      1412170809001
             Current File Seq:    1412171025001
             Oldest File TStamp:  20141217080942000000

--------------------------------------------------------------------------------
--------------------------   CCL REGISTRATIONS   -------------------------------
--------------------------------------------------------------------------------
Incarnation: 20141217080907000000 Group #: 0 Reg Seq#: 5
             Name:                ORAORCLbonus1
             Status:              Active
             Activation Date:     20141217102409000000
             Inactivation Date:
             Default Schema:      ORAAUS1
Incarnation: 20141217080907000000 Group #: 0 Reg Seq#: 4
             Name:                ORAORCLdatetime1
             Status:              Active
             Activation Date:     20141217080907000000
             Inactivation Date:
             Default Schema:      ORAAUS1
Incarnation: 20141217080907000000 Group #: 0 Reg Seq#: 3
             Name:                ORAORCLdept1
             Status:              Active
             Activation Date:     20141217080907000000
             Inactivation Date:
             Default Schema:      ORAAUS1
Incarnation: 20141217080907000000 Group #: 0 Reg Seq#: 2
             Name:                ORAORCLemp1
             Status:              Active
             Activation Date:     20141217080907000000
             Inactivation Date:
             Default Schema:      ORAAUS1
Incarnation: 20141217080907000000 Group #: 0 Reg Seq#: 1
             Name:                ORAORCLempencr1
             Status:              Active
             Activation Date:     20141217080907000000
             Inactivation Date:
             Default Schema:      ORAAUS1
Incarnation: 20141217080907000000 Group #: 0 Reg Seq#: 0
             Name:                ORAORCLtable11
             Status:              Active
             Activation Date:     20141217080907000000
             Inactivation Date:
             Default Schema:      ORAAUS1

--------------------------------------------------------------------------------
------------------------------      CCL FILES     ------------------------------
--------------------------------------------------------------------------------
Incarnation: 20141217080907000000 Group #: 0 File Seq#: 1412170809001
             Name:                C:\Informatica\PowerExchangev.r.m\\resources\ccl\group1\grp0.CND.CP141217.T0809001
resources\ccl\group1\
             Open TimeStamp:      20141217080917000000
             Close TimeStamp:     20141217080942000000
             Record Count:        180
             Commit Count:        20
             Has Uncommitted data No
             Is encrypted:        Yes
             Status:              Closed
             Fmt Version:         961
             Encryption Key Test: 277790D3E372B2623D2CFD98FF2ACD62FC92A267D2095013C7FC2D3C44339F6E

Restart Information:   Sequence                                                                        Restart
Timestamp
Begin:                 D40000007BC97C0000000000000000007BC97B000000000000001680000B64E001000020000 0000007BC97BD35844C4
```

```
20141215150331000000
End:                    D40000007CE166000000000000FFFFFFFFFFFFFFFFFFFF0000016D000011D30084FFFF0000 0000007CE165D35844C4
20141217080931000000

Incarnation: 20141217080907000000 Group #: 0 File Seq#: 1412170809002
             Name:                C:\Informatica\PowerExchangev.r.m\\resources\ccl\group1\grp0.CND.CP141217.T0809002
             Open TimeStamp:      20141217080950000000
             Close TimeStamp:     20141217081055000000
             Record Count:        36
             Commit Count:        4
             Has Uncommitted data No
             Is encrypted:        Yes
             Status:              Closed
             Fmt Version:         961
             Encryption Key Test: 277790D3E372B2623D2CFD98FF2ACD62FC92A267D2095013C7FC2D3C44339F6E

Restart Information:   Sequence                                                                  Restart
Timestamp
Begin:                 D40000007CE171000000000000000000007CE170000000000000016D000011DC001000020000 0000007CE170D35844C4
20141217080948000000
End:                   D40000007CE189000000000000FFFFFFFFFFFFFFFFFFFF0000016D000011F7008CFFFF0000 0000007CE188D35844C4
20141217081032000000

Incarnation: 20141217080907000000 Group #: 0 File Seq#: 1412170811001
             Name:                C:\Informatica\PowerExchangev.r.m\resources\ccl\group1\grp0.CND.CP141217.T0811001
             Open TimeStamp:      20141217081102000000
             Close TimeStamp:     20141217081117000000
             Record Count:        36
             Commit Count:        4
             Has Uncommitted data No
             Is encrypted:        Yes
             Status:              Closed
             Fmt Version:         961
             Encryption Key Test: A2649CDC44DB6CF714E0739CB1F9FCD035CC834E90AB3E2059F7BBEC41D62742

Restart Information:   Sequence                                                                  Restart
Timestamp
Begin:                 D40000007CE198000000000000000000007CE197000000000000016D000011FA001000020000 0000007CE197D35844C4
20141217081101000000
End:                   D40000007CE19D000000000000FFFFFFFFFFFFFFFFFFFF0000016D0000120A01E8FFFF0000 0000007CE19CD35844C4
20141217081101000000

Incarnation: 20141217080907000000 Group #: 0 File Seq#: 1412171025001
             Name:                C:\Informatica\PowerExchangev.r.m\resources\ccl\group1\grp0.CND.CP141217.T1025001
             Open TimeStamp:      20141217102522000000
             Close TimeStamp:     20141217102605000000
             Record Count:        1
             Commit Count:        1
             Has Uncommitted data No
             Is encrypted:        Yes
             Status:              Closed
             Fmt Version:         961
             Encryption Key Test: CB9202C5D9D0E01CAA57C0128D6244C2964AAEB866613EA0C7E3EE5151C3C255

Restart Information:   Sequence                                                                  Restart
Timestamp
Begin:                 D40000007CFC610000000000000000007CFC600000000070000016D0000462400100020000 0000007CFC60D35844C4
20141217102520000000
End:                   D40000007CFC670000000000000FFFFFFFFFFFFFFFFFFFF0000016D000046300108FFFF0000 0000007CFC66D35844C4
20141217102535000000
```

The report contains separate sections for the PowerExchange Logger (CCL) configuration incarnation, the groups that are defined in the group definition file that is specified in the pwxccl.cfg configuration file, the capture registrations for the tables, and the associated Logger log files. In the CCL Files section, the information that is reported for each log file includes whether the log file is encrypted and the restart tokens.

CHAPTER 19

PWXUCREG - Capture Registration Suspend Utility

This chapter includes the following topics:

PWXUCREG Utility Overview

Use the PWXUCREG utility to temporarily suspend change capture processing for registered sources. Later, you can use the utility to reactivate the suspended registrations to resume change capture.

This utility has many potential uses. For example, use it to suspend change capture while you make DDL changes such as cascade delete or delete all row changes or corrections to a source or target object.

This utility runs on z/OS only. It processes registrations only for data sources from which the following ECCRs capture data:

- Adabas ECCR
- Datacom table-based ECCR
- IDMS log-based ECCR
- IMS log-based ECCR

You can use the utility to perform the following tasks:

- Suspend capture registrations to temporarily stop change capture activity for registered sources during the suspension window.
- Reactivate suspended capture registrations after a suspension to resume change data capture.

- Display the status setting for capture registrations to verify a status change.
- Skip all change records in the change stream that have timestamps earlier than the current system time when starting change capture for a registration that has been activated for the first time from the PowerExchange Navigator.

When you use the utility to change the registration status, it sets suspension and activation timestamps. The period bounded by the suspension and activation timestamps is called the *suspension window*.

You must refresh the ECCR after a registration status change to have it detect the status change and get the suspension and activation timestamps. During the suspension window, the ECCR discards change records that have a timestamp later than the suspension timestamp. The utility writes message output for registration status changes for audit and monitoring purposes.

The PowerExchange Navigator displays the current registration status in the **Status** field of the Resource Inspector. After the utility processes a PWXUCREG registration suspension request, the Resource Inspector displays the suspension timestamp in current system time in the **Suspend Time** field and displays the value **Suspended** in the **Status** field. After the utility processes a PWXUCREG registration reactivation request, the Resource Inspector displays the activation timestamp in the **Active Time** field and resets the **Status** to **Active**.

PWXUCREG Usage Considerations

Before you begin using the PWXUCREG utility to change the status of registrations, review this list of usage considerations.

- Only one suspension window can be open for a capture registration at a time.
- If you issue a SUSPEND_REGISTRATION command followed by an ACTIVATE_REGISTRATION command and then issue another SUSPEND_REGISTRATION command before the ECCR starts processing the change records later than the first suspension window, unpredictable results might occur. Wait until the ECCR has processed all of the change records with timestamps earlier than the activation timestamp before issuing another suspension request.
- To generate suspension and activation timestamps, the utility uses the current system time at the time when the SUSPEND_REGISTRATION or ACTIVATE_REGISTRATION command is processed, without any adjustment for the local time. These timestamps are included in messages and in the **Suspend Time** and **Active Time** fields of the PowerExchange Navigator Resource Inspector. These timestamps define the start and end of the *suspension window*. The utility uses current system time for the timestamps because the supported database types store timestamps in CDC records in current system time.
- You must issue an ECCR REFRESH command after issuing any PWXUCREG command that changes the registration status. This refresh process enables the ECCR to read the registration information from the CCT data set again to get the suspension and activation timestamps and the new registration status.
- The first time you activate a capture registration in the PowerExchange Navigator, the activation timestamp is not set. The **Active Time** field is blank until you use the PWXUCREG utility to submit a SUSPEND_REGISTRATION command followed by an ACTIVATE_REGISTRATION command.
- If the ECCR ends abnormally and then warm starts within the same suspension window, the utility issues a message when it encounters the first change record in the suspension window to be discarded.
- If you have multiple registrations with the same registration tag name, you must suspend or reactivate each one. The utility cannot process all registrations with the same tag name in a single SUSPEND_REGISTRATION or ACTIVATE_REGISTRATION command.

- When discarding change records for a suspended registrations, the ECCR verifies that the associated UOW started within the suspension window. If the UOW started before the beginning of the suspension window, the ECCR either issues a warning and continues or issues an error message and ends, depending on the ON_SUSPENSION_ERROR_CONTINUE parameter setting in the ECCR configuration file.
- When capturing change records for an activated registration, the ECCR verifies that the associated UOW started after the suspension window closed. If the UOW started before the end of the suspension window, the ECCR either issues a warning and continues or issues an error message and ends, depending on the ON_SUSPENSION_ERROR_CONTINUE parameter setting in the ECCR configuration file.
- To have the ECCR discard change records that have timestamps earlier than the current system time, use the DROP_OLD_REGISTRATION_DATA command. You can issue this command for an Active registration only. This command sets a special suspension window that extends from the earliest point in the logs to the current system time.
- You can cancel a suspension or activation operation before you refresh the ECCR for the registration status change. You might want to cancel a suspension or reactivation request because it was issued at the wrong time related to the database processing or the command input contained errors. The cancel command resets the suspension or activation timestamp.
- In the PowerExchange Navigator, you can change the registration **Status** value of **Suspended** only to **History**. Make this change only if you no longer want to use the registration for change capture. You cannot change a registration **Status** value of **Active** to **Suspended** from the PowerExchange Navigator. You must use the PWXUCREG utility to make this status change.

Supported Operating Systems for the PWXUCREG Utility

The PWXUCREG utility runs on z/OS systems only.

Suspending Change Capture for Registered Sources Temporarily

Use this task flow to suspend change capture processing for registered sources temporarily.

You perform some tasks with the PWXUCREG utility and other tasks outside of the utility on the z/OS system.

Before you begin, ensure that the REFRESH_ALLOWED=Y parameter is specified in the ECCR configuration file. Also, you must have the authority to issue a REFRESH command after each registration status change.

1. Stop database activity for the registered source or sources for which you want to suspend capture registrations.
2. To suspend the capture registrations, use the PWXUCREG utility to issue the SUSPEND_REGISTRATION command.

   The suspension window opens. The utility sets the suspension timestamp to the current system time without any adjustment for the local time. Also, the utility issues message PWX-03716 to the DTLLOG log to report the registration status change.

For each suspended registration, the PowerExchange Navigator Resource Inspector displays **Suspended** in the **Status** field and the suspension timestamp in the **Suspend Time** field. The **Suspend Time** value is not adjusted for the local time.

3. For Adabas sources only, perform a PLOG switch.

   This step ensures that all of the changes up to the point of the PLOG switch are captured for the active registration.

4. Enter the ECCR REFRESH command with the MVS MODIFY (F) command:

   ```
   F eccr_task_name,REFRESH
   ```

   The ECCR becomes aware of the registration status change and suspension timestamp. When the ECCR encounters the first change record to discard, it issues message PWX-07752. The ECCR discards change records that have a timestamp later than the suspension timestamp.

5. Run the jobs or processes that generate the changes that you do not want to capture for the source or sources that are associated with the suspended registrations.

6. To reactivate the capture registrations, use the PWXUCREG utility to issue the ACTIVATE_REGISTRATION command.

   The suspension window closes. The utility sets the activation timestamp to the current system time without any adjustment for the local time. Also, the utility issues message PWX-03716 to the DTLLOG log to report the registration status change.

   For each reactivated registration, the PowerExchange Navigator Resource Inspector displays **Active** in the **Status** field and the activation timestamp in the **Active Time** field. The **Active Time** value is not adjusted for the local time.

7. For Adabas sources only, perform a PLOG switch.

   This step ensures that all of the changes that occur during the suspension window up to the PLOG switch are discarded for the suspended registration.

8. Enter the ECCR REFRESH command with the MVS MODIFY (F) command again.

   The ECCR becomes aware of the registration status change and activation timestamp.

9. Enable database activity to resume on the registered source or sources.

   The ECCR starts capturing change records that have timestamps later than the activation timestamp. The ECCR issues message PWX-07753 when it encounters the first change record in the change stream after the end of the suspension window.

Note: You can automate this processing if appropriate for your environment.

General Syntax for PWXUCREG Commands

Use the following general syntax to enter global commands and registration-specific commands in the PWXSYSIN input for a PWXUCREG job:

```
//PWXSYSIN  DD  *
  SET_CONTROL_VALUE,global_parameter1=value1;
    global parameter2=value,
    global parameter3
  ;
  Registration-specific command primary keyword,
    parameter1=value,
    parameter2=value,
    paraemter3=value
  ;
```

```
 <additional registration-specific commands>
 ;
SET_CONTROL_VALUE,global_parameter1=value2;
 Registration-specific command primary keyword,
   parameter1=value,
   parameter4=value
 ;
<additional registration commands>
 ;
/*
```

The following syntax rules apply:

- A command consists of a primary keyword followed by one or more valid parameters.
- A command ends with a semicolon (;).
- Use a comma (,) to separate a primary keyword from a parameter. Also, use a comma between parameters. Do not use a comma after the last parameter in a command, before the ending semicolon.
- In a SET_CONTROL_VALUE command, you can specify one or more global parameters. Global parameters are optional, but you must include at least one global parameter in the command. Alternatively, you can specify a separate SET_CONTROL_VALUE statement for each parameter.
- In a registration-specific command, you can enter multiple parameters. Some parameters are required to identify the registrations to process, and other parameters are optional.
- The utility parses and executes commands in the input stream one at a time from top to bottom.
- You can repeat the SET_CONTROL_VALUE command in the input stream with different parameters or with the same parameters but different values for a subsequent set of registration-specific commands.
- You can include parameters in a registration-specific command that overwrite corresponding global parameters in the preceding SET_CONTROL_VALUE statement. For example, include the REGISTRATION_LOCATION parameter in a registration-specific command to overwrite the GLOBAL_REGISTRATION_LOCATION parameter in the previous SET_CONTROL_VALUE command.

Summary of PWXUCREG Commands

Use the summary of PWXUCREG utility commands to determine which command keyword and parameters you want to use.

The SET_CONTROL_VALUE keyword sets global parameter values such as the user ID and password for subsequent commands in the PWXSYSIN input stream for the PWXUCREG job. The registration-specific commands apply to the registration or registrations that match the selection criteria that you enter in parameters such as those for the database instance and registration name.

The following table describes the SET_CONTROL_VALUE global command and each registration-specific command, including the primary keyword and associated parameters:

Command Keyword	Description	Parameters[1]
SET_CONTROL_VALUE	Sets global parameter values that apply to subsequent registration-specific commands in the PWXSYSIN input of the PWXUCREG JCL. Use this command if you run PWXUCREG jobs that contain multiple commands and want to define common parameter values once for a set of commands. You can override any global value in a subsequent registration-specific command or by specifying another SET_CONTROL_VALUE command later in the input stream. You must specify at least one parameter for this command.	- [DISPLAY_REGISTRATION_AFTER_COMMAND] - [DISPLAY_REGISTRATION_BEFORE_COMMAND] - [GENERIC] - [GLOBAL_REGISTRATION_LOCATION] - [GLOBAL_USER] - [GLOBAL_EPWD] - [GLOBAL_PWD] - [SYSTEM_CONSOLE_MESSAGES_COMMAND] - [SYSTEM_CONSOLE_MESSAGES_DISPLAY] - [SHOW_EXPANDED_STATEMENT]
ACTIVATE_REGISTRATION	Reactivates a capture registration that has a status of Suspended so that the ECCR resumes change capture for the registered source. Also sets an activation timestamp in current system time, not adjusted for local time, to indicate the end of the suspension window. You can issue this command only for registrations that were previously suspended with the SUSPEND_REGISTRATION command. This command can apply to multiple registrations if you specify GENERIC=Y and enter the asterisk (*) wildcard, or a string followed by the wildcard, in the REGISTRATION_NAME or DATABASE_INSTANCE parameter.	- REGISTRATION_NAME - DATABASE_INSTANCE - [DATABASE_TYPE] - Recommended - [EPWD] - [GENERIC] - [PWD] - [REGISTRATION_LOCATION] - [USER] - [VALIDATE]

Command Keyword	Description	Parameters[1]
CANCEL_ACTIVATE_REGISTRATION	Cancels a previous ACTIVATE_REGISTRATION request before you refresh the ECCR for the activation. Also sets the registration status back to Suspended. This command can apply to multiple registrations if you specify GENERIC=Y and enter the asterisk (*) wildcard, or a string followed by the wildcard, in the REGISTRATION_NAME or DATABASE_INSTANCE parameter. **Tip:** To cancel the reactivation action for all registrations that were specified in the previous ACTIVATE_REGISTRATION command, specify the same parameter values as in the ACTIVATE_REGISTRATION command.	- REGISTRATION_NAME - DATABASE_INSTANCE - [DATABASE_TYPE] - Recommended - [EPWD] - [GENERIC] - [PWD] - [REGISTRATION_LOCATION] - [USER] - [VALIDATE]
CANCEL_SUSPEND_REGISTRATION	Cancels a previous SUSPEND_REGISTRATION command before you refresh the ECCR. Also resets the registration status to Active and resets the start and end times for the suspension window. This command can apply to multiple registrations if you specify GENERIC=Y and enter the asterisk (*) wildcard, or a string followed by the wildcard, in the REGISTRATION_NAME or DATABASE_INSTANCE parameter. **Tip:** To cancel the suspension action for all registrations that were specified in the previous SUSPEND_REGISTRATION command, specify the same parameter values as in the SUSPEND_REGISTRATION command.	- REGISTRATION_NAME - DATABASE_INSTANCE - [DATABASE_TYPE] - Recommended - [EPWD] - [GENERIC] - [PWD] - [REGISTRATION_LOCATION] - [USER] - [VALIDATE]

Command Keyword	Description	Parameters[1]
DISPLAY_REGISTRATION	Displays registration status information before or after another command that changes registration status so that you can verify the status change. This information includes the current registration status setting and the activation and suspension timestamps. The timestamps are in current system time and not adjusted for local time. This command does not support the GENERIC parameter or the global GENERIC parameter of the SET_CONTROL_VALUE command. This command can display status information for multiple registrations without the GENERIC parameter. **Tip:** Instead of specifying this command in the JCL input stream multiple times, you can specify the global SET_CONTROL_VALUE keyword with the DISPLAY_REGISTRATION_BEFORE_COMMAND and DISPLAY_REGISTRATION_AFTER_COMMAND parameters. These global parameters automatically display the registation status information before and after each command that changes the registration status.	- REGISTRATION_NAME - DATABASE_INSTANCE - [DATABASE_TYPE] - Recommended - [EPWD] - [PWD] - [REGISTRATION_LOCATION] - [USER] - [VALIDATE]

Command Keyword	Description	Parameters[1]
DROP_OLD_REGISTRATION_DATA	Sets a special suspension window that enables you to begin change capture for a registration from the current system time. The suspension window extends from the earliest available point in the change stream to the current system time. The ECCR discards change records that have timestamps within the suspension window. Often, this command is used for new registrations that were just activated for the first time from the PowerExchange Navigator. This command can apply to multiple registrations if you specify GENERIC=Y and enter the asterisk (*) wildcard, or a string followed by the wildcard, in the REGISTRATION_NAME or DATABASE_INSTANCE parameter.	- REGISTRATION_NAME - DATABASE_INSTANCE - [DATABASE_TYPE] - Recommended - [EPWD] - [GENERIC] - [PWD] - [REGISTRATION_LOCATION] - [USER] - [VALIDATE]
RESET_SUSPENSION_WINDOW	Clears any activation and suspension timestamps that define the current suspension window and resets the registration status to Active. This command can apply to multiple registrations if you specify GENERIC=Y and enter the asterisk (*) wildcard, or a string followed by the wildcard, in the REGISTRATION_NAME or DATABASE_INSTANCE parameter.	- REGISTRATION_NAME - DATABASE_INSTANCE - [DATABASE_TYPE] - Recommended - [EPWD] - [GENERIC] - [PWD] - [REGISTRATION_LOCATION] - [USER] - [VALIDATE]

Command Keyword	Description	Parameters[1]
SUSPEND_REGISTRATION	Suspends a capture registration that has a status of Active so that the ECCR stops capturing changes for the registered source. Also sets a suspension timestamp in current system time, not adjusted for local time, to indicate the start of the suspension window. This command can apply to multiple registrations if you specify GENERIC=Y and enter the asterisk (*) wildcard, or a string followed by the wildcard, in the REGISTRATION_NAME or DATABASE_INSTANCE parameter.	- REGISTRATION_NAME - DATABASE_INSTANCE - [DATABASE_TYPE] - Recommended - [EPWD] - [GENERIC] - [PWD] - [REGISTRATION_LOCATION] - [USER] - [VALIDATE]
1. Parameters enclosed in square brackets [ ] are optional.		

Global SET_CONTROL_VALUE Parameters

You can specify one or more global parameters in a SET_CONTROL_VALUE command that you include in the PWXSYSIN input for a PWXUCREG job.

The global parameters apply to the subsequent registration-specific commands in the PWXSYSIN input, unless you override the global parameter value in one of the following ways:

- You specify a subsequent registration-specific command with the corresponding parameter, if available.
- You specify another SET_CONTROL_VALUE command with a different global parameter value later in the input stream.

All of the global parameters are optional but you must specify at least one in a SET_CONTROL_VALUE command.

In the following parameter descriptions, curly brackets indicate that one of the options must be entered, and underlining indicates the default value.

Parameter Descriptions

DISPLAY_REGISTRATION_AFTER_COMMAND={N|Y}

Displays information about the capture registration or registrations that were processed by the preceding registration-specific command that changed the registration status. This information includes the registration current status setting and activation and suspension timestamps. You can use this information to verify that the preceding command correctly changed the status. Options are:

- **N**. Do not display registration information after each command that changes the registration status.
- **Y**. Display registration information after each command that changes the registration status.

Default is Y. If you accept the default value, you do not need to specify the DISPLAY_REGISTRATION command in the PWXUCREG JCL to display registration status information after a command, unless you want to override this global parameter setting.

You can use this parameter with the DISPLAY_REGISTRATION_BEFORE_COMMAND parameter to display registration status information before and after a command that changes the status.

DISPLAY_REGISTRATION_BEFORE_COMMAND={N|Y}

Displays information about the capture registration or registrations that will be processed by a subsequent registration-specific command that changes the registration status. This information includes the registration current status setting and activation and suspension timestamps. You can use this information to verify that the command correctly changes the status for the original value to the target value. Options are:

- **N**. Do not display registration information before each command that changes the registration status.
- **Y**. Display registration information before each command that changes the registration status.

Default is N. If you accept the default value, you can still specify the DISPLAY_REGISTRATION command in the PWXUCREG JCL to display registration status information before a particular command.

You can use this parameter with the DISPLAY_REGISTRATION_AFTER_COMMAND parameter to display registration status information before and after a command that changes the status.

GENERIC={N|Y}

Enables you to issue registration-specific commands that change the registration status for multiple registrations. You must also specify masks that contain the asterisk (*) wildcard in certain parameters of the registration-specific commands, such as REGISTRATION_NAME and DATABASE_INSTANCE.

This parameter is not required or supported for the DISPLAY_REGISTRATION command, which is generic by default.

You can override this global setting for a particular registration-specific command by including the GENERIC parameter in the command.

GLOBAL_EPWD=*encrypted_password*

Specifies an encrypted password to use with the user ID that is specified in the associated GLOBAL_USER parameter.

Consult with your system security administrator to determine the requirements for a user ID and a password or encrypted password based on your system security and the SECURITY statement in the DBMOVER configuration member.

Tip: You can use the PowerExchange Navigator to create an encrypted password. Click **File** > **Encrypt**.

If you specify this parameter, do not also specify the GLOBAL_PWD parameter. If you specify both, the GLOBAL_EPWD parameter takes precedence. Use the GLOBAL_EPWD parameter if you are not allowed to store passwords in readable format.

You can override this value for a particular registration-specific command by including the EPWD parameter in the command.

GLOBAL_PWD=*password*

Specifies a nonencrypted password to use with the user ID that is specified in the associated GLOBAL_USER parameter. This value is case-sensitive.

Consult with your system security administrator to determine the requirements for a user ID and a password or encrypted password based on your system security and the SECURITY statement in the DBMOVER configuration member.

If you specify this parameter, do not also specify the GLOBAL_EPWD parameter. If you specify both, the GLOBAL_EPWD parameter takes precedence.

You can override this value for a particular registration-specific command by including the PWD parameter in the command.

GLOBAL_REGISTRATION_LOCATION={*hlq.data_set_name*|local}

Specifies the location of the VSAM CCT data set that contains the capture registrations. Default value is "local."

Usually, the default value of "local" is acceptable because the capture registrations must reside on the z/OS source system where the PWXUCREG utility runs.

You can override this value for a particular registration-specific command by including the REGISTRATION_LOCATION parameter in the command.

GLOBAL_USER=*user_id*

Specifies a user ID that has the authority to access capture registrations in the CCT data set on the source system. This value is case-sensitive.

Consult with your system security administrator to determine the requirements for a user ID and a password or encrypted password based on your system security and the SECURITY statement in the DBMOVER configuration member.

If you specify this parameter, also specify either the GLOBAL_EPWD or GLOBAL_PWD parameter.

You can override the global user ID for a particular registration-specific command by including the USER parameter in the command.

SHOW_EXPANDED_STATEMENT={N|Y}

Displays the input command statements in an expanded format that includes the parameters that you specified in the PWXUCREG job and the other valid parameters that you did not specify but are in effect with their default values or an asterisk (*) wildcard entry.

SYSTEM_CONSOLE_MESSAGES_COMMAND={N|Y}

Controls whether the utility routes message output from registration-specific commands that change the registration status to the z/OS system console as well as to the DTLLOG log. If you enter Y, messages are sent to the system console and the DTLLOG log. If you accept the default value of N, the messages are sent only to the DTLLOG log.

SYSTEM_CONSOLE_MESSAGES_DISPLAY={N|Y}

Controls whether the utility routes message output from an explicit or automatic display registration request to the z/OS system console as well as to the DTLLOG log. These requests result from a DISPLAY_REGISTRATION command or a SET_CONTROL_VALUE command that includes the global DISPLAY_REGISTRATION_AFTER_COMMAND or DISPLAY_REGISTRATION_BEFORE_COMMAND parameter. If you enter Y, the messages are sent to the system console and the DTLLOG log. If you accept the default value of N, the messages are sent only to the DTLLOG log.

Registration-specific Command Parameters

You can specify multiple parameters for any registration-specific command that you include in the PWXSYSIN input for a PWXUCREG job.

The DATABASE_INSTANCE and REGISTRATION_NAME parameters are required.

Registration-specific command parameters override any corresponding global parameters.

In the following parameter descriptions, curly brackets indicate that one of the options must be entered, and underlining indicates the default value.

Parameter Descriptions

DATABASE_TYPE=*type*

Recommended. Specifies the type of source database and CDC that is associated with the registration or registrations that the PWXUCREG command processes. Options are:

- **ADA**. For Adabas.
- **DCM**. For Datacom table-based.
- **IDL**. For IDMS log-based.
- **IMS**. For IMS log-based. If you try to process registrations for IMS synchronous sources, the PWXUCREG utility rejects the registrations and ends with error message PWX-03723.

You can use the asterisk (*) wildcard, or a string followed by the wildcard such as B08*, in this parameter.

DATABASE_INSTANCE=*instance_id*

Required. Specifies one of the following values that is defined for the registration group in the PowerExchange Navigator:

- For Adabas, the collection identifier.
- For Datacom, the MUF name.
- For IDMS, the LOGSID identifier.
- For IMS, the RECON identifier.

This value is case-sensitive. You can use the asterisk (*) wildcard, or a string followed by the wildcard, in this parameter.

EPWD=*encrypted_password*

Specifies an encrypted password to use with the user ID that is specified in the associated USER parameter.

Consult with your system security administrator to determine the requirements for a user ID and a password or encrypted password based on your system security and the SECURITY statement in the DBMOVER configuration member.

Tip: You can use the PowerExchange Navigator to create an encrypted password. Click **File** > **Encrypt**.

If you specify this parameter, do not also specify the PWD parameter. If you specify both, the EPWD parameter takes precedence. Use the EPWD parameter if you are not allowed to store passwords in readable format.

Overrides the GLOBAL_EPWD parameter in a preceding SET_CONTROL_VALUE command, if specified.

GENERIC={N|Y}

Optonal. Enables you to issue registration-specific commands that change the registration status for multiple registrations that match a wildcard mask if you specify Y. You must also specify the asterisk (*) wildcard, or a string followed by the wildcard such as b80*, in one or more of the following registration-specific parameters: REGISTRATION_NAME, DATABASE_INSTANCE, and DATABASE_TYPE.

Default is N.

This parameter is not supported for any explicit DISPLAY_REGISTRATION command or automatic display operation based on the global DISPLAY_REGISTRATION_BEFORE_COMMAND and DISPLAY_REGISTRATION_AFTER_COMMAND parameters.

Overrides the global GENERIC parameter in a preceding SET_CONTROL_VALUE command.

PWD=*password*

Specifies a nonencrypted password to use with the user ID that is specified in the associated GLOBAL_USER parameter. This value is case-sensitive.

Consult with your system security administrator to determine the requirements for a user ID and a password or encrypted password based on your system security and the SECURITY statement in the DBMOVER configuration member.

If you specify this parameter, do not also specify the EPWD parameter. If you specify both, the EPWD parameter takes precedence.

Overrides the GLOBAL_PWD parameter in a preceding SET_CONTROL_VALUE command, if specified.

REGISTRATION_LOCATION={*hlq.data_set_name*|local}

Optional. Specifies the location of the VSAM CCT data set that contains the capture registrations. Default value is "local."

Usually, the default value of "local" is acceptable because the capture registrations must reside on the z/OS source system where the PWXUCREG utility runs.

Overrides the GLOBAL_REGISTRATION_LOCATION parameter in a preceding SET_CONTROL_VALUE command, if specified.

REGISTRATION_NAME=*user_registration_name*

Required. Specifies the user-defined name for the registration that is defined for the capture registration in the PowerExchange Navigator. This value is case-sensitive.

You can use the asterisk (*) wildcard, or a string followed by the wildcard, in this parameter.

USER=*user_id*

Optional. Specifies a user ID that has the authority to access capture registrations in the CCT data set on the source system.

Consult with your system security administrator to determine the requirements for a user ID and a password or encrypted password based on your system security and the SECURITY statement in the DBMOVER configuration member.

This user ID value is case-sensitive.

If you specify this parameter, also specify either the EPWD or PWD parameter. If you specify both, the EPWD parameter takes precedence.

Overrides the GLOBAL_USER parameter in a preceding SET_CONTROL_VALUE command, if specified.

VALIDATE={N|Y}

Optional. Validates the command syntax and reads the registrations that match the selection criteria to ensure that the command can process them, if you specify Y. This parameter does not actually run the command to change the registration status. Default is N, which disables validation.

Running the PWXUCREG Utility

After you define the JCL for the PWXUCREG job, you can manually submit the job or schedule it to run in an autmated manner as part of a batch job.

The following JCL includes the basic statements:

```
//PWXURACT JOB  'PWXUCREG',MSGLEVEL=(1,1),MSGCLASS=X,NOTIFY=&SYSUID,
//          CLASS=A
//LIBSRCH   JCLLIB ORDER=DTLUSR.V951.RUNLIB
//          SET HLQ=DTLUSR.V951
//          SET RUNLIB=DTLUSR.V951.RUNLIB
//STEP1     EXEC PGM=PWXUCREG
//STEPLIB   DD  DISP=SHR,DSN=&HLQ..LOADLIB
//*
//DTLAMCPR  DD  DISP=SHR,DSN=&HLQ..CCT
//DTLMSG    DD  DISP=SHR,DSN=&HLQ..DTLMSG
//DTLCFG    DD  DISP=SHR,DSN=&RUNLIB(DBMOVER)
//DTLKEY    DD  DISP=SHR,DSN=&RUNLIB(LICENSE)
//DTLSGN    DD  DISP=SHR,DSN=&RUNLIB(SIGNON)
//DTLLOG    DD  SYSOUT=*
//SYSUDUMP  DD  SYSOUT=*
//SYSOUT    DD  SYSOUT=*
//SYSPRINT  DD  SYSOUT=*
//PWXSYSIN  DD  *
  <pwxucreg commands and parameters>
/*
```

Examples of PWXUCREG Utility Commands

Use the example PWXUCREG commands to learn how to code simple commands and recognize their message output in the job log.

Example 1. Suspending a Capture Registration

You need to perform a cascade delete operation for a single registered source but do not want the ECCR to capture these deletes.

To stop change capture temporarily, you plan to suspend the capture registration named b800tbl that is associated with the source. You run a PWXUCREG job that contains a SUSPEND_REGISTRATION command. Later, you will reactivate the capture registration to resume capture processing.

Use the following JCL to run a PWXUCREG job that suspends the capture registration:

```
//PWXURSUS JOB  'PWXUCREG',MSGLEVEL=(1,1),MSGCLASS=X,NOTIFY=&SYSUID,
//          CLASS=A
//LIBSRCH   JCLLIB ORDER=DTLUSR.V951.RUNLIB
//          SET HLQ=DTLUSR.V951
//          SET RUNLIB=DTLUSR.V951.RUNLIB
//STEP1     EXEC PGM=PWXUCREG
```

```
//STEPLIB   DD  DISP=SHR,DSN=&HLQ..LOADLIB
//*
//DTLAMCPR  DD  DISP=SHR,DSN=&HLQ..CCT
//DTLMSG    DD  DISP=SHR,DSN=&HLQ..DTLMSG
//DTLCFG    DD  DISP=SHR,DSN=&RUNLIB(DBMOVER)
//DTLKEY    DD  DISP=SHR,DSN=&RUNLIB(LICENSE)
//DTLSGN    DD  DISP=SHR,DSN=&RUNLIB(SIGNON)
//DTLLOG    DD  SYSOUT=*
//SYSUDUMP  DD  SYSOUT=*
//SYSOUT    DD  SYSOUT=*
//SYSPRINT  DD  SYSOUT=*
//PWXSYSIN  DD  *
SET_CONTROL_VALUE,SYSTEM_CONSOLE_MESSAGES_COMMAND=Y;
SUSPEND_REGISTRATION,
DATABASE_INSTANCE=R11G4,
REGISTRATION_NAME=b800tb1
/*
```

The JCL includes the global SYSTEM_CONSOLE_MESSAGES_COMMAND=Y parameter in the SET_CONTROL_VALUE command to write messages related to suspension processing to the z/OS system console.

If the JCL is successfully processed, the following messages are printed to both the job log and z/OS system console:

```
16.22.35 JOB03118 ---- TUESDAY,   23 OCT 2012 ----
16.22.35 JOB03118  IRR010I  USERID DTLUSR   IS ASSIGNED TO THIS JOB.
16.22.35 JOB03118  ICH70001I DTLUSR   LAST ACCESS AT 14:01:58 ON TUESDAY, OCTOBER 23, 2012
16.22.35 JOB03118  £HASP373 PWXURSUS STARTED - INIT 3    - CLASS A - SYS MHZ1
16.22.35 JOB03118  IEF403I PWXURSUS - STARTED - TIME=16.22.35
16.22.37 JOB03118  PWX-03716 PWXUCREG: Registration "b800tb1", version "1", instance "R11G4", status changed from "Active" to
                        "Suspended"  .
16.22.37 JOB03118  -                                          --TIMINGS (MINS.)--            -----PAGING COUNTS----
16.22.37 JOB03118  -STEPNAME PROCSTEP    RC   EXCP   CONN    TCB    SRB  CLOCK   SERV  WORKLOAD  PAGE  SWAP   VIO SWAPS
16.22.37 JOB03118  -STEP1               00   1020    417    .00    .00     .0   5542  BATCH        0     0     0     0
16.22.38 JOB03118  IEF404I PWXURSUS - ENDED - TIME=16.22.38
16.22.38 JOB03118  -PWXURSUS ENDED.  NAME-                     TOTAL TCB CPU TIME=   .00  TOTAL ELAPSED TIME=    .0
16.22.38 JOB03118  £HASP395 PWXURSUS ENDED
------ JES2 JOB STATISTICS ------
  23 OCT 2012 JOB EXECUTION DATE
           24 CARDS READ
          142 SYSOUT PRINT RECORDS
            0 SYSOUT PUNCH RECORDS
            8 SYSOUT SPOOL KBYTES
         0.04 MINUTES EXECUTION TIME



PWX-33314 TIMEOUTS configuration parameter is
deprecated


PWX-15799 DD:PWXSYSIN <> PARM INPUT FILE:
START>>> .
PWX-15799
SET_CONTROL_VALUE,SYSTEM_CONSOLE_MESSAGES_COMMAND=Y;.
PWX-15799
SUSPEND_REGISTRATION,.
PWX-15799
DATABASE_INSTANCE=R11G4,.
PWX-15799
REGISTRATION_NAME=b800tb1.
PWX-15799 ;.

PWX-15799 DD:PWXSYSIN <> PARM INPUT FILE:
END(COMPLETE).




PWX-03716 PWXUCREG: Registration "b800tb1", version "1", instance "R11G4", status changed from "Active" to
"Suspended".


PWX-03717 PWXUCREG: Number of registrations processed
1  .


PWX-03712 PWXUCREG: Registration "b800tb1", type "DCM", instance "R11G4", version "1", current status
"S".
PWX-03713 PWXUCREG: Registration "b800tb1", suspended at "2012/10/23 16:22:37.235636", current time "2012/10/23
16:22:37.727037".
```

```
PWX-03714 PWXUCREG: Registration "b800tb1", activated at "2012/10/19 09:46:26.007478", current time "2012/10/23
16:22:37.727037".


PWX-03724 PWXUCREG: Number of registrations displayed 1.
```

Message PWX-03716 indicates that the utility successfully changed the registration status to Suspended. Messages PWX-03712 through PWX-03714 are printed because you accepted the default value of Y for global DISPLAY_REGISTRATION_AFTER_COMMAND parameter. These messages report registration information such as the current registration status, the suspension timestamp in current system time, and the activation timestamp from a previous ACTIVATE_REGISTRATION command.

Example 2. Reactivating a Capture Registration

You previously suspended a capture registration for a source to ignore cascade delete operations and now want to reactivate the registration to begin capturing changes again.

To close the suspension window and resume change capture for the capture registration named b800tbl, you run a PWXUCREG job that contains an ACTIVATE_REGISTRATION command.

Use the following JCL to run a PWXUCREG job that suspends the capture registration:

```
//PWXURACT JOB  'PWXUCREG',MSGLEVEL=(1,1),MSGCLASS=X,NOTIFY=&SYSUID,
//          CLASS=A
//LIBSRCH   JCLLIB ORDER=DTLUSR.V951.RUNLIB
//          SET HLQ=DTLUSR.V951
//          SET RUNLIB=DTLUSR.V951.RUNLIB
//STEP1     EXEC PGM=PWXUCREG
//STEPLIB   DD  DISP=SHR,DSN=&HLQ..LOADLIB
//*
//DTLAMCPR  DD  DISP=SHR,DSN=&HLQ..CCT
//DTLMSG    DD  DISP=SHR,DSN=&HLQ..DTLMSG
//DTLCFG    DD  DISP=SHR,DSN=&RUNLIB(DBMOVER)
//DTLKEY    DD  DISP=SHR,DSN=&RUNLIB(LICENSE)
//DTLSGN    DD  DISP=SHR,DSN=&RUNLIB(SIGNON)
//DTLLOG    DD  SYSOUT=*
//SYSUDUMP  DD  SYSOUT=*
//SYSOUT    DD  SYSOUT=*
//SYSPRINT  DD  SYSOUT=*
//PWXSYSIN  DD  *
SET_CONTROL_VALUE,SYSTEM_CONSOLE_MESSAGES_COMMAND=Y;
ACTIVATE_REGISTRATION,
DATABASE_INSTANCE=R11G4,
REGISTRATION_NAME=b800tb1
/*
```

The JCL includes the global SYSTEM_CONSOLE_MESSAGES_COMMAND=Y parameter in the SET_CONTROL_VALUE command to write messages related to reactivation processing to the z/OS system console.

If the JCL is successfully processed, the following messages are printed to both the job log and z/OS system console:

```
16.22.57 JOB03119 ---- TUESDAY,   23 OCT 2012 ----
16.22.57 JOB03119  IRR010I  USERID DTLUSR   IS ASSIGNED TO THIS JOB.
16.22.58 JOB03119  ICH70001I DTLUSR   LAST ACCESS AT 16:22:35 ON TUESDAY, OCTOBER 23, 2012
16.22.58 JOB03119  £HASP373 PWXURACT STARTED - INIT 3    - CLASS A - SYS MHZ1
16.22.58 JOB03119  IEF403I PWXURACT - STARTED - TIME=16.22.58
16.22.59 JOB03119  PWX-03716 PWXUCREG: Registration "b800tb1", version "1", instance "R11G4", status changed from "Suspended" to
                   "Active"  .
16.23.00 JOB03119  -                                          --TIMINGS (MINS.)--            -----PAGING COUNTS----
16.23.00 JOB03119  -STEPNAME PROCSTEP    RC   EXCP   CONN    TCB    SRB  CLOCK   SERV  WORKLOAD  PAGE  SWAP   VIO SWAPS
16.23.00 JOB03119  -STEP1               00   1019    346    .00    .00     .0   5350  BATCH        0     0     0     0
16.23.00 JOB03119  IEF404I PWXURACT - ENDED - TIME=16.23.00
16.23.00 JOB03119  -PWXURACT ENDED.  NAME-                     TOTAL TCB CPU TIME=   .00  TOTAL ELAPSED TIME=    .0
16.23.00 JOB03119  £HASP395 PWXURACT ENDED
------ JES2 JOB STATISTICS ------
  23 OCT 2012 JOB EXECUTION DATE
           24 CARDS READ
          142 SYSOUT PRINT RECORDS
            0 SYSOUT PUNCH RECORDS
            8 SYSOUT SPOOL KBYTES
```

```
         0.03 MINUTES EXECUTION TIME


PWX-33314 TIMEOUTS configuration parameter is deprecated

PWX-15799 DD:PWXSYSIN <> PARM INPUT FILE: START>>> .
PWX-15799 SET_CONTROL_VALUE,SYSTEM_CONSOLE_MESSAGES_COMMAND=Y;.
PWX-15799 ACTIVATE_REGISTRATION,.
PWX-15799 DATABASE_INSTANCE=R11G4,.
PWX-15799 REGISTRATION_NAME=b800tb1.
PWX-15799 ;.
PWX-15799 DD:PWXSYSIN <> PARM INPUT FILE: END(COMPLETE).


PWX-03716 PWXUCREG: Registration "b800tb1", version "1", instance "R11G4", status changed from "Suspended" to "Active"  .

PWX-03717 PWXUCREG: Number of registrations processed 1  .

PWX-03712 PWXUCREG: Registration "b800tb1", type "DCM", instance "R11G4", version "1", current status "A".
PWX-03713 PWXUCREG: Registration "b800tb1", suspended at "2012/10/23 16:22:37.235636", current time "2012/10/23 16:23:00.219928".
PWX-03714 PWXUCREG: Registration "b800tb1", activated at "2012/10/23 16:22:59.789082", current time "2012/10/23 16:23:00.219928".

PWX-03724 PWXUCREG: Number of registrations displayed
1.
```

Message PWX-03716 indicates that the utility successfully changed the registration status to Activated. Messages PWX-03712 through PWX-03714 are printed because you accepted the default value of Y for global DISPLAY_REGISTRATION_AFTER_COMMAND parameter. These messages report registration information such as the current registration status, the activation timestamp, and the suspension timestamp from the last SUSPEND_REGISTRATION command.

CHAPTER 20

PWXUDMX - Data Maps Update Time ECSA Memory Utility

This chapter includes the following topics:

PWXUDMX Utility Overview

Use the PWXUDMX utility to allocate, display, and delete ECSA memory, which holds time stamps of the latest updates to data maps files, and to modify the use counts of a file.

This processing is relevant if you configure data maps caching in multiple jobs mode by defining DMXCACHE_MULTIPLEJOBS=Y in the DBMOVER configuration file.

With the PWXUDMX utility, you can complete the following tasks:

- Allocate less than the 4096 bytes of ECSA memory that the system dynamically allocates.
- Delete ECSA memory.
- Display the contents of ECSA memory with file names and time stamps in legible format.
- Display the contents of ECSA memory in hexadecimal format.
- If a PowerExchange Listener or netport job does not shut down cleanly, decrement the use count of a file.
- Increment the use count of a file.

Supported Operating Systems for the PWXUDMX Utility

The PWXUDMX utility runs on z/OS systems.

Running the PWXUDMX Utility on z/OS

You run the PWXUDMX utility by submitting the PWXUDMX job. The input control statements for this utility are read from SYSIN.

The following is an example of JCL to use when you run this utility on z/OS.

```
//jobname  JOB  'UDMX        ',MSGLEVEL=(1,1),MSGCLASS=X,
//          CLASS=A,NOTIFY=&SYSUID
//*
//STEP1     EXEC PGM=PWXUDMX,
//          PARM='CMD=command'
//STEPLIB   DD  DSN=CEE.SCEERUN,DISP=SHR
//          DD  DSN=&HLQ..LOADLIB,DISP=SHR
//DATAMAP   DD  DSN=&HLQ..V1.DATAMAPS,DISP=SHR
//DTLMSG    DD  DSN=&HLQ..DTLMSG,DISP=SHR
//DTLCFG    DD  DSN=&HLQ..RUNLIB(DBMOVER),DISP=SHR
//DTLKEY    DD  DSN=&HLQ..RUNLIB(LICENSE),DISP=SHR
//DTLSGN    DD  DSN=&HLQ..RUNLIB(SIGNON),DISP=SHR
//DTLOUT    DD  SYSOUT=*
//SYSPRINT  DD  SYSOUT=*
//DTLLOG    DD  SYSOUT=*
//SYSOUT    DD  SYSOUT=*
//SYSPRINT  DD  SYSOUT=*
//CEEDUMP   DD  SYSOUT=*
//SYSUDUMP  DD  SYSOUT=*
//DTLCFG    DD  *
```

The JCL statements are:

JOB

Defines the PWXUDMX job card to z/OS, including the job name.

EXEC PGM=PWXUDMX

Identifies the name of the program, PWXUDMX, to run.

PARM='CMD=*command*'

Identifies the name of the PWXUDMX command to run. For a description of the commands, see "PWXUDMX Commands and Parameters" on page 246.

STEPLIB DD

Defines the PowerExchange LOAD library that contains the utility.

DATAMAP DD

If you do not specify the optional FILE parameter for the DECREMENT_FILE_COUNT or INCREMENT_FILE_COUNT command, PowerExchange modifies the file count of the file specified in the DATAMAP DD statement.

SYSPRINT DD

Defines the print location for the output of the DISPLAY_ECSA or DUMP_ECSA commands.

PWXUDMX Commands and Parameters

This section describes the commands that you can enter in the CMD statement of the PWXUDMX syntax and the command-specific parameters.

CREATE_ECSA Command

Creates ECSA memory.

```
//STEP1 EXEC PGM=PWXUDMX,
//      PARM='CMD=CREATE_ECSA [LENGTH=length]'
```

To allocate less than the 4096 bytes of ECSA memory that the system dynamically allocates, issue this command with the optional LENGTH parameter. For the *length* variable, specify the amount of storage to allocate for ECSA memory. The PWXUDMX utility allocates the specified amount of ECSA memory and creates a named token called PWX_DMXTIME_1, which defines the address of the ECSA memory.

Maximum value is 4096 bytes.

DECREMENT_FILE_COUNT Command

Decrements the use count of a file.

```
//STEP1 EXEC PGM=PWXUDMX,
//PARM='CMD=DECREMENT_FILE_COUNT [FILE=file]'
```

If a PowerExchange Listener or netport job does not shut down cleanly, issue this command to correct the use count.

If you do not specify the optional FILE parameter, the PWXUDMX utility decrements the use count of the file specified in the DATAMAP DD statement.

DELETE_ECSA Command

Deletes ECSA memory.

```
//STEP1 EXEC PGM=PWXUDMX,
//      PARM='CMD=DELETE_ECSA [FORCE={N|Y}]'
```

Issue this command to delete ECSA memory when you uninstall PowerExchange.
If ECSA memory has a nonzero use count, use either the DECREMENT_FILE_COUNT command to reduce the use count or the optional FORCE=Y parameter on the DELETE_ECSA command to force deletion of ECSA memory.

When you run the DELETE_ECSA command, the PWXUDMX utility deletes the ECSA memory and deletes the named token called PWX_DMXTIME_1, which defines the address of the ECSA memory.

DISPLAY_ECSA Command

Displays ECSA memory.

```
//STEP1 EXEC PGM=PWXUDMX,
//      PARM='CMD=DISPLAY_ECSA'
```

Display the contents of ECSA memory, including file names and time stamps in legible format.

DUMP_ECSA Command

Displays ECSA memory in hexadecimal format.

```
//STEP1 EXEC PGM=PWXUDMX,
//      PARM='CMD=DUMP_ECSA'
```

INCREMENT_FILE_COUNT Command

Increments the use count of a file.

```
//STEP1 EXEC PGM=PWXUDMX,
//      PARM='CMD=INCREMENT_FILE_COUNT [FILE=file]'
```

If you do not specify the optional FILE parameter, PowerExchange increments the file count of the file specified in the DATAMAP DD statement.

CHAPTER 21

PWXUSSL - PowerExchange SSL Reporting Utility

This chapter includes the following topics:

PWXUSSL Utility Overview

Use the PWXUSSL utility to generate reports about SSL libraries and certificates on Linux, UNIX, and Windows.

You can generate the following reports:

Certificate report

Reports information from a certificate chain file. The report can include multiple certificates in a PEM chain file.

Ciphers report

Reports the cipher suites that are available in the OpenSSL cryptographic library. The report includes the hexidecimal codes that you can use to correlate OpenSSL cipher suites to the AT-TLS cipher suites on z/OS.

Version report

Reports the version of OpenSSL that was used to build the cryptographic library. On Linux and UNIX, the cryptographic library file is named **libpmcrypto**. On Windows, the file is named **PMLIBEAY32.DLL**. The report includes the date of the build and compiler settings.

Supported Operating Systems for the PWXUSSL Utility

The PWXUSSL utility runs on computers that have the following operating systems:

- Linux
- UNIX
- Windows

Running the PWXUSSL Utility

You can run a PWXUSSL utility command from a command line on a system where PowerExchange is installed.

Navigate to the directory where the pwxussl executable is located. By default, this directory is in the PowerExchange installation directory. Then enter `pwxussl` followed by a command and any parameters, as follows:

```
C:\Informatica\PowerExchangev.r.m pwxussl CMD=command_name parameters
```

Use one of the following values for *command_name*:

- REPORT_VERSION. Generates the version report.
- REPORT_CIPHERS. Generates the ciphers report.
- REPORT_CERTIFICATE. Generates the certificate report.

Certificate Report

The certificate report provides information from a certificate chain file.

To generate a certificate report, enter the following command:

```
C:\Informatica\PowerExchangev.r.m pwxussl CMD=REPORT_CERTIFICATE infile=infle
```

The following output is an example of a certificate report:

```
Processing console program. pwxussl cmd=report_certificate infile=c:\OpenSSL-win32\bin
\PEM\client.pem
REPORT FOR COMMAND REPORT_CERTIFICATE

File contains 1 X509 certificates and 1 subject names.
Certificate 1. Subject Name "/C=AU/ST=Queensland/O=CryptSoft Pty Ltd/CN=Client test cert
(512 bit)"


Certificate 1. Serial "02". Version "1 (0x0)".
Valid from "2097-06-09 13:57:56" time zone "Z". Valid to "2098-06-09 13:57:56" time zone
"Z".
Certificate has expired
Subject name "/C=AU/ST=Queensland/O=CryptSoft Pty Ltd/CN=Client test cert (512 bit)"
Issuer  name "/C=AU/ST=Queensland/O=CryptSoft Pty Ltd/CN=Test CA (1024 bit)"
Signature algorithm "md5WithRSAEncryption"
Public Key algorithm "rsaEncryption". Size 512 bits.
```

```
***** START OF RESULT SET FROM API X509_print_ex_fp *****
Certificate:
    Data:
        Version: 1 (0x0)
        Serial Number: 2 (0x2)
        Signature Algorithm: md5WithRSAEncryption
        Issuer: C=AU, ST=Queensland, O=CryptSoft Pty Ltd, CN=Test CA (1024 bit)
        Validity
            Not Before: Jun  9 13:57:56 1997 GMT
            Not After : Jun  9 13:57:56 1998 GMT
        Subject: C=AU, ST=Queensland, O=CryptSoft Pty Ltd, CN=Client test cert (512 bit)
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
            RSA Public Key: (512 bit)
                Modulus (512 bit):
                    00:bb:6f:e7:94:32:cc:6e:a2:d8:f9:70:67:5a:5a:
                    87:bf:be:1a:ff:0b:e6:3e:87:9f:2a:ff:b9:36:44:
                    d4:d2:c6:d0:00:43:0d:ec:66:ab:f4:78:29:e7:4b:
                    8c:51:08:62:3a:1c:0e:e8:be:21:7b:3a:d8:d3:6d:
                    5e:b4:fc:a1:d9
                Exponent: 65537 (0x10001)
    Signature Algorithm: md5WithRSAEncryption
        70:b4:c9:88:ee:81:94:1b:c9:ca:99:fb:50:b2:c0:13:56:21:
        f9:35:14:6d:a0:4c:34:ec:3c:49:a7:f2:df:6a:d1:dd:ae:1a:
        90:07:bd:de:19:d2:f9:58:82:d9:25:79:38:e9:7c:f6:7b:d5:
        8c:49:48:d5:09:26:21:74:ac:6d:7e:55:37:51:1d:80:8e:fd:
        4e:a3:4b:13:35:d7:f3:d3:00:ea:24:d8:ab:2c:db:73:ca:18:
        6c:6a:af:2a:31:3a:cb:c1:7a:c2:3f:7d:55:c4:18:a2:80:54:
        90:49:41:67:67:24:c4:f5:32:b0:85:2e:06:97:06:ed:09:fc:
        52:29
***** END OF RESULT SET FROM API X509_print_ex_fp *****

Private key information MIIBOwIBAAJBALtv55QyzG6i2PlwZ1pah7++Gv8L5j6Hnyr/uTZE1NLG0ABDDexm;
q/R4KedLjFEIYjocDui+IXs62NNtXrT8odkCAwEAAQJAbwXq0vJ/+uyEvsNgxLko
```

Ciphers Report

The ciphers report lists the ciphers that are available in the OpenSSL cryptographic library.

To generate a cipers report, enter the following command:

```
C:\Informatica\PowerExchangev.r.m pwxussl CMD=REPORT_CIPHERS
```

The following output is an example of a ciphers report:

```
REPORT FOR COMMAND REPORT_CIPHERS
35 available ciphers

Ciphers Report for Hex Id, Strength and Version

 0 DHE-RSA-AES256-SHA          hex id=39 strength=0081 version=3
 1 DHE-DSS-AES256-SHA          hex id=38 strength=0081 version=3
 2 AES256-SHA                  hex id=35 strength=0081 version=3
 3 EDH-RSA-DES-CBC3-SHA        hex id=16 strength=0081 version=3
 4 EDH-DSS-DES-CBC3-SHA        hex id=13 strength=0081 version=3
 5 DES-CBC3-SHA                hex id=0A strength=0081 version=3
 6 DES-CBC3-MD5                hex id=07 strength=0081 version=2
 7 DHE-RSA-AES128-SHA          hex id=33 strength=0081 version=3
 8 DHE-DSS-AES128-SHA          hex id=32 strength=0081 version=3
 9 AES128-SHA                  hex id=2F strength=0081 version=3
10 IDEA-CBC-SHA                hex id=07 strength=0041 version=3
11 IDEA-CBC-MD5                hex id=05 strength=0041 version=2
12 RC2-CBC-MD5                 hex id=03 strength=0041 version=2
13 DHE-DSS-RC4-SHA             hex id=66 strength=0041 version=3
14 RC4-SHA                     hex id=05 strength=0041 version=3
```

```
15 RC4-MD5                     hex id=04 strength=0041 version=3
16 RC4-MD5                     hex id=01 strength=0041 version=2
17 RC4-64-MD5                  hex id=08 strength=0021 version=2
18 EXP1024-DHE-DSS-DES-CBC-SHA hex id=63 strength=0012 version=3
19 EXP1024-DES-CBC-SHA         hex id=62 strength=0012 version=3
20 EXP1024-RC2-CBC-MD5         hex id=61 strength=0012 version=3
21 EDH-RSA-DES-CBC-SHA         hex id=15 strength=0021 version=3
22 EDH-DSS-DES-CBC-SHA         hex id=12 strength=0021 version=3
23 DES-CBC-SHA                 hex id=09 strength=0021 version=3
24 DES-CBC-MD5                 hex id=06 strength=0021 version=2
25 EXP1024-DHE-DSS-RC4-SHA     hex id=65 strength=0012 version=3
26 EXP1024-RC4-SHA             hex id=64 strength=0012 version=3
27 EXP1024-RC4-MD5             hex id=60 strength=0012 version=3
28 EXP-EDH-RSA-DES-CBC-SHA     hex id=14 strength=000A version=3
29 EXP-EDH-DSS-DES-CBC-SHA     hex id=11 strength=000A version=3
30 EXP-DES-CBC-SHA             hex id=08 strength=000A version=3
31 EXP-RC2-CBC-MD5             hex id=06 strength=000A version=3
32 EXP-RC2-CBC-MD5             hex id=04 strength=000A version=2
33 EXP-RC4-MD5                 hex id=03 strength=000A version=3
34 EXP-RC4-MD5                 hex id=02 strength=000A version=2
```

Ciphers Report for Key Exchange, Encryption, Signature and Message Authentification

```
 0 DHE-RSA-AES256-SHA          Key Ex=DH        Enc=AES(256)  Au=RSA MAC=SHA1
 1 DHE-DSS-AES256-SHA          Key Ex=DH        Enc=AES(256)  Au=DSS MAC=SHA1
 2 AES256-SHA                  Key Ex=RSA       Enc=AES(256)  Au=RSA MAC=SHA1
 3 EDH-RSA-DES-CBC3-SHA        Key Ex=DH        Enc=3DES(168) Au=RSA MAC=SHA1
 4 EDH-DSS-DES-CBC3-SHA        Key Ex=DH        Enc=3DES(168) Au=DSS MAC=SHA1
 5 DES-CBC3-SHA                Key Ex=RSA       Enc=3DES(168) Au=RSA MAC=SHA1
 6 DES-CBC3-MD5                Key Ex=RSA       Enc=3DES(168) Au=RSA MAC=MD5
 7 DHE-RSA-AES128-SHA          Key Ex=DH        Enc=AES(128)  Au=RSA MAC=SHA1
 8 DHE-DSS-AES128-SHA          Key Ex=DH        Enc=AES(128)  Au=DSS MAC=SHA1
 9 AES128-SHA                  Key Ex=RSA       Enc=AES(128)  Au=RSA MAC=SHA1
10 IDEA-CBC-SHA                Key Ex=RSA       Enc=IDEA(128) Au=RSA MAC=SHA1
11 IDEA-CBC-MD5                Key Ex=RSA       Enc=IDEA(128) Au=RSA MAC=MD5
12 RC2-CBC-MD5                 Key Ex=RSA       Enc=RC2(128)  Au=RSA MAC=MD5
13 DHE-DSS-RC4-SHA             Key Ex=DH        Enc=RC4(128)  Au=DSS MAC=SHA1
14 RC4-SHA                     Key Ex=RSA       Enc=RC4(128)  Au=RSA MAC=SHA1
15 RC4-MD5                     Key Ex=RSA       Enc=RC4(128)  Au=RSA MAC=MD5
16 RC4-MD5                     Key Ex=RSA       Enc=RC4(128)  Au=RSA MAC=MD5
17 RC4-64-MD5                  Key Ex=RSA       Enc=RC4(64)   Au=RSA MAC=MD5
18 EXP1024-DHE-DSS-DES-CBC-SHA Key Ex=DH(1024)  Enc=DES(56)   Au=DSS MAC=SHA1 export
19 EXP1024-DES-CBC-SHA         Key Ex=RSA(1024) Enc=DES(56)   Au=RSA MAC=SHA1 export
20 EXP1024-RC2-CBC-MD5         Key Ex=RSA(1024) Enc=RC2(56)   Au=RSA MAC=MD5  export
21 EDH-RSA-DES-CBC-SHA         Key Ex=DH        Enc=DES(56)   Au=RSA MAC=SHA1
22 EDH-DSS-DES-CBC-SHA         Key Ex=DH        Enc=DES(56)   Au=DSS MAC=SHA1
23 DES-CBC-SHA                 Key Ex=RSA       Enc=DES(56)   Au=RSA MAC=SHA1
24 DES-CBC-MD5                 Key Ex=RSA       Enc=DES(56)   Au=RSA MAC=MD5
25 EXP1024-DHE-DSS-RC4-SHA     Key Ex=DH(1024)  Enc=RC4(56)   Au=DSS MAC=SHA1 export
26 EXP1024-RC4-SHA             Key Ex=RSA(1024) Enc=RC4(56)   Au=RSA MAC=SHA1 export
27 EXP1024-RC4-MD5             Key Ex=RSA(1024) Enc=RC4(56)   Au=RSA MAC=MD5  export
28 EXP-EDH-RSA-DES-CBC-SHA     Key Ex=DH(512)   Enc=DES(40)   Au=RSA MAC=SHA1 export
29 EXP-EDH-DSS-DES-CBC-SHA     Key Ex=DH(512)   Enc=DES(40)   Au=DSS MAC=SHA1 export
30 EXP-DES-CBC-SHA             Key Ex=RSA(512)  Enc=DES(40)   Au=RSA MAC=SHA1 export
31 EXP-RC2-CBC-MD5             Key Ex=RSA(512)  Enc=RC2(40)   Au=RSA MAC=MD5  export
32 EXP-RC2-CBC-MD5             Key Ex=RSA(512)  Enc=RC2(40)   Au=RSA MAC=MD5  export
33 EXP-RC4-MD5                 Key Ex=RSA(512)  Enc=RC4(40)   Au=RSA MAC=MD5  export
34 EXP-RC4-MD5                 Key Ex=RSA(512)  Enc=RC4(40)   Au=RSA MAC=MD5  export
```

Version Report

The version report reports the version of OpenSSL that was used to build the cryptographic library.

To generate a version report, enter the following command:

```
C:\Informatica\PowerExchangev.r.m pwxussl CMD=REPORT_VERSION
```

The following output is an example of a version report:

```
REPORT FOR COMMAND REPORT_VERSION

SSLEAY_VERSION       = OpenSSL 0.9.8a 11 Oct 2005
SSLEAY_BUILT_ON      = built on: Fri Feb 27 23:04:22 2009
SSLEAY_PLATFORM      = platform: VC-WIN32
SSLEAY_DIR           = OPENSSLDIR: "/usr/local/ssl"

SSLEAY_CFLAGS
compiler: icl  /MD /Ox /O2 /Ob2 /W3 /WX /Gs0 /GF /Gy /nologo -DOPENSSL_SYSNAME_WIN32 -
DWIN32_LEAN_AND_MEAN -DL_ENDIAN -DDSO_WIN32 -D_CRT_SECURE_NO_DEPRECATE -
DOPENSSL_USE_APPLINK -I. /Fdout32dll -DOPENSSL_NO_RC5 -DOPENSSL_NO_MDC2 -DOPENSSL_NO_KRB5
```

INDEX

E

F

G

H

I

L

M

O

P

R

S

T

U

V

X