Navigator User Guide

Informatica® PowerExchange®
(Version 8.6)
## Table of Contents

Preface ................................................................. ix

Informatica Resources ............................................. ix
  Informatica Customer Portal ..................................... ix
  Informatica Documentation ........................................ x
  Informatica Web Site .............................................. x
  Informatica Knowledge Base ..................................... x
  Informatica Global Customer Support ........................ x

Chapter 1: Using the PowerExchange Navigator ............. 1
  Overview .............................................................. 1
  Preparing to Use the PowerExchange Navigator .......... 2
    Installing PowerExchange on Windows .................... 2
    Configuring Windows ODBC Data Sources ............... 2
    Starting the PowerExchange Listener on Windows .... 2
    Testing the Windows Listener ............................... 3
    Testing a Remote Listener ................................... 3
  Preparing to Access Sources and Targets ................. 3
    Sourcing Relational Data ....................................... 3
    Sourcing MVS VSAM Data Sets and Flat Files from any Platform ................. 4
    Sourcing Data from IMS ......................................... 4
    Sourcing Data from Adabas .................................... 5
    Sourcing Data from IDMS ....................................... 5
    Sourcing Data from Datacom ................................... 6
    Targeting MVS VSAM Data Sets or Flat Files on any Platform .................... 6
    Targeting Adabas ................................................. 6
    Targeting IMS .................................................... 7
    Targeting C-ISAM Data ......................................... 7
    Targeting Relational Data ..................................... 8
  Defining the Resource Configuration ....................... 8
    Current Configuration Tab .................................... 8
    Edit Configuration Tab ....................................... 8
    Shared Resource ............................................... 8
  Selecting Views .................................................. 9

Chapter 2: Creating Data Maps ................................ 11
  Data Map Concepts ............................................... 11
    Structure and Content of a Data Map .................... 12
  Creating Data Maps - Examples .............................. 12
    A Simple Example .............................................. 12
    A Second Example .............................................. 16
    A Third Example .............................................. 18
  Creating Adabas Data Maps .................................. 20
<table>
<thead>
<tr>
<th>Testing Adabas Data Maps</th>
<th>23</th>
</tr>
</thead>
<tbody>
<tr>
<td>Creating Datacom Data Maps</td>
<td>24</td>
</tr>
<tr>
<td>Testing Datacom Data Maps</td>
<td>25</td>
</tr>
<tr>
<td>Datacom Record Properties Dialog</td>
<td>25</td>
</tr>
<tr>
<td>Creating DB2 Data Maps</td>
<td>26</td>
</tr>
<tr>
<td>Creating Data Maps Using the DB2 Catalog</td>
<td>27</td>
</tr>
<tr>
<td>Creating Data Maps Using DB2 Unload Files</td>
<td>29</td>
</tr>
<tr>
<td>Amending the Record Layout</td>
<td>32</td>
</tr>
<tr>
<td>Creating IDMS Data Maps</td>
<td>32</td>
</tr>
<tr>
<td>Adding IDMS Data Maps</td>
<td>33</td>
</tr>
<tr>
<td>Database Row Test</td>
<td>34</td>
</tr>
<tr>
<td>Changing IDMS Record Properties</td>
<td>35</td>
</tr>
<tr>
<td>Adding a New Table to an IDMS Data Map</td>
<td>37</td>
</tr>
<tr>
<td>Adding Expressions</td>
<td>37</td>
</tr>
<tr>
<td>Creating IMS Data Maps</td>
<td>37</td>
</tr>
<tr>
<td>Data Map Tasks</td>
<td>37</td>
</tr>
<tr>
<td>Data Map Structure</td>
<td>38</td>
</tr>
<tr>
<td>Examples</td>
<td>40</td>
</tr>
<tr>
<td>Prerequisites for Creating IMS Data Maps</td>
<td>40</td>
</tr>
<tr>
<td>Adding IMS Data Maps Using PowerExchange Navigator</td>
<td>42</td>
</tr>
<tr>
<td>Importing IMS DBD Source for IMS Data Maps</td>
<td>43</td>
</tr>
<tr>
<td>Returning an IMS RBA</td>
<td>46</td>
</tr>
<tr>
<td>Modifying IMS Data Maps</td>
<td>46</td>
</tr>
<tr>
<td>Storing IMS Data Maps</td>
<td>46</td>
</tr>
<tr>
<td>Using Lookup Transformations with IMS Databases</td>
<td>47</td>
</tr>
<tr>
<td>Writing Data to IMS Databases</td>
<td>47</td>
</tr>
<tr>
<td>Testing IMS Data Maps</td>
<td>49</td>
</tr>
<tr>
<td>IMS Data Map Examples</td>
<td>50</td>
</tr>
<tr>
<td>Creating VSAM Data Maps</td>
<td>52</td>
</tr>
<tr>
<td>Improving Bulk Read Performance for VSAM Data Sets</td>
<td>55</td>
</tr>
<tr>
<td>Using Offload Processing for VSAM Data Sets</td>
<td>57</td>
</tr>
<tr>
<td>Retrieving the RRN or RBA for Records in VSAM Data Sets</td>
<td>57</td>
</tr>
<tr>
<td>Testing VSAM Data Maps</td>
<td>58</td>
</tr>
<tr>
<td>Creating C-ISAM Data Maps</td>
<td>59</td>
</tr>
<tr>
<td>Creating MQSeries Data Maps</td>
<td>61</td>
</tr>
<tr>
<td>Data Map Code Pages</td>
<td>63</td>
</tr>
<tr>
<td>PowerExchange Code Pages</td>
<td>63</td>
</tr>
</tbody>
</table>

**Chapter 3: Importing Copybooks**

<table>
<thead>
<tr>
<th>Importing COBOL Copybooks</th>
<th>65</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data File Input</td>
<td>65</td>
</tr>
<tr>
<td>Data File Input Viewed via Copybook</td>
<td>66</td>
</tr>
<tr>
<td>COBOL Copybook with OCCURS</td>
<td>66</td>
</tr>
<tr>
<td>Output Row to be Constructed:</td>
<td>66</td>
</tr>
<tr>
<td>Loading a COBOL Copybook</td>
<td>66</td>
</tr>
<tr>
<td>Importing COBOL Copybooks Containing Redefines</td>
<td>71</td>
</tr>
</tbody>
</table>
Chapter 4: Managing Data Maps ............................................................. 93
Data Checking ................................................................................. 93
Importing Exported Data Maps ....................................................... 95
Searching in Data Maps ................................................................. 96
Sorting Data Map Records and Tables ............................................ 96
File List Processing ....................................................................... 97
Restrictions ................................................................................... 97
ODBC ............................................................................................ 97
Filelist File .................................................................................... 97
Example Filelist Files .................................................................. 98
Example Scenario ......................................................................... 98
Chapter 5: Working with Personal Metadata .................................... 101
Overview ....................................................................................... 101
Creating a Personal Metadata Profile ........................................... 101
Step 1. Add Personal Metadata Profile ......................................... 101
Step 2. Enter Personal Metadata Filters ....................................... 102
Step 3. Explore Metadata .............................................................. 103
Step 4. Perform Row Test .............................................................. 104
Metadata Column Display ............................................................ 104
Search Facility for Metadata Display .......................................... 105
Chapter 6: Using Database Row Test ............................................ 107
Database Row Test Overview ......................................................... 107
Performing a Database Row Test .................................................. 107
Specifying Access Methods .......................................................... 108
PowerExchange SQL for Metadata Requests ............................. 108
Specifying Advanced Parameters ............................................... 109
<table>
<thead>
<tr>
<th>Chapter</th>
<th>Working with Capture Restart Tokens</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Chapter 7: Working with Capture Registrations</strong></td>
<td></td>
<td><strong>115</strong></td>
</tr>
<tr>
<td>Capture Registration Overview</td>
<td></td>
<td><strong>115</strong></td>
</tr>
<tr>
<td>Adding Registration Groups</td>
<td></td>
<td><strong>115</strong></td>
</tr>
<tr>
<td>Adding Capture Registrations</td>
<td></td>
<td><strong>118</strong></td>
</tr>
<tr>
<td>Defining the Condense Option</td>
<td></td>
<td><strong>121</strong></td>
</tr>
<tr>
<td>Adding a Capture Registration to an Existing Group</td>
<td></td>
<td><strong>122</strong></td>
</tr>
<tr>
<td>Deleting a Registration Group</td>
<td></td>
<td><strong>122</strong></td>
</tr>
<tr>
<td>Deleting a Registration Entry</td>
<td></td>
<td><strong>122</strong></td>
</tr>
<tr>
<td>Viewing a Registration Group</td>
<td></td>
<td><strong>123</strong></td>
</tr>
<tr>
<td>Viewing a Capture Registration</td>
<td></td>
<td><strong>123</strong></td>
</tr>
<tr>
<td>Resource Explorer</td>
<td></td>
<td><strong>123</strong></td>
</tr>
<tr>
<td>Capture Registration Details</td>
<td></td>
<td><strong>123</strong></td>
</tr>
<tr>
<td>Modifying Capture Registrations</td>
<td></td>
<td><strong>124</strong></td>
</tr>
<tr>
<td>Changing a Capture Registration</td>
<td></td>
<td><strong>124</strong></td>
</tr>
<tr>
<td>Changing Properties of the Registration Definition</td>
<td></td>
<td><strong>126</strong></td>
</tr>
<tr>
<td>Setting to Active</td>
<td></td>
<td><strong>126</strong></td>
</tr>
<tr>
<td>Setting to History</td>
<td></td>
<td><strong>127</strong></td>
</tr>
<tr>
<td><strong>Chapter 8: Working with Capture Extraction Maps</strong></td>
<td></td>
<td><strong>129</strong></td>
</tr>
<tr>
<td>Extraction Maps Overview</td>
<td></td>
<td><strong>129</strong></td>
</tr>
<tr>
<td>Adding Extraction Groups</td>
<td></td>
<td><strong>130</strong></td>
</tr>
<tr>
<td>Adding Extraction Maps</td>
<td></td>
<td><strong>132</strong></td>
</tr>
<tr>
<td>Adding an Extraction Map to an Existing Extraction Group</td>
<td></td>
<td><strong>134</strong></td>
</tr>
<tr>
<td>Deleting an Extraction Group</td>
<td></td>
<td><strong>134</strong></td>
</tr>
<tr>
<td>Deleting an Extraction Map</td>
<td></td>
<td><strong>134</strong></td>
</tr>
<tr>
<td>Viewing an Extraction Map</td>
<td></td>
<td><strong>134</strong></td>
</tr>
<tr>
<td>Removing Columns from the Extraction Map</td>
<td></td>
<td><strong>136</strong></td>
</tr>
<tr>
<td>Changing the Capture Registration Associated with an Extraction Map</td>
<td></td>
<td><strong>137</strong></td>
</tr>
<tr>
<td>Version Indicator in Extraction Maps</td>
<td></td>
<td><strong>138</strong></td>
</tr>
<tr>
<td>Using Change Indicator and Before Image Columns</td>
<td></td>
<td><strong>139</strong></td>
</tr>
<tr>
<td>Selecting CI and BI Columns in an Extraction Map</td>
<td></td>
<td><strong>139</strong></td>
</tr>
<tr>
<td>Testing the Column Changes Using Database Row Test</td>
<td></td>
<td><strong>140</strong></td>
</tr>
<tr>
<td><strong>Chapter 9: Working with Capture Applications</strong></td>
<td></td>
<td><strong>141</strong></td>
</tr>
<tr>
<td>Application Groups</td>
<td></td>
<td><strong>141</strong></td>
</tr>
<tr>
<td>Reset to New Start Point</td>
<td></td>
<td><strong>141</strong></td>
</tr>
<tr>
<td><strong>Chapter 10: Working with User-Defined Fields</strong></td>
<td></td>
<td><strong>143</strong></td>
</tr>
<tr>
<td>Overview</td>
<td></td>
<td><strong>143</strong></td>
</tr>
<tr>
<td>Expressions Tab</td>
<td></td>
<td><strong>143</strong></td>
</tr>
<tr>
<td>Expression Editor Dialog Box</td>
<td></td>
<td><strong>145</strong></td>
</tr>
<tr>
<td>Creating a User-Defined Field</td>
<td></td>
<td><strong>146</strong></td>
</tr>
</tbody>
</table>
Available Functions ................................................................. 147
CallProg ................................................................. 147
Check ................................................................. 148
Concat ................................................................. 149
Copydata ................................................................. 149
Fragment ................................................................. 149
GenVRowKey ................................................................. 150
GetDatabaseKey ................................................................. 150
GetDataFlowType ................................................................. 151
GetDbKey ................................................................. 152
GetDbKeyOwner ................................................................. 152
GetFullDbKey ................................................................. 152
GetFullDbKeyOwner ................................................................. 153
GetIMSRBAByLevel ................................................................. 153
GetPageGroup ................................................................. 154
GetPgGrpAndRdx ................................................................. 154
GetPgGrpOwner ................................................................. 154
GetRadix ................................................................. 154
GetRadixOwner ................................................................. 155
GetPgGrpAndRdxOwner ................................................................. 155
GetSeqWithinLevel ................................................................. 155
GetSeqWithinParent ................................................................. 156
LengthOf ................................................................. 157
LTrim ................................................................. 157
RTrim ................................................................. 157
Split ................................................................. 158
Strip ................................................................. 159
SetBitA ................................................................. 159
SetBitN ................................................................. 159
ToLower ................................................................. 160
ToUpper ................................................................. 160
Using External Programs with CALLPROG ................................................................. 161
Invoking External Programs ................................................................. 161
Result Argument and Error Handling ................................................................. 162
External C Routines ................................................................. 163
External COBOL Routines ................................................................. 163
External OS Routines ................................................................. 164
External OS400 Routines ................................................................. 165
External PL/I Routines ................................................................. 166
Using SQL with User-Defined Fields ................................................................. 167

Chapter 11: Creating User Access Method Programs ......................... 169
Overview ................................................................. 169
Explanation of Input and Return Parameter ................................................................. 170
Explanation of Parameter Structure ................................................................. 171
Example Programs ................................................................. 172
Assembler Example. .............................................................. 173
C Example. ................................................................. 176
COBOL Example. ......................................................... 179

Index. ................................................................. 183
Preface

This manual explains how the PowerExchange Navigator is used to build and maintain your PowerExchange installation.

This manual 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

The PowerExchange Navigator comprises the following main components:

- **Data maps.** The relational to non-relational definitions that are used to access data sources like IMS databases, VSAM files and flat files.
- **Personal metadata.** Parameters to browse metadata from remote databases.
- **Change Data Capture, if installed.**

PowerExchange Navigator runs on Windows 2000 and XP. PowerExchange Listener is installed as a Windows Service and communicates status information through the System Tray, with start and stop PowerExchange Listener applications being provided as standard. The latest Microsoft Controls are used, so the key Microsoft DLLs that have been used to develop PowerExchange are shipped along with the Informatica developed binaries.

Informatica Resources

**Informatica Customer Portal**

As an Informatica customer, you can access the Informatica Customer Portal site at http://my.informatica.com. The site contains product information, user group information, newsletters, access to the Informatica customer support case management system (ATLAS), the Informatica Knowledge Base, Informatica Documentation Center, and access to the Informatica user community.
Informatica Documentation

The Informatica Documentation team takes every effort to create accurate, usable documentation. If you have questions, comments, or ideas about this documentation, contact the Informatica Documentation team through email at infra_documentation@informatica.com. We will use your feedback to improve our documentation. Let us know if we can contact you regarding your comments.

Informatica Web Site

You can access the Informatica corporate web site at http://www.informatica.com. The site contains information about Informatica, its background, upcoming events, and sales offices. You will also find product and partner information. The services area of the site includes important information about technical support, training and education, and implementation services.

Informatica Knowledge Base

As an Informatica customer, you can access the Informatica Knowledge Base at http://my.informatica.com. Use the Knowledge Base to search for documented solutions to known technical issues about Informatica products. You can also find answers to frequently asked questions, technical white papers, and technical tips.

Informatica Global Customer Support

There are many ways to access Informatica Global Customer Support. You can contact a Customer Support Center through telephone, email, or the WebSupport Service.

Use the following email addresses to contact Informatica Global Customer Support:
- support@informatica.com for technical inquiries
- support_admin@informatica.com for general customer service requests

WebSupport requires a user name and password. You can request a user name and password at http://my.informatica.com.

Use the following telephone numbers to contact Informatica Global Customer Support:

<table>
<thead>
<tr>
<th>North America / South America</th>
<th>Europe / Middle East / Africa</th>
<th>Asia / Australia</th>
</tr>
</thead>
<tbody>
<tr>
<td>Informatica Corporation</td>
<td>Informatica Software Ltd.</td>
<td>Informatica Business Solutions Pvt. Ltd.</td>
</tr>
<tr>
<td>Headquarters</td>
<td>6 Waltham Park</td>
<td>Diamond District</td>
</tr>
<tr>
<td>100 Cardinal Way</td>
<td>Waltham Road, White Waltham</td>
<td>Tower B, 3rd Floor</td>
</tr>
<tr>
<td>Redwood City, California</td>
<td>Maidenhead, Berkshire</td>
<td>150 Airport Road</td>
</tr>
<tr>
<td>94063 United States</td>
<td>SL6 3TN</td>
<td>Bangalore 560 008</td>
</tr>
<tr>
<td>Toll Free</td>
<td>Toll Free</td>
<td>India</td>
</tr>
<tr>
<td>+1 877 463 2435</td>
<td>00 800 4632 4357</td>
<td>Australia: 1 800 151 830</td>
</tr>
<tr>
<td>Standard Rate</td>
<td>Standard Rate</td>
<td>Singapore: 001 800 4632 4357</td>
</tr>
<tr>
<td>Brazil: +55 11 3523 7761</td>
<td>Belgium: +32 15 281 702</td>
<td>India: +91 80 4112 5738</td>
</tr>
<tr>
<td>Mexico: +52 55 1168 9763</td>
<td>France: +33 1 41 38 92 26</td>
<td>United States: +1 650 385 5800</td>
</tr>
<tr>
<td>United States</td>
<td>Germany: +49 1805 702 702</td>
<td>United Kingdom: +44 1628 511 445</td>
</tr>
<tr>
<td></td>
<td>Netherlands: +31 306 022 797</td>
<td></td>
</tr>
<tr>
<td></td>
<td>United Kingdom: +44 1628 511 445</td>
<td></td>
</tr>
</tbody>
</table>

x  Preface
Overview

Many of the world's largest enterprises have vast amounts of information locked away in mainframe and open systems. There is often a requirement to move this data into another format, typically for use in a data warehouse, data mart or decision support system. The traditional way of moving this data could require that you perform the following steps:

1. Extract data from the operational system.
2. Translate data on the source or target system.
3. Move data to the target system.
4. When the data arrives, load it into the application, typically a relational database.

PowerExchange, in conjunction with PowerCenter, eliminates these steps and does the complete process in a single-step. PowerExchange not only extracts source data, it also moves the data at high-speed and loads it into target databases. This processing can be initiated from any platform. Because it is not restricted to vendor-specific solutions like ODBC, PowerExchange does not have any pre-requisite software to access any database.

The PowerExchange Navigator is used to build and maintain PowerExchange source and target definitions. It comprises the following main components:

- **Data Capture.** Use the Data Capture folder to define and manage capture registrations and extraction maps for PowerExchange Change Data Capture (CDC).
- **Data Maps.** Use the Data Maps folder to define and manage non-relational data maps to access data sources like IMS databases, VSAM files, and flat files. You can also define DB2 data maps.
- **Personal Metadata.** Use the Personal Metadata folder to browse remote database metadata and data.
Preparing to Use the PowerExchange Navigator

The PowerExchange Navigator runs on Windows 2000 and XP. The PowerExchange Listener is installed as a Windows Service and communicates status information through the System Tray, with start and stop PowerExchange Listener applications being provided as standard. The latest Microsoft Controls are used, so the key Microsoft DLLs that have been used to develop PowerExchange are shipped along with the Informatica developed binaries.

Installing PowerExchange on Windows

Before using the PowerExchange Navigator, complete the following prerequisites:

♦ Install the Windows version of PowerExchange. This action installs PowerExchange Navigator, a crucial component in the sourcing of data, irrespective of where the source or target databases are located.
♦ Restart the computer.

Additionally, you must have a basic understanding of Listeners and issues like COBOL copybooks and their usage.

Configuring Windows ODBC Data Sources

On Windows, configure ODBC data sources to access local or remote sources.

To define an ODBC data source:

1. Click Start > Settings > Control Dialog Box > ODBC Data Sources > System DSN > Add.
2. On the Create new Data Source dialog box, select Informatica PowerExchange, and then click Finish.
3. On the PowerExchange Data Source dialog box, enter the following information.

<table>
<thead>
<tr>
<th>Component</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Name by which the driver is referenced</td>
</tr>
<tr>
<td>Location</td>
<td>Name of a NODE entry in the PowerExchange configuration (dbmover.cfg) file, pointing to the local or remote platform</td>
</tr>
<tr>
<td>Type</td>
<td>Select the database type, such as CAPX, CAPXRT, DB2, DB2390IMG, DB2400C, DB2UDB, INFORMIX, MSSQL, NRDB, NRDB2, ORACLE or SYBASE, NRDB/NRDB2 for all flat files, IMS, and VSAM.NRDB2 for tools that only support a two-tier table naming convention.</td>
</tr>
<tr>
<td>Local Codepage</td>
<td>Contains the available code pages. Default is Default. At run time, the value from the PowerExchange configuration file or the default for the platform is used.</td>
</tr>
</tbody>
</table>

Depending on the database type you selected you will need to enter different information. Click Help to get context-sensitive help information.

4. Complete as required by your site and click OK.

The ODBC source is now fully defined.

Starting the PowerExchange Listener on Windows

Use options in the Informatica PowerExchange Program Group to start and stop the PowerExchange Listener service. When active, the PowerExchange Listener will put an icon in the System Tray. If this fails to appear there is a problem and the PowerExchange detail.log file will need to be inspected. This is updated in the current working directory when PowerExchange is running. The service is considered to start in the system directory so it is likely to be found in the C:\winnt\system32 directory or the equivalent name on your system.
To stop the PowerExchange Listener service, select Stop PowerExchange Listener from the Informatica PowerExchange Program Group. Services can also be started and stopped from the Services dialog box.

To start the PowerExchange Listener manually from a command prompt dialog box, issue the command:

dtllst node1

As it starts it will display messages to the console showing which port is being polled and after starting completely it will put out the PWX-00607 message. It is now ready for communication. Enter c, to close it from the keyboard.

Note: You do not need to run the PowerExchange Listener on Windows systems to define data maps and capture registrations. You only need to run the PowerExchange Listener on the Windows system if you are extracting data from this machine using a PowerExchange Listener on another system.

Testing the Windows Listener

Go to the Program Group typically called PowerExchange and there will be a BAT file icon for DTLPING. This will run a PowerExchange Client ping, which is a test that the connection to the PowerExchange Listener is alive. The loc=node1 is set in the shipped PowerExchange configuration file, dbmover.cfg, to use the TCP/IP Loopback IP address of 127.0.0.1. Contact should be made with the PowerExchange Listener and the request should complete with a zero return code.

If it does not, research the reason for failure. The error messages output by PowerExchange contain a lot of information. The most likely causes of failure are the IP address and port number on the NODE and PowerExchange Listener for node1 in the PowerExchange configuration file, dbmover.cfg, being incorrect. Other program loading issues are likely to be set-up errors. The steps taken during the install should be revisited to ensure that they all completed successfully with zero return codes.

Testing a Remote Listener

If you are going to use a remote PowerExchange Listener, at some stage you are going to need to start it. After you start it, check whether it is started, and whether connectivity is established by entering the following command:

dtelrexe loc=xxxx prog=ping

where xxxx is the NODE name in the PowerExchange configuration file, dbmover.cfg, pointing to the remote platform.

Preparing to Access Sources and Targets

To use PowerExchange to source data, you need to know specific information about the databases, tables, and files. The purpose of the following roadmap is to take someone with PowerExchange installed on one or more platforms and show them how to tackle various problems related to accessing supported data formats on the local platform or on a remote one.

Sourcing Relational Data

♦ Obtain the valid user ID and password for the source database.
♦ Obtain any database specific information, such as:
  – Oracle SID
  – DB2 for z/OS subsystem ID or group name
  – DB2 for Linux, UNIX, and Windows database name and instance name
  – DB2 for i5/OS database name
Sybase and Microsoft SQL Server server and database names

- Create any database-specific connectivity, such as Oracle TNS Names.
- Ping the IP address of the database source machine to check it is visible.
- Add a NODE line pointing to the database machine in the PowerExchange configuration file.
- To test that the remote PowerExchange Listener is started, use the command:
  ```
dtlrexe loc=xxxxx prog=ping uid=userid pwd=pwd
  ```
  Where xxxx is the NODE name from the PowerExchange configuration file.
- Use the PowerExchange Navigator to create a Personal Metadata profile and view some real data, having checked that the metadata is returned successfully.
- Use PowerCenter or an OEM tool to execute the request to move data.

Sourcing MVS VSAM Data Sets and Flat Files from any Platform

- If the file is not a comma or delimited text file, find the file layout, such as a COBOL or PL/1 copybook.
- Find the exact file name on the remote platform.
- Ping the IP address of the source machine to check it is visible.
- Add a NODE line pointing to the remote platform in the PowerExchange configuration file.
- To test that the remote PowerExchange Listener is started, use the command:
  ```
dtlrexe loc=xxxxx prog=ping uid=userid pwd=pwd
  ```
  Where xxxx is the NODE name from the PowerExchange configuration file.
- Use the PowerExchange Navigator to create a data map.
- Import a COBOL or PL/1 copybook to define the file layout or manually create it.
- Do a row test to return some valid data and run this process as many times as necessary, making changes to the data map until the data returned is as required.
- Use PowerCenter or an OEM tool to execute the request to move data.

Sourcing Data from IMS

- Ping the IP address of the source machine to check it is visible.

If sourcing data using DL/I Batch or BMP:

- Update the MVS PowerExchange configuration file to add a second PowerExchange Listener entry and a NETPORT line with the correct PSB name.
  
  **Note:** PowerExchange does not support PSBs coded with LANG=PL/I.

  Stop and restart MVS PowerExchange Listener.
- Update the IMSJCL member in the MVS PowerExchange RUNLIB PDS. The name of this file is also defined in the NETPORT line updated above.
- Add a NODE line pointing to the MVS NETPORT PowerExchange Listener entry in the local Windows PowerExchange configuration file.

If sourcing data using IMS ODBA:

- Ensure RESLIB data set is allocated to the PowerExchange Listener, such as STEPLIB or MVS LNKLST concatenation.
- Use the PSB which has PCBNAMES defined.
- To test that the remote PowerExchange Listener is started, use the command:
  ```
dtlrexe loc=xxxxx prog=ping uid=userid pwd=pwd
  ```
  Where xxxx is the NODE name from the PowerExchange configuration file.
Use the PowerExchange Navigator to create a data map. You need to know the IMS SSID, PSB, and PCB names.

If a DBD is available, import it to define the segments and the hierarchical sequence.

If there was no DBD to be imported, or there are COBOL copybooks defining the segment layouts, import them, possibly deleting any DBD generated ones.

Do a row test to return some valid data and run this process as many times as necessary, making changes to the data map until the data returned is as required.

Use PowerCenter or an OEM tool to execute the request to move data.

**Sourcing Data from Adabas**

- Ping the IP address of the source machine to check it is visible.
- Add a NODE line pointing to the remote platform in the PowerExchange configuration file.
- Use the following command:
  
  ```
  dtlrexe loc=xxxx prog=ping uid=userid pwd=pwd
  ```

  This command tests that the remote PowerExchange Listener is started, where xxxx is the NODE name from the PowerExchange configuration file.

- Use the PowerExchange Navigator to create a data map. Select Adabas as the access method in the map wizard and specify the Adabas database ID and file number.
- Decide which Adabas data definition to import. The PowerExchange Navigator can import from a variety of sources:
  - DDM
  - DDM OPEN SYS (Open System)
  - PREDICT
  - FDT
  - ADACMP
  - TEXT.

  It is also possible to import COBOL and PL/1 copybooks and manually associate the PowerExchange field definitions with Adabas field definitions.

- Do a row test to return some valid data and run this process as many times as necessary, making changes to the data map until the data returned is as required.
- Use PowerCenter or an OEM tool to execute the request to move data.

**Sourcing Data from IDMS**

- Ping the IP address of the source machine to check it is visible.
- Ensure DBMOVER configuration member on MVS points to the correct library containing the IDMS metadata JCL IDMSMJCL (LOADJOBFILE parameter).
- Customize the IDMSMJCL member for your installation.
- Customize the MVS PowerExchange Listener procedure/job for your installation. Ensure it is updated to allow IDMS cv or local access.
- Start the MVS PowerExchange Listener.
- Add a NODE line pointing to the remote platform in the PowerExchange configuration file.
- To test that the remote PowerExchange Listener is started, use the command:
  
  ```
  dtlrexe loc=xxxx prog=ping uid=userid pwd=pwd
  ```

  Where xxxx is the NODE name from the PowerExchange configuration file.
Use the PowerExchange Navigator to create a data map. Select IDMS as the access method in the map wizard and specify the Sub Schema and Dictionary names.

Perform a remote import. It is also possible to manually create PowerExchange record definitions or import COBOL and PL/1 copybooks. They can then be manually associated with IDMS record, area and set definitions.

Do a row test to return some valid data and run this process as many times as necessary, making changes to the data map until the data returned is as required.

Use PowerCenter or an OEM tool to execute the request to move data.

**Sourcing Data from Datacom**

- Ping the IP address of the source machine to check it is visible.
- Customize the MVS PowerExchange Listener procedure/job for your installation.
- Start the MVS PowerExchange Listener.
- Add a NODE line pointing to the remote platform in the PowerExchange configuration file.
- To test that the remote PowerExchange Listener is started, use the command:
  
  ```
  dtlrexe loc=xxxx prog=ping uid=userid pwd=pwd
  
  ```

  Where `xxxx` is the NODE name from the PowerExchange configuration file.

- Use the PowerExchange Navigator to create a data map. Select Datacom as the access method in the map wizard and specify the Database name and ID.

- Perform a remote import. It is also possible to manually create PowerExchange record definitions or import COBOL and PL/1 copybooks.

- Do a row test to return some valid data and run this process as many times as necessary, making changes to the data map until the data returned is as required.

- Use PowerCenter or an OEM tool to execute the request to move data.

**Targeting MVS VSAM Data Sets or Flat Files on any Platform**

- Find the layout of the file to be created, such as a COBOL or PL/1 copybook.
- Decide on the exact file name on the remote platform.
- If it is an MVS system, preallocate the file.
- Ping the IP address of the target machine to check it is visible.
- Add a NODE line pointing to the remote platform in the PowerExchange configuration file.
- To test that the remote PowerExchange Listener is started, use the command:
  
  ```
  dtlrexe loc=xxxx prog=ping uid=userid pwd=pwd
  
  ```

  Where `xxxx` is the NODE name from the PowerExchange configuration file.

- Use the PowerExchange Navigator to create a data map.

- Import a COBOL copybook to define the file layout or manually create it.

- Do a row test even though it is likely that no data will be present. This is needed to send the data map to the remote platform.

- Use PowerCenter or an OEM tool to execute the request to move data.

**Targeting Adabas**

- Determine the Adabas database ID and file number.
- Ping the IP address of the source machine to check it is visible.
- Add a NODE line pointing to the remote platform in the PowerExchange configuration file.
To test that the remote PowerExchange Listener is started, use the command:

dtlrexe loc=xxxx prog=ping uid=userid pwd=pwd

Where xxxx is the NODE name from the PowerExchange configuration file.

Use the PowerExchange Navigator to create a data map. Select Adabas as the access method in the map wizard and specify the Adabas database ID and file number.

Decide which Adabas data definition to import. The PowerExchange Navigator can import from a variety of sources:

- DDM
- DDM OPEN SYS (Open System)
- PREDICT
- FDT
- ADACMP
- TEXT

It is also possible to import COBOL and PL/1 copybooks and manually associate the PowerExchange field definitions with Adabas field definitions.

Do a row test to return some valid data and run this process as many times as necessary, making changes to the data map until the data returned is as required.

Use PowerCenter or an OEM tool to execute the request to move data.

**Targeting IMS**

- Find the layout of the file to be created.
- Decide on the exact file name on the remote platform.
- Ping the IP address of the target machine to check it is visible.
- Add a NODE line pointing to the remote platform in the PowerExchange configuration file.
- To test that the remote PowerExchange Listener is started, use the command:

  dtlrexe loc=xxxx prog=ping uid=userid pwd=pwd

  Where xxxx is the NODE name from the PowerExchange configuration file.

- Use the PowerExchange Navigator to create a data map.
- Manually define the file layout or import a copybook.
- Do a row test even though it is likely that no data will be present. This is needed to send the data map to the remote platform.
- Use PowerCenter or an OEM tool to execute the request to move data.

**Targeting C-ISAM Data**

- Find the layout of the file to be created.
- Decide on the exact file name on the remote platform.
- Ping the IP address of the target machine to check it is visible.
- Add a NODE line pointing to the remote platform in the PowerExchange configuration file.
- To test that the remote PowerExchange Listener is started, use the command:

  dtlrexe loc=xxxx prog=ping uid=userid pwd=pwd

  Where xxxx is the NODE name from the PowerExchange configuration file.

- Use the PowerExchange Navigator to create a data map.
- Manually define the file layout or import a copybook.
Do a row test even though it is likely that no data will be present. This is needed to send the data map to the remote platform.

Use PowerCenter or an OEM tool to execute the request to move data.

Targeting Relational Data

- Obtain the valid user ID and password for target database.
- Obtain any database specific information, such as Oracle SID, DB2 SSID, and Sybase Server.
- Create any database specific connectivity, such as Oracle TNS Names.
- Ping the IP address of the target machine to check it is visible.
- Add a NODE line pointing to the remote platform in the PowerExchange configuration file.
- To test that the remote PowerExchange Listener is started, use the command:

  dtlrexe loc=xxxx prog=ping uid=userid pwd=pwd

  Where xxxx is the NODE name from the PowerExchange configuration file.

- Use the PowerExchange Navigator to create a Personal Metadata profile and view some real data, having checked that the metadata is returned successfully.

- Use PowerCenter or an OEM tool to execute the request to move data, setting the table creation parameters as required.

Defining the Resource Configuration

The Resource Configuration dialog box specifies the folders and paths used for keeping Resources.

Resources comprise Data Maps and personal metadata. Data Maps are shareable, while Personal Metadata is not. This means that personal metadata is always held under the Local Path, while Data Maps can be stored under either the Local Path or the Shared Path.

Current Configuration Tab

Access the Resource Configuration dialog box through the menu option Options, Resource Configuration from the main dialog box. The Resource Configuration dialog box shows the paths that PowerExchange Navigator uses for its resources.

Edit Configuration Tab

You can change the resource path through the Edit Configuration tab.

It is possible to add, edit or delete configurations using the appropriate buttons. To edit or delete the appropriate configuration must be highlighted.

Search for the Local Path and Shared Path by clicking the Browse button.

Shared Resource

To make use of the Shared Resource Configuration facility users need to map a network drive to a local disk drive, as \ (double backslash) is an invalid string in the PowerExchange path name.

In the My Network Places dialog box, select the device to be mapped.

Use this device in the Path on the Map Network Drive dialog box. Assign a Drive to this path and click OK.

Complete the fields on the Add Configuration dialog box and check the Set as current configuration box.
Selecting Views

On the toolbar there is a Views icon. Click the black arrow immediately to its right, the standard Windows options are offered:

♦ Large Icons
♦ Small Icons
♦ List
♦ Details

By selecting one of these or clicking the Views icon, the viewing layout and style of the Resources dialog box might be altered. The default setting is Small Icons.
CHAPTER 2

Creating Data Maps

This chapter includes the following topics:
- Data Map Concepts, 11
- Creating Data Maps - Examples, 12
- Creating Adabas Data Maps, 20
- Creating Datacom Data Maps, 24
- Creating DB2 Data Maps, 26
- Creating IMS Data Maps, 37
- Creating VSAM Data Maps, 52
- Creating C-ISAM Data Maps, 59
- Creating MQSeries Data Maps, 61
- Data Map Code Pages, 63

Data Map Concepts

PowerExchange data maps consist of record and table definitions for a source or target. Data maps define the field layout of the records in the source or target. The table definition in the data map provides a relational view of the data records. PowerExchange uses the table definitions in data maps to construct SQL statements to access source and target data. In the case of non-relational sources and targets, PowerExchange interprets the SQL statements internally. For relational sources and targets, PowerExchange passes the SQL to the RDBMS for processing.

Complete the following steps to building a data map:
1. Define the layout of the actual physical data. Adding both record and field definitions does this. This is the non-relational view of the data.
2. Define tables and columns from the defined records and fields to provide the relational view of the data.
3. Send the data map to a PowerExchange Listener on the local or remote node. This action converts the data map into a platform independent file, which can be accessed by the PowerExchange extract run-time routines.

After you complete these actions, run relational type extracts using SQL against the data source or move data from a source to a non-relational target. Test a data source from the PowerExchange Navigator using the Row Test feature.
This actually sources data as one or more columns from the REAL source and displays it back in a tabular character format with column headings, so that you can see what would be retrieved by the run-time routines. If it is not what you expected, it is a simple matter of going back and changing the data map and running the Row Test again. This iterative process is very fast and productive.

Structure and Content of a Data Map

The PowerExchange Reference Manual contains a chapter about non-relational SQL syntax. For example, the following is an example of non-relational SQL syntax:

```sql
select col_a, col_b, …, col_n from schema.mapname.tablename [ where condition ]
```

The `schema.mapname` component comes from the data map naming convention. PowerExchange provides a two level qualifier system, so that you can organize your data maps in projects and groupings with logical naming conventions.

Within the data map, you can have any number of record and table definitions but the essential restriction is that only one physical file can be defined in a data map. Multiple record definitions make sense only if there are multiple record types on the file, whereas from a single record many different table definitions can be created. The fewer columns retrieved, the faster the file can be processed and data extracted.

Creating Data Maps - Examples

A Simple Example

Overview

The following example shows some basic features and points of note about data maps. As part of the install process, PowerExchange creates a sub-directory called examples under the main program directory. It contains the file `demo1.dat`. Open it with a text editor, such as Notepad. Each record in this file holds three fields. It is the basis for our first example.

If you do not have the file, create a file with the following text:

```
10, Mickey Mouse, M
20, Shirley Temple, F
30, John Wayne, M
40, Donald Duck, M
50, Greta Garbo, F
```

Step 1. Setting the PowerExchange Configuration

When you start up PowerExchange Navigator for the first time you are prompted for the location of your data maps and other resources. The directory that you choose during installation becomes the default.

**Note:** You are not prompted for this location if there are registry entries from a prior installation.

You can change the default location on the Edit Configuration tab. Additional configurations, such as separating test and production environments, can also be defined in this manner. Sub-directories are created below this one, where the individual members are held.

Step 2. Main Application Window

The main application window is now displayed.

The standard installation includes various demonstration and example members and these are listed in the resources view and in the left hand pane the categories would show the + symbol, indicating that there are specific members to be displayed.
There are several ways to create a data map. For example:

1. Use the toolbar **Add Data Map** icon.
2. Click **Add Data Map**.
3. Right-click **Data Maps** and click **Add Data Map**, as shown in the following figure:

*Note:* The Resources area must be highlighted for the Toolbar and Menu option methods to be active.

**Step 3. Data Map Name**

1. After you select **Add Data Map**, the main data map Name dialog box appears.
2. Enter a Schema Name, such as demo, and a data map name, such as map1.
3. The Access Method is important. In this example SEQ has been selected as data is being sourced from a flat or text file. Choose one of the following values for Access Method:

<table>
<thead>
<tr>
<th>Access Method Value</th>
<th>Data Source Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADABAS</td>
<td>Adabas files</td>
</tr>
<tr>
<td>DATACOM</td>
<td>CA-Datacom databases and tables</td>
</tr>
<tr>
<td>DB2</td>
<td>DB2 for i5/OS, DB2 for Linux, UNIX, and Windows, and DB2 for z/OS tables</td>
</tr>
<tr>
<td>DB2UNLD</td>
<td>Unload data sets for DB2 for z/OS tables</td>
</tr>
<tr>
<td>DL/1 Batch</td>
<td>IMS databases, accessed using BMP and DL/I batch jobs</td>
</tr>
<tr>
<td>ESDS</td>
<td>VSAM ESDS data sets</td>
</tr>
<tr>
<td>IDMS</td>
<td>CA-IDMS databases</td>
</tr>
<tr>
<td>IDMSX</td>
<td>no longer used</td>
</tr>
<tr>
<td>IMS ODBA</td>
<td>IMS databases, accessed using the ODBA interface</td>
</tr>
<tr>
<td>ISAM</td>
<td>C-ISAM files</td>
</tr>
<tr>
<td>KSDS</td>
<td>VSAM KSDS data sets</td>
</tr>
<tr>
<td>MQSERIES</td>
<td>Websphere MQ queues</td>
</tr>
<tr>
<td>RRDS</td>
<td>VSAM RRDS data sets</td>
</tr>
<tr>
<td>SEQ</td>
<td>Sequential data sets and flat files</td>
</tr>
<tr>
<td>TAPE</td>
<td>Tape data sets on MVS</td>
</tr>
<tr>
<td>TurboIMAGE</td>
<td>no longer used</td>
</tr>
<tr>
<td>USER</td>
<td>User files, accessed using a user access method module instead of the native PowerExchange access method</td>
</tr>
</tbody>
</table>

4. The Import Record Definitions option should be disabled, as the record definitions will be specified later.
5. Click **Next**.

**Step 4. Access Method**

The dialog box now displayed will vary according to the access method that was previously selected. The following dialog box shows the appropriate one for the SEQ access method.
For sequential files you will need to provide the full path name for the PowerExchange Listener. This must be relative to where the PowerExchange Listener will be run from. If in doubt, enter the full path and file name. If you are sourcing a file from the local computer or network click the Browse button on the right side of the File Name to locate the file.

1. Click demo1.dat, for example, and then click **Open**.

   The data file, demo1.dat, is sourced from is a character-separated file. The PowerExchange Navigator does not prompt for the separator character.

2. Select **Field Separator** and type in the separator character, which is a comma (,).

   **Note:** If you cannot enter the separator character through the keyboard, enter its hexadecimal representation as: x'nn'.

3. To complete the basic set-up of the new data map, click **Finish**.

   The Data Map tab appears. Also, the name of any data map open is shown in the Resources dialog box.

4. Click the **Resources** tab to return to the main view.

5. For this example leave offload processing as default. If the PowerExchange Listener is running on a CPU constrained host the process of converting and merging data records can be offloaded from the PowerExchange Listener to the client. For certain maps (especially those containing variable fields) the savings can be upwards of 90% with only a small increase in network traffic.

<table>
<thead>
<tr>
<th>Yes</th>
<th>Always offload processing where possible.</th>
</tr>
</thead>
<tbody>
<tr>
<td>No</td>
<td>Never offload processing.</td>
</tr>
<tr>
<td>Automatic</td>
<td>Allows PowerExchange to optimize the location. Offload processing will be used if the source contains variable length fields.</td>
</tr>
<tr>
<td>Default</td>
<td>Refer to the OFFLOADPROCESSING parameter in the dbmover.cfg file.</td>
</tr>
</tbody>
</table>

   Message "PWX-00672 Record Mapping moved to Client for <<xxxx source>>" appears in the log if data map processing has been off-loaded where xxxx describes the data source.

   Offload processing does not occur if the data map is being used for parameterized SQL, group source, or if Data Checking has been selected for the data map.

**Step 5. Adding a Record**

**To add a record:**

1. Right-click demo.map1 and click **Add Record**.

   A blank Add Record dialog box appears.

2. Enter a record name; this can be anything you like as it is used for reference when building tables. For example, record1.

3. Click **OK**.

**Step 6. Adding a Field**

**To add a field:**

1. Add each field by using the toolbar Add Field icon by clicking **Add > Field**.

   Add the following fields to the sample data file: seqno, name, and gender.

2. Complete the following tasks for each field:
   - Enter a **Field Name**
   - Select a **Field Type**.
The details for each field that is to be added are as follows:

**Add the seqno field, which is a number field:**
1. From the Field Type list, select NUMCHAR (numeric character).
2. From the Precision list, select 2.
3. From the Scale list, select 0.
4. From the Length list, select 2. The Length value must be equal or greater than the Precision. This field indicates the maximum size of the input, allowing for leading blanks or other characters.
5. Click OK.
   The field is added to the record display.

**Add the name field (which is a character field):**
1. From the Field Type list, select CHAR (character).
2. From the Length list, select 20.
3. Click OK.

**Add the gender field (which is a character field):**
1. From the Field Type list, select CHAR (character).
2. From the Length list, select 1.
3. Click OK.

**Step 7. Adding a Table**

To add a table:
1. Right-click the data map name, and click Add Table.
   This action displays the Add Table dialog box.
   In this example all the data from an input record is going to make up a row in the new table. The Filter tab allows overrides of defaults. This is not required in this example.
2. Click OK. The table is created using all the fields in record1 and displayed in the Table display window.
   This data map is now complete. It can be used to test our data source immediately. To keep it simple we are not going to connect to a remote PowerExchange Listener, but go directly to the data file locally.

**Step 8. Row Test**

1. Use the toolbar Row Test icon or click **File > Database Row Test** to start the process.
   A message box appears asking you to confirm that you want to send the data map to a remote location.
2. Click Yes.
3. Set the Location to local.
4. Click OK.
5. On the main Database Row Test dialog box, click Go.
   Go might not be visible with a low screen resolution value. Increase this value or drag the Database Row Test window to the left.
   If the filename is not correct or the file is not in the correct place, an error message dialog box might appear, which indicates that the data file that you selected was not found.
6. To correct a missing data file, close the Database Row Test dialog box, right-click Data Map and click Properties. Correct either the file name or put the file in the place where the name is pointing.

In this example, all the table columns were used. It is possible to specify the columns to be included by selecting the columns in the Table display window.

7. Save the data map. Close the Database Row Test dialog box and select File > Close Resource.

You now see demo.map1 under the Data Maps resource folder.

A Second Example

Overview

The following example illustrates some more advanced features regarding data maps. This includes features such as Array Generation and how to distinguish between multiple record types within a file. The sample data file for this example is similar to the one used above, but it has been extended to show the OCCURS (array) capabilities of PowerExchange. The files used in this example are included in the installation:

Data file, demo2.dat:

10, Mickey Mouse, M, 3, apple, orange, pear
20, Shirley Temple, F, 1, raspberry
20, John Wayne, M, 2, pansy, daisy
10, Donald Duck, M, 0
30, Goofy, M, 1, fox
10, Greta Garbo, F, 3, dog, cat, rabbit
20, Ronald Reagan, M, 2, horse, pony
20, Bette Midler, F, 1, wolf

Data map, demo.map2

Step 1. Open Data Map and select Data File

1. Double-click demo.map2 in the Resources area.
2. Right-click demo.map2 and click Properties.
3. Click the SEQ Access Method tab.
4. Click the Browse button and select demo2.dat from the following directory:
   Program Files\Informatica\PowerExchange\examples
5. Click OK to return to the Data Map area.

Step 2. Exclude a Data Record

1. Click record1 to show its contents: A single record and fields that have been defined.
2. Right-click the rectype field and click Properties.

   The two values displayed in the Record ID Values show how to distinguish between multiple record types. As the = operator is used, in this instance the field RECTYPE is limited to those records containing either ‘10’ or ‘20.’ In the sample data the Goofy record with a value of 30 will not be selected. If the <> operator is used, then the field is limited to those records not containing ‘10’ or ‘20’ and only Goofy would be selected.

   Note: The record ID is taken from the map and not the data. This is because the record id fields may not actually be exposed in the columnar view of the data. This means that record id filtering cannot be used on WRITE.

3. To close Field Properties dialog box, click OK.
Step 3. Define Tables and Columns

1. Right-click the row_out table and click Properties.

   In this example there is a variable array. With the Group and Arrays only checked only those fields defined as a group field or an array should be displayed; in this case, one field, items.

   As the items checkbox is checked, each occurrence of the data will be treated as one row and the table is built as follows.

   - The color of the items_L column, showing that it is part of an array specification.
   - At run-time the data will generate multiple output rows from a single input record, depending on the number of elements in the array.

2. Return to the Table Properties dialog box. The two boxes are still checked.

Step 4. One Row per Record

1. Clear the Groups and Arrays Only box and note the changes to the screen display as shown below. All fields are now displayed.

2. If you clear the items box in the Fields section of the Table Properties dialog box, a different table is generated as follows.

   - The change is caused by un-checking the items box. Clearing the Groups and Arrays Only box varies the fields listing displayed in the Table Properties, not the final output.
   - At run-time the data will be retrieved as follows.

   - There are now only seven rows, yet three separate item fields.
   - How many rows are to be output from multiple input records or segments is controlled by the way you answer the question How do you want to handle multiple instances of selected record?
Step 5. Multiple Rows per Record

- Go to the Table Properties dialog box:
  - Recheck the items box.
  - Change Ignore to Array.
  - Set the Array value to 2.
  - Check the New Row on Overflow box.

The table is displayed as follows.

At run time, the data is retrieved as follows.

A Third Example

Overview

This example has a data map and data file supplied, these are, respectively, demo.map3 and demo3.dat. This example demonstrates how:

- A data map table can have multiple data map records associated with it.
- To specify the relationship between multiple data map records (in this instance parent/child).

Two data record types will be associated with their own data map record (rec1 and rec2) based on the numeric value in column one of the data file. This is different to the last example where data records with values of '10' and '20' were assigned a single data map record - rectype. Here is the file:

```
1,Mickey Mouse2,Disneyland2,Mouse Hole1,Shirley Temple2,Hollywood1,John Wayne1,Donald Duck2,Disneyworld1,Ronald Reagan2,White House2,Hollywood
```

The file consists of two record types, 1 and 2. Here are the record definitions.
1. Right-click the rectype fields and click Properties. This displays the appropriate Table Properties dialog box for the selected record, for example, rec1.

   **Note:** Because PowerExchange has been told of the two record types and how to identify each, based on the Record ID Values, the records can now have their relationship defined.

   When you go to the Add Table dialog box, you have to make some decisions.

   On a single record data map, the record was automatically defaulted into the Record Dependencies area. In this instance you will need to select which records (or segments for DL/1) will make up the table.

2. Right-click rec1 in the Available Records list and click Add Record. This will move rec1 to the Record Dependencies list. After an initial record or segment is selected, select other segments as children to create a hierarchy. If rec1 is selected as the main record, rec2 can be made a child.

3. Enter row_out as the valid Table Name and then click OK. The following table is generated.

   **Note:** The source field name for each table column is prefixed with the source record name so that each has a unique reference (in this case the rectype fields each have a unique reference).

4. To remove a column, select it and click the Delete icon. (PowerExchange will ask you to confirm that you wish to delete the column). Delete both rectype fields and the table will now look as follows.

   The data retrieved from the file using the row test feature is as follows.

   **Note:** The data from rec1, column name, has been carried through for each address associated with it.
Creating Adabas Data Maps

The first step to accessing Adabas through PowerExchange is to create a data map with PowerExchange Navigator. Adabas provides several data definitions that you use to create data maps:

- **DDM and PREDICT.** DDM and Predict are the best options as they provide long name and scale information.
  
  To import these types it is necessary to know the FDIC database ID and file number. Given this information it is possible to browse for the required data structure (wildcards are supported) or specify the name directly if known.
  
  **Note:** If you import DDM metadata for Adabas databases on MVS containing wide character datatypes, PowerExchange does not automatically select the Wide Char option on the Code Page tab for these fields. You must either import Predict or FDT metadata or manually select the Wide Char option in the field properties for all wide character datatypes.

- **FDT.** The Field Definition Table (FDT) is an Adabas construct containing the database definition. It also enables validation of Adabas related information such as field names, array sizes, and override lengths. Importing the Field Definition Table (FDT) ensures that the latest definition is selected but provides only short names and no scale information. Once imported the FDT is also cached for subsequent sessions. It can be refreshed through the FDT options. The caching of FDT allows certain validity checks to be made prior to run time.

- **DDM OPEN SYS.** DDM OPEN SYS (or Open System) is used to support Adabas databases on Windows and UNIX.

- **ADACMP.** ADACMP cards can be imported directly from the host as is done for the other file types or from a local file on the machine.

- **TEXT.** TEXT is a delimited format file which uses an Informatica internal format.

  **Note:** You can also import COBOL and PL/1 copybooks. As these do not contain any Adabas field information the resulting PowerExchange field definitions must be associated with Adabas field definitions through the field properties dialog box.

You can subdivide, or redefine, any PowerExchange field mapped to an Adabas field. To subdivide a field, defining it as a group field with subsidiary fields that re-map the group.

**To create an Adabas data map:**

1. Choose **Add > Data Map.**
   -or-
   Use the **Add Data Map** icon on the toolbar.
   -or-
   Right-click **Data Maps**, and then click **Add Data Map.**

   The Name dialog box displays.

2. Select **ADABAS** in **Access Method**, and enter values for **Schema Name** and **Data Map Name**.

   By default, the PowerExchange Navigator selects the **Import Record Definitions** and **Import Key Fields/FDT** options.

3. Click **Next.**

   The **ADABAS Access Method** dialog box displays, which contains the following fields:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Database ID</td>
<td>Specifies the Adabas database number</td>
</tr>
<tr>
<td>File Number</td>
<td>Specifies the file number in the Adabas database</td>
</tr>
</tbody>
</table>
The Database ID and File Number normally equate to the Database ID and File Number of the expected source of the Adabas data. PowerExchange provides the flexibility at this point to set the Database ID to zero.

If the actual Database ID is zero, then PowerExchange will access the data correctly. Setting Database ID to zero enables the database to be overridden at runtime using one of the following methods:

- ADABAS_DEFAULT_DBID statement in the DBMOVER configuration file
- Database Id Override option in the session properties in PowerCenter
- Override File Name in the ODBC escape sequence DTLDSN or ODBC parameter DBQUAL1.

Tip: Setting the Database ID to zero and overriding the value in PowerCenter makes migrating data maps from one Adabas environment to another easier.

4. Click Finish.

The Adabas FDT Import dialog box appears. Enter the node name specified in dbmover.cfg on Windows for the Adabas source system. If the source system is MVS and PowerExchange is configured for security, enter a valid MVS user ID and password in the User ID and Password fields. If you use Adabas file security, enter the password in the File Password field.

5. Click OK.

The Import Copybook - Source Details dialog box displays. This dialog box has the following fields:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source</td>
<td>Specifies the location of the copybook or metadata information. Select Local if the metadata is available on this Windows machine or Remote if the metadata is remote from this Windows machine.</td>
</tr>
</tbody>
</table>
6. Click **Next**.

The Import Copybook - Remote Details dialog box displays, which has the following fields:

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>File Password</strong></td>
<td>Adabas file password, if required.</td>
</tr>
<tr>
<td><strong>Location</strong></td>
<td>Target location as defined in the dbmover.cfg file.</td>
</tr>
<tr>
<td><strong>UserId/Password</strong></td>
<td>If PowerExchange is configured for security, a valid MVS user ID and password.</td>
</tr>
<tr>
<td><strong>Save File Locally As:</strong></td>
<td>File into which the copybook or metadata will be saved locally.</td>
</tr>
<tr>
<td><strong>Name</strong></td>
<td>Enter Adabas name or select from the Browse List.</td>
</tr>
<tr>
<td><strong>Name Browse</strong></td>
<td>To generate a selection list leave the Name blank and click Name Browse. Click the selected name from the list which will then appear in the Name field.</td>
</tr>
<tr>
<td><strong>Preview</strong></td>
<td>Click this button to see the Adabas layout of the selected Name.</td>
</tr>
</tbody>
</table>

If you select **DDM**, **DDM OPEN SYS**, or **PREDICT**, enter a name in the **Name** field. The **Name Browse** button provides the opportunity to browse for the required file and the **Preview** button allows you to view it.

7. Click **Next**. The Import Copybook - Configuration Details dialog box displays. Only options related to the copybook type are enabled. Configure the options you want to use and click **Finish**.

8. The Record Definition dialog box displays with the Adabas record name in the **Name** field.

9. Click **OK**. The data map definition is now complete.

10. Close and save the new data map.

**FDT Import**

When you open an Adabas data map that has not had the **FDT** imported, a message box is displayed asking you if you wish to import the **FDT** details now. If you click **No**, the data map is displayed. If you want to import the **FDT** later, click **Resource > FDT**. Alternatively, right-click the data map, and then click **FDT**.
To import the Adabas database FDT:

1. When you open the data map, click Yes to the prompt to import the FDT.

2. The Adabas FDT Import dialog box displays. Enter the Location of the FDT information. If required, enter values for User ID, Password, and File Password.

3. Click OK. The Adabas FDT dialog box displays showing the FDT information.
   
   Click Refresh FDT to refresh the FDT information.

4. Click OK.

Importing Adabas Keys

You can import Adabas keys when importing FDT metadata.

To import Adabas keys for FDT data maps:

1. Choose Add > Data Map.

   -or-

   Use the Add Data Map icon on the toolbar.

   -or-

   Right-click Data Maps and click Add Data Map.

   The Resources Navigator must be active to use the toolbar or menu options.

2. Enter values for Schema Name and Data Map Name.

3. Select Import Record Definitions and Import Key Fields/FDT. This populates a list on the Keys tab of the Data Map Properties dialog box.

4. For each mapped table you can define a key. Highlight the table and select the Properties right-menu option.

5. Click the Keys tab.

6. The Primary Key list holds valid keys for the table that has been defined. The validity of a key is determined by checking that the table contains all the fields that are defined in the record as having the Adabas field names needed by the descriptor. If any of these fields have been mapped into groups, then all members of the group must also be present. Fields in vertical arrays are not valid.

   The Key Details list shows the key that has been selected for the table from the Primary Key list.

Testing Adabas Data Maps

To test the definition, use Database Row Test. The database row test options retrieves data from the selected Adabas database. For more information about Database Row Test, see “Using Database Row Test” on page 107.
Creating Datacom Data Maps

The first step to accessing Datacom through PowerExchange is to create a data map with the PowerExchange Navigator.

Data maps create a relational view of the non-relational data. These definitions are then used to access the non-relational data.

The recommended method of obtaining the Datacom definition is for the Datacom copybook to be imported automatically, as described in the following procedure.

There is also a manual method that allows the use of COBOL and PL/1. For more information about importing a COBOL or PL/I copybook, see “Importing Copybooks” on page 65.

To create a Datacom data map:

1. In the PowerExchange Navigator, right-click the Data Maps folder and select Add Data Map. Alternatively, select Add > Data Map from the toolbar.
2. Select DATACOM in the Access Method.
3. Enter values for Schema Name and Data Map Name. These names do not need to relate to any Datacom structures.
4. Select the Import Record Definitions box to allow the definitions to be imported from Datacom.
5. Enter the appropriate values in Database Name and Database ID.
   
   Note: Both the Database Name and Database ID are mandatory. If the data map is being built from scratch, and if the Import Record Definition was not checked on the previous screen, then enter the appropriate value in Record Size.
6. Click Finish.
7. The Import Copybook wizard appears. Select Remote.
8. Select DATACOM, COBOL, or PL/I for the copybook source Type.
9. Click Next.
10. The Import Copybook dialog box appears. Enter the appropriate details.
    
    There are two user IDs and passwords, and these will need to be entered as required by the configuration of PowerExchange and Datacom. The need to enter the first user ID and password is dependent upon the SECURITY parameter in the DBMover configuration file. For more information about PowerExchange security, see the PowerExchange Reference Manual.
    
    The Dictionary User ID and Password are required if Datacom requires them. Enter the Datacom Table Name, Location, which is a node in the dbmover.cfg file on the PC, and a name for the generated map to be held. If overwriting an existing one, the Browse button can be used to find the required folder and file.
    
    If a Copybook is to be imported with the metadata, select Copybook and then select either COBOL or PL/I.
    
    Note: The Table Status and Key and Elements fields are disabled at this time.
11. Click Next or Finish.
    
    If you select Next, the Configuration Details dialog box displays. This dialog box is skipped if you click Finish.
    
    If the Configuration Details dialog box displays, click Finish to display the summary information, and then click OK.
12. If Record Definition dialog box displays, click Apply to import the rest of the copybook and then click OK.
13. The Datacom Import dialog box appears after all records have been imported. Close the Datacom Import dialog box.

Testing Datacom Data Maps

To test the definition, use Database Row Test. The database row test options retrieves data from the selected Datacom database. For more information about Database Row Test, see “Using Database Row Test” on page 107.

Datacom Record Properties Dialog

To view Datacom-specific record properties, right-click the record in data map and click Properties. Edit the attributes of the record definition from this screen.

Elements Tab

An element is the unit of transfer used between applications issuing Datacom commands and Datacom. It consists of one or more contiguous columns, which are FIELD entity occurrences. An element should contain only those columns that an application program uses at execution time. Each table requires one to 255 elements.

To display the element information:
> Click the Elements tab. The Record Properties dialog box is displayed.
> The Elements Record Properties dialog box contains the following fields:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Name of the record element to be used.</td>
</tr>
<tr>
<td>Position</td>
<td>Position offset from the start of the record.</td>
</tr>
<tr>
<td>Length</td>
<td>Length of the element.</td>
</tr>
</tbody>
</table>

To add an element:
1. Click the Elements tab for the record on the data map.
2. Click the Add icon.
3. Enter a name for the element in the Name field, and then enter values for the Position and Length fields.
4. Click OK. The new element details display in the Record Properties dialog box.

To delete an element:
1. Click the Element tab for the record on the data map.
2. Highlight the desired Element entry you want to delete.
3. Click the Delete icon.

Keys Tab

Keys are structures used to optimize data access or order data retrieval. A key is composed of columns that can be contiguous or non-contiguous and in any sequence. Each column in a key can be ascending or descending in value. Each key can be up to 180 characters long. Define up to 99 keys for each table or 999 keys for each database. Any key can be defined as unique, that is, requiring that each row in the table have a unique value for the key.
All tables must have a Master Key and a Native Key defined. The Master Key functions as any other key, but it can be defined as updateable or non-updateable. The Native Key dictates the physical sequence in which the data is stored. The Native Key can be the same as the Master Key.

Click the **Keys** tab. The Keys Record Properties dialog box displays with the long and short names for all of the keys.

**To add a key:**
1. Click the **Keys** tab for the record on the data map.
2. Click the **Add** icon.
3. The Key Properties dialog box displays. Enter key name values in the **Long Key Name** and **Short Key Name** fields. Select any appropriate key attributes.
4. Click the **Segment Details** tab to assign the new segment details. Enter the segment name in the **Name** field. Enter values for the **Position**, **Length**, **Order**, and **Sensitivity** fields.
5. Click **OK** to display the new structure. Then click **OK** again. The key will appear in the Record Properties dialog box.

**To delete a key:**
1. Click the **Keys** tab for the record on the data map.
2. Highlight the key entry you want to delete.
3. Click the **Delete** icon.

---

### Creating DB2 Data Maps

DB2, being a relational database, is normally accessed by PowerExchange with Personal Metadata. However, there are reasons when it would be preferable to create a data map of the DB2 data.

Some DB2 implementations use single DB2 columns to store an array of fields in a format not necessarily consistent with the column type. For example, a set number of packed data fields in a CHAR column or a varying number of packed fields in a VARCHAR column.

PowerExchange enables users to query these fields, manipulate them using expression fields and expose them to downstream processing by PowerCenter.

After DB2 data maps are available they can be extended to make it easier to map columns that have been unloaded into flat files by standard utilities.

PowerExchange provides data maps for DB2 tables and for DB2 unload files. For DB2 unload files, the following IBM and BMC Software formats are supported:

- DB2 for z/OS online REORG TABLESPACE utility with UNLOAD EXTERNAL
- DB2 for z/OS online UNLOAD utility
- DB2 for z/OS sample unload program DSNTIAUL
- BMC Software Unload Plus
- DB2 for Linux, UNIX, and Windows High Performance Unload utility

**Note:** When you create a DB2 unload data map selecting **Unload Type** of DSNUUTILB UNLOAD, PowerExchange includes a field for the OBID in the record. If you create DB2 unload files specifying **HEADER NONE** in the **DB2 UNLOAD** control statements, select **Unload Type UNDEFINED**.
Creating Data Maps Using the DB2 Catalog

The data map creation process is the same as for any other data source that you have mapped using PowerExchange. The following sections give an example of creating the data map and then testing it.

Note: Although this example uses DB2 for z/OS, the same process can be used for DB2 for i5/OS and DB2 for Linux, UNIX, and Windows.

When naming the new Data Map, ensure that the Access Method is DB2.

1. Enter a Schema Name, such as db2 and a Data Map Name, such as map1. The Import Record Definitions box must be checked.
2. Click Next. The DB2 Access Method dialog box appears.
3. Enter the required DB Instance, such as DSN1.
4. Either enter the required table name (ignore steps 5 and 6) or select it by clicking the Browse button. The DB2 Table Filter dialog box appears.
   For speed and ease of selection, restrict the number of returned tables by entering filter criteria based on Creator/Schema name, database name and/or table name.
5. Click Next. The DB2 Table Filter dialog box appears showing a list of available tables that match your criteria.
6. Select the required table and click Finish.
7. Click Finish. The Import Copybook wizard opens automatically.
8. In Source, select Remote radio. This is the node name where the member exists.
9. In Source, select DB2 Catalog from the Type pull-down box.
10. Click Next. The Import Copybook - Remote DB2 Catalog Details dialog box appears.
   The input fields are automatically filled in from previous selections but these can be changed to import details from any other source that you require.
   Select Preview/Amend Columns to display a dialog box that allows you to select those columns in which you have an interest.
   Save the copybook as a local file by entering the file name in the Save File Locally As box.
11. Click Next. The Import Copybook - Configuration Details dialog box appears requesting what actions to be taken when loading the copybook.
12. The options request prompts in various situations and actions that will be taken if duplicate record, fields and/or tables are detected.
13. Click Finish.
14. Click OK on the Import Copybook Information dialog box. The Record Definition dialog box appears.
15. Click OK for the EMPLOYEE.
The following Error/Log Message should be displayed.

16. Click EMPLOYEE record and table to view the fields and columns that have been imported.

17. Thereafter, the data can be sourced in the normal way. After the import is complete, the data map is identical to one created by any other means.

Row Testing a DB2 Data Map

1. Click EMPLOYEE table and click the Row Test icon.

2. On the Database Row Test dialog box click Go. After performing the Row Test with the results should be as shown below.

3. Close the Database Row Test dialog box.

Creating Data Maps Using DB2 Unload Files

In addition to direct data mapping of DB2 tables, you can also map DB2 unload files.

The data map creation process is the same as for any other data source that you have mapped using PowerExchange. The following sections give an example of creating the data map and then testing it.

1. When naming the new Data Map, ensure that the Access Method is DB2UNLD. Enter a Schema Name, such as db2unload, and a Data Map Name, such as map1. The Import Record Definitions box must be checked.

2. Enter the name of the target db2 unload file into the File Name box.

3. Enter the required DB Instance, such as DSN1.

4. Either enter the required table name or select it by clicking the Browse button. The DB2 Table Filter dialog box appears.

5. For speed and ease of selection, you can restrict the number of returned tables by entering filter criteria based on Creator/Schema name, database name and/or table name.

6. Click Next. The DB2 Table Filter dialog box appears showing a list of available tables that match your criteria.

7. Select the required table and click Finish.

8. Select the Unload Type that will be used. For bulk access to Unload files only the following IBM and BMC formats will be supported:

   - REORG UNLOAD EXTERNAL
   - DSNTIAUL/BMC UNLOAD+
   - DSNUTILB UNLOAD
   - UNDEFINED

   This is designed to be modified by users to their own specific requirements rather than use any of the preset unload types.

   By default, the UNDEFINED type map will be generated with null indicators and count fields present and in the same position as for the REORG UNLOAD EXTERNAL type. These may be moved or deleted to correspond to user-defined unload formats, but they can only be interpreted if their position is before the data field to which they apply.

9. Click Finish. The Import Copybook wizard opens automatically.

10. In Source, select Remote radio. This is the node name where the member exists.

11. In Source, select DB2 Catalog from the Type pull-down box. The DB2 Catalog is, most likely, the preferred method of importing metadata, but if required you can create or amend the data map by importing a Cobol copybook.

   Click Next. The Import Copybook - Remote DB2 Catalog Details dialog box appears.

   You can save the copybook as a local file by entering the file name in the Save File Locally As box.

   - Pad Variable. Whether padding is checked or not will be determined when the user imports from the DB2 Catalog during data map creation.

   - Null Indicator. Current Null Control is configurable. The indicator to be used is picked up from the new data map property “Null Indicator.”

12. Click Next. The Import Copybook - Configuration Details dialog box appears requesting what actions to be taken when loading the copybook.

   The options request prompts in various situations and actions that will be taken if duplicate record, fields and/or tables are detected.

13. Click Finish.

14. Click OK on the Import Copybook Information dialog box.
15. On the Record Definition dialog box, click OK for the EMPLOYEE.

16. The next screen display is shown below.

17. This dialog box could have been suppressed if the Prompt on record import box had been disabled. The following Error/Log Message should be displayed.

18. Click EMPLOYEE record and table to view the fields and columns that have been imported.

19. Thereafter, the data can be sourced in the normal way. After the import is complete, the data map is identical to one created by any other means.
Row Testing a DB2 Unload File Data Map

1. Click EMPLOYEE table and click the Row Test icon.
2. On the Database Row Test dialog box, click Go. The result of the Row Test should be as shown below.

3. Close the Database Row Test dialog box.

Accessing Unload Data from Multiple Tables

Multi-table tablespaces will result in a single unload file containing data from several tables. Individual tables can be selected from the unload using the OBID of the table.

Two types of unload file are supported for this type of multiple table access:

- **Type 1 REORG UNLOAD**
  - OBID is held in the fourth and fifth bytes of each record
- **Type 3 DSNUTILB UNLOAD**
  - OBID is held in the first two bytes of each record

The process for creating a data map that can be used to access multiple tables in the same unload file is as follows:

1. A data map is created as normal.
2. The underlying tables are added.
3. The underlying table for each record type in the map is specified by entering the appropriate OBID into the record properties dialog box for the DTL__OBID field.

Example

The following example uses a Type 3 DSNUTILB UNLOAD file but the process is the same if you are using a Type 1 REORG UNLOAD file.

Scenario

- The unload file that is being mapped contains data from two underlying tables each with its own structure.
- Two tables are PWXTAB3 and PWXTAB4.
- The underlying table is specified by the OBID in the first two (for Type 3) bytes. The OBIDs for the tables are x’0004’ and x’0005’ respectively.
Procedure

1. Map the table (PWXTAB3) in the normal way. See “Creating Data Maps Using DB2 Unload Files” on page 29.

2. Right-click the DTL_OBID field and select the Properties menu item. Enter the OBID hex value of x’0004’ into the Record ID Values box.

3. Click OK. The DTL_OBID field icon is green showing that the field has a Record ID filter set.

4. You can then import the next underlying table (PWXTAB4) and repeat steps 2 and 3 using an OBID value of x’0005’.

You can now test that the OBID filter is working by performing a row test against one of the two tables. In this case, you will row test the PWXTAB4 table.

5. Select table PWXTAB4 and click the Row Test icon.

6. Click Go.

Note: Only two rows from the unload file are returned. These were the rows that were identified by an OBID of x’0005’.

Amending the Record Layout

One of the advantages of DB2 data maps is that the record layout of the DB2 table can be enhanced by amending the data map.

To group these three fields together as a single field called MISC_DATA, do the following:

1. In the record display, select the MISC_1 field.

2. Right-click to select Properties.

This displays the Field Properties dialog box.

3. Change the Field Type to GROUP and the name to MISC_DATA.

You can then add fields to the GROUP field.

4. Right-click and choose Add Field as Child. Add the MISC_1, MISC_2, and MISC_3 fields as children.

PowerExchange maps this as follows:

Creating IDMS Data Maps

PowerExchange enables access to non-relational data through SQL as though the data source was relational. The first step to accessing IDMS through PowerExchange is to create a data map with PowerExchange Navigator. Use PowerExchange Navigator to build the following components:

♦ Data maps. The relational to non-relational definitions that will be used to access IDMS data sources.
♦ Personal metadata. Parameters to browse metadata from remote databases.
Adding IDMS Data Maps

The IDMS copybook should be imported automatically. There is a manual option to allow for the use of COBOL and PL/1.

PowerExchange downloads metadata from MVS to create the data maps for IDMS. When running with a security level of 0 or 1, data sets are created on MVS prefixed with the user ID under which the PowerExchange Listener is running. If this is not a suitable user ID, you can use TEMPHLQ statement in the DBMOVER configuration file. For more information about PowerExchange security, see the PowerExchange Reference Manual.

To add a data map:
1. Click Add > Data Map from the toolbar of the PowerExchange Navigator.
2. Select IDMS in the Access Method field and enter values for Schema Name and Data Map Name.
3. Select Import Record Definitions to import the record layout, and then click Next.
4. The IDMS Access Method dialog box appears. This dialog box has the following fields:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sub Schema Name</td>
<td>Required. Name of the sub-schema to be used for data retrieval.</td>
</tr>
<tr>
<td>DBName</td>
<td>Database name</td>
</tr>
<tr>
<td>Program Name</td>
<td>Program identification string</td>
</tr>
<tr>
<td>DBNode</td>
<td>Distributed Database System (DDS) node name</td>
</tr>
<tr>
<td>Dictionary Name</td>
<td>Name of the dictionary</td>
</tr>
<tr>
<td>Dictionary Node</td>
<td>DDS dictionary node name</td>
</tr>
<tr>
<td>Ready Mode</td>
<td>Select one of:</td>
</tr>
<tr>
<td></td>
<td>- Exclusive. Prevents concurrent use by any others.</td>
</tr>
<tr>
<td></td>
<td>- Protected. No updates are allowed by other processes until the run unit is complete.</td>
</tr>
<tr>
<td></td>
<td>- Shared. Default. No locking or protection is activated.</td>
</tr>
<tr>
<td></td>
<td>- Select one of:</td>
</tr>
<tr>
<td></td>
<td>- Retrieval. Default. Data retrieval mode.</td>
</tr>
<tr>
<td></td>
<td>- Update. Data update mode.</td>
</tr>
<tr>
<td>Ready All</td>
<td>If selected, ensures all areas are readied.</td>
</tr>
</tbody>
</table>

Note: The Sub Schema Name field is mandatory. All of the other fields are optional and are determined by the installation configuration and standards.

5. Click Finish.
6. The Import Copybook wizard opens automatically. Select Remote and IDMS in the Type field.

COBOL and PL/I imports are allowed with IDMS. However, you must add the IDMS record navigation information manually. For more information, see “Adding Owner and Set Names” on page 36. For more information about COBOL and PL/I copybook imports, see “Importing Copybooks” on page 65.

7. Click Next.
8. The Import Copybook - Remote IDMS Details dialog box displays. This dialog box has the following fields:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dictionary Name</td>
<td>Required. Specifies the name of the dictionary.</td>
</tr>
<tr>
<td>Location</td>
<td>Specifies the node name from the NODE statement in dbmover.cfg on Windows for the MVS system where the IDMS database resides.</td>
</tr>
</tbody>
</table>
9. Click **Advanced**. The **IDMS Advanced Properties** dialog box displays. Use the Advanced button to provide additional information for acquiring the IDMS metadata. This dialog box has the following fields:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Save File Locally As</td>
<td>Specifies the local file name for the IDMS metadata. Click <strong>Browse</strong> to search for the file.</td>
</tr>
<tr>
<td>DBName</td>
<td>Specifies the database name.</td>
</tr>
</tbody>
</table>

**Note:** IDMS parameters are determined by the installation configuration and standards.

10. Click **OK** or **Cancel**.

When you click **OK**, the PowerExchange Navigator communicates with the PowerExchange Listener on the MVS system described by the node name specified in the **Location** field. The PowerExchange Listener then submits batch job IDMSMJCL to import the IDMS metadata. IDMSMJCL is a member in the RUNLIB library on that MVS system.

11. Once the IDMS metadata has been imported, click **Next** or **Finish**.

12. When you click **Next**, the **Import Copybook - Configuration Details** dialog box displays.

   IDMS log-based change data capture works at a single record level. On this dialog box, there are two options that determine the representation of data required. If this data map is to be used for change data capture, make sure the **Create Table on Each Record Imported** option is checked.

   If this data map is being set up for bulk data movement, relational representation of the IDMS data may be required. If so, check the **Create Tables for IDMS Hierarchical Paths** option.

   Click **Finish**.

13. Click **OK**.

14. On the **Record Definition** dialog box, click **Apply** to import the rest of records.

15. Click **OK** for the **COVERAGE** record.

16. Close the **IDMS Import** dialog box.

**Database Row Test**

After you create a data map and store the data map in PowerExchange, you can test the data source by using **Database Row Test**. You can perform a row test against the data map to preview the data in the IDMS database and to ensure that the data map is functional. For more information about **Database Row Test**, see “Using **Database Row Test**” on page 107.
Changing IDMS Record Properties

You can change IDMS record properties after you have imported the IDMS metadata.

To change IDMS record properties:

1. Right-click on the data map record, and then click **Properties**.
2. The **Record Properties** dialog box displays. The **Name** tab of this dialog box has the following fields:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Record Name</td>
<td>PowerExchange record name in the data map.</td>
</tr>
<tr>
<td>IDMS Record Name</td>
<td>The internal IDMS record name as known to IDMS.</td>
</tr>
<tr>
<td>Area Name</td>
<td>The internal IDMS area name within which the IDMS record exists.</td>
</tr>
<tr>
<td>Record ID</td>
<td>Uniquely identifies each record type within the schema.</td>
</tr>
</tbody>
</table>
| Location Mode          | The technique that IDMS uses to physically store occurrences of the record type:  
                         | - **CALC.** A method of determining the target page for storage of a record in the database. The target page is calculated by a randomising routine executed against the CALC key in the record. Enables the Duplicates list.  
                         | - **DIRECT.** Occurrences of the record are stored on or near a page specified by the user program at runtime.  
                         | - **VIA.** Occurrences of the record in a specific set are stored relative to their owner.  
                         | - **VSAM.** Specifies the record is native VSAM format.  
                         | - **VSAM CALC.** Specifies the record is native VSAM format for which CALC access is required. Enables the VSAM Type and Duplicates lists and the CALC Element Names tab.  
| Duplicates             | Enabled if CALC or VSAM CALC is selected from the Location list:  
                         | - Not Allowed. IDMS does not store occurrences with duplicate CALC keys.  
                         | - First. Logically positions record occurrences with a duplicate CALC key before the duplicate record already stored.  
                         | - Last. Logically positions record occurrences with a duplicate CALC key after the duplicate record already stored.  
                         | - By DBKey. Logically positions record occurrences with a duplicate CALC key according to the db-key.  
| VSAM Type              | Identifies the record as a native VSAM and how the file containing the record was defined. Enabled if VSAM or VSAM CALC is selected from the Location list:  
                         | - Fixed NonSpanned. Specifies a fixed length record that cannot span VSAM control intervals.  
                         | - Fixed Spanned. Specifies a fixed length record that can span VSAM control intervals.  
                         | - Variable NonSpanned. Specifies a variable length record that cannot span VSAM control intervals.  
                         | - Variable Spanned. Specifies a variable length record that can span VSAM control intervals.  
| Page Group             | Maximum value is 32767.                                                    |
| Radix                  | Valid values are 0 to 12.                                                  |
| Compress               | Records held in compressed format.                                         |
| Variable               | Records held as variable record length.                                    |
3. Click the **Owner Records and Set Names** tab. This tab has the following fields:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Owner Name</td>
<td>IDMS owner records of this record as defined within IDMS.</td>
</tr>
<tr>
<td>Set Name</td>
<td>Relevant IDMS set names.</td>
</tr>
</tbody>
</table>

### Adding Owner and Set Names

You can add owner and set names for the records in the display on the **Owner Records and Set Names** tab.

**To add an owner and set name:**

1. In the **Owner Records and Set Names** tab display, click on the record.
2. Click the **Add Owner** icon.
3. The **Owner Record and Set Details** dialog box displays.
   - The field values above must be valid IDMS record names and set names and have a valid relationship with the record for which the properties are being added.
4. Click **OK**.

   The new details appear on the **Owner Records and Set Names** tab.

   **Note:** This is not the recommended process for setting up an IDMS data map, but is available for flexibility. For information about the recommended process, see the import procedure outlined in “Adding IDMS Data Maps” on page 33.

   For more information about the objects on this tab, see the online help.

### Deleting an Owner and Set Name

**To delete an owner and set name:**

1. In the **Owner Records and Set Names** tab display, click on the record.
2. Click the **Delete Owner** icon.

### Calc Element Names

The Calc Element Names tab appears if you select a location of **CALC** or **VSAM CALC** on the **Name** tab. This allows you to define one or more fields that are combined to form a single key for the record. The following table describes the fields in this dialog box:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Name of the record element to be used.</td>
</tr>
<tr>
<td>Position</td>
<td>Position offset from the start of the record.</td>
</tr>
<tr>
<td>Length</td>
<td>Length of the element.</td>
</tr>
</tbody>
</table>

**To add a calc element name:**

1. Click the **Calc Element Name** tab for the record on the data map.
2. Click the **Add** icon. Enter the new CALC element name in the **Name** field and provide values in the **Position** and **Length** fields.
3. Click **OK**. The new details appear in the **Calc Element Name** dialog box.
To delete a Calc Element Name:
1. Click the **Calc Element Name** tab for the record on the data map.
2. Select the calc element entry.
3. Click the **Delete** icon.

**Adding a New Table to an IDMS Data Map**

PowerExchange develops a relational view of the IDMS network environment. Additional tables might need to be created for particular applications.

**To add a new table:**
1. Open the data map, and then right-click the map name.
2. Select **Add Table**. The Add Table dialog box appears.
3. Right-click on a record in the **Available Records** list, and select **Add Record**.
   **Note:** Capture data is only generated for the lowest level IDMS record within a table. When a table that contains information from two IDMS records is registered for capture, only data changes to the lowest level record within the table appear.

**Adding Expressions**

Additional information can be returned on each record by adding expressions to the data map. Expressions such as `GetDbKeyOfOwner` and `GetDataFlowType` can be carried through to logged capture information. This functionality is designed to help the creation of relational structures, possibly data warehouses, from IDMS source data.

For more information about using these expressions, see “Expressions Tab” on page 143.

**Creating IMS Data Maps**

Data maps for IMS allow SQL statements to access IMS hierarchical data. PowerExchange uses SQL to run relational-type extracts against source data. To use IMS as a source database for PowerExchange, create data maps to map IMS segments and fields to relational tables.

You can use data maps to align fields, convert dates, and filter and enhance source data, which improves data accuracy and reduces data volume. You can use the data maps to extract data and move it from a source to target.

**Data Map Tasks**

Use the PowerExchange Navigator to add data maps to PowerExchange. When you create a data map, use the nonrelational view of data to define the layout of actual physical data by adding segment and field definitions. Provide the relational source view of data by defining tables and columns based on the record and field definitions.

If the IMS DBD source is available, you can import it to define segments, key fields, and search fields for the data map. For each IMS segment that you import, also import a COBOL copybook to overlay the segment with its COPYLIB.

During the import, the IMS segments and fields are automatically mapped to tables, creating the relational format of source data that PowerExchange requires.
For each data map you create, store the data map in PowerExchange by sending it to a PowerExchange Listener on the local or remote node so that PowerExchange can extract the related source data. This converts the data map into a platform-independent file which can be accessed by PowerExchange extract run-time routines.

Test data maps by using the PowerExchange Navigator Row Test options. This process sources data as one or more columns from the real source and displays it back in a tabular character format with column headings, so that you can see what would be retrieved by the run-time routines.

You can use data maps for Lookup transformations to look up related values and determine if rows exist for an IMS target. When you create a data map for IMS, you can also use IMS as a source or target database.

Table 2-1 summarizes tasks to complete when you create IMS data maps:

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add a new data map to the PowerExchange Navigator</td>
<td>Use the PowerExchange Navigator to add a new data map to PowerExchange by creating a data map name and select the method for accessing IMS. For more information, see &quot;Adding IMS Data Maps Using PowerExchange Navigator&quot; on page 42.</td>
</tr>
<tr>
<td>Import an IMS DBD source to a data map</td>
<td>If the DBD source is available for the IMS data, import the DBD source to define segments and the hierarchical sequence for the data map. For more information, see &quot;Importing IMS DBD Source for IMS Data Maps&quot; on page 43.</td>
</tr>
<tr>
<td>Import a COBOL copybook for each segment in the data map</td>
<td>For each IMS segment in a data map, import a COBOL copybook to overlay the segment with its COPYLIB. For more information, see &quot;Importing Copybooks for IMS Data Maps&quot; on page 85.</td>
</tr>
<tr>
<td>Store the data map in PowerExchange</td>
<td>For each data map you create, store the data map in PowerExchange by sending it to the PowerExchange Listener on the local or remote node. This converts the data map into a platform-independent file which can be accessed by the PowerExchange extract run-time routines.</td>
</tr>
<tr>
<td>Test a data map</td>
<td>After you create a data map, you can test the data source by using the Row Test options. This process shows you which data will be retrieved by PowerExchange run-time routines. For more information, see &quot;Testing IMS Data Maps&quot; on page 49.</td>
</tr>
</tbody>
</table>

Data Map Structure

As a default, PowerExchange provides a three-tier qualifier system so you can organize data maps in projects and groupings with logical naming conventions. Use this standard when you name data maps for the NRDB access method. For the NRDB2 access method, choose a two-tier qualifier system by selecting a data map and then selecting Options > Preferences.

For example, when you add a data map, specify a schema name such as "customer," a data map name such as "billing," and a table name such as "tablenn."

At runtime, PowerExchange uses SQL to select data from the table with the schema and data map name as a prefix to the table name. For a two-tier-qualifier system, the data map created is "customer.billing" and SQL to select data is "select * from customer.billing_tablenn." For a three-tier-qualifier system, the SQL to select data is "select * from customer.billing.tablenn."

Within a data map, you can use any number of segment and table definitions. However, you can define one physical file for each data map. For example, you might have multiple segment definitions if multiple segments exist in the IMS database, whereas from a single segment, many different table definitions can be created.

Note: In PowerExchange, the term "record" describes a root segment and its children.

Data maps that you create appear in the Resources tab of the PowerExchange Navigator.
Nonrelational View of Data

Figure 2-1 shows an IMS segment that was added to a data map during an import of the IMS DBD source. This figure shows the nonrelational view of data for the data map:

![Figure 2-1. IMS Segment View](image)

**Note:** When you import data for a single segment, a single segment appears in the data map. If you import data for multiple segments, multiple segments appear in the data map.

Relational View of Data

Figure 2-2 shows a table generated for an IMS segment during an import of the IMS DBD source. This figure shows the relational view of data for the data map:

![Figure 2-2. Table View](image)
**View of IMS Hierarchy**

The IMS hierarchy that results from the DBD source import is available for each IMS data map that you create.

**To view an IMS Hierarchy:**
- In the Resource Explorer, right-click a data map and select **Display IMS Hierarchy**.

**Examples**

The PowerExchange installation includes a basic IMSDEMOS example that provides a sample IMS DBD source that you can use to create a sample IMS data map. For more information, see “IMS Data Map Examples” on page 50.

**Prerequisites for Creating IMS Data Maps**

Before you create and test an IMS data map, complete the following prerequisites:

- **Acquire IMS source information.** Make sure that you are aware of specific IMS information you need to create a data map. For more information, see “Acquiring IMS Source Information.”

- **Determine the PSB for access to IMS.** Decide which PSB to use and which PCB within the PSB to use. For more information, see “Determining the PSB for Access.”

- **Update the PowerExchange configuration on MVS.** Update the DBMOVER member of the RUNLIB library. For more information, see “Updating the PowerExchange Configuration on MVS” on page 41.

- **Update the PowerExchange configuration on Windows.** Update the dbmover.cfg file for the PowerExchange Navigator. For more information, see “Updating the PowerExchange Configuration on Windows” on page 41.

- **Ensure that the data map represents a complete IMS segment.** If you plan to perform an INSERT or UPDATE to an IMS segment, the data map must represent the complete segment length as defined in the IMS DBD. If it does not, an INSERT or UPDATE to the segment could result in non-blank data being written to the end of the segment that was not defined as FILLER.

**Note:** To avoid this issue, you can add a FILLER definition to the COPYLIB before you import it to PowerExchange.

**Acquiring IMS Source Information**

Table 2-2 describes the information you should gather for the IMS source data before you create an IMS data map:

<table>
<thead>
<tr>
<th>Source Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IMS database to access</td>
<td>IMS database and IMS database data set name</td>
</tr>
<tr>
<td>PSB</td>
<td>PSB to use and which PCB within the PSB to use -and- If the PSB has COMPAT=YES?</td>
</tr>
<tr>
<td>IMS subsystem</td>
<td>SSID</td>
</tr>
<tr>
<td>IMS segments</td>
<td>Which segments to use and if any segments are of variable length -and-</td>
</tr>
<tr>
<td></td>
<td>Hierarchical sequence as defined in the DBD source</td>
</tr>
<tr>
<td>IMS DBD source</td>
<td>Data set location</td>
</tr>
</tbody>
</table>
Determining the PSB for Access

Before you create a data map, determine the PSB access as follows:

- **For IMS Non-ODBA access.** Determine which PCB within a PSB to use. Count the number of PCBs in the PSB. If the PSB has COMPAT=YES, add one to that number.
- **For IMS ODBA access.** Determine which PCB to use within a PSB. Note the PCB name because you will need to enter it when you create a data map.
- **PROCOPT of the PCB.** To reduce the number of locking conflicts, make sure to set the PROCOPT to read-only for the PCB for the IMS database.
- **PSB definition in the IMS SYSGEN.** If the PSB is used with the IMS ODBA interface or a Netport BMP job, make sure the PSB is defined in the IMS SYSGEN.

Updating the PowerExchange Configuration on MVS

Before you can access IMS data using a data map, you must update PowerExchange on MVS so that it is configured to access the IMS source data. Configuration varies depending on whether you access IMS by using DL/I Batch or IMS ODBA. To activate configuration changes, restart the PowerExchange Listener if it is running.

**To use DL/I Batch or BMP to access IMS:**

1. Update the PowerExchange DBMOVER member in RUNLIB as follows:
   
   Check the PowerExchange LISTENER or NETPORT statement in the DBMOVER configuration member to verify that the PSB name is correct. You might need to add PowerExchange LISTENER or NETPORT entries if you use more than one PSB. For more information, see the PowerExchange Reference Manual.

2. Update the IMSJCL member in the PowerExchange RUNLIB library. Modify this member so that it works in your environment as either a DL/I Batch or BMP job.

**To use IMS ODBA to access IMS:**

1. Include the RESLIB in the PowerExchange Listener STEPLIB or make sure it is accessible through the MVS LNKLST concatenation.

2. Use a PSB that has PCBNAMES defined.

Updating the PowerExchange Configuration on Windows

Before you create a data map, update PowerExchange on Windows where the PowerExchange Navigator is located so that it is configured to access the IMS source data. Configuration varies depending on whether you access IMS by using DL/I Batch or IMS ODBA. To activate configuration changes, restart the PowerExchange Listener if it is running.

**Note:** You do not need to start the PowerExchange Listener on Windows to access the IMS data.

You must update the PowerExchange dbmover.cfg configuration file located in the PowerExchange installation directory.

**To use DL/I Batch or BMP to access IMS:**

- Add a NODE statement pointing to the NETPORT port for the PowerExchange Listener on MVS. The format is:
  
  \[NODE=(location_name,TCP/IP,[hostname_or_IPaddress],port_number)\]

**To use IMS ODBA to access IMS:**

- Add a NODE statement pointing to the PowerExchange Listener port on MVS. The format is:
  
  \[NODE=(location_name,TCP/IP,[hostname_or_IPaddress],port_number)\]
Adding IMS Data Maps Using PowerExchange Navigator

Use the Name and Access Method options to add a data map to the PowerExchange Navigator. Specify a schema and data map name, and then select an access method to define how PowerExchange connects to an IMS database to retrieve the source data. The data map you add appears as “schema.datamapname” in the Data Maps folder of the Resources tab.

For IMS data maps, you can use DL/I Batch or IMS ODBA as the access method to define how to connect to the IMS database. A DL/I Batch method submits JCL to access the source data. Although the access method is DL/I Batch, you can choose DL/I or BMP for the Netport job. An IMS ODBA method connects to an IMS database through an open source database connection that has been established.

Before you add a data map, you should have the following information:

- IMS subsystem ID
- PCB name or number
- For the DL/I Batch access method, know the PCB number within the PSB. If the PSB has COM PAT=YES then PCB 1 is reserved and your database PCBs start with number 2. If the PSB does not have COM PAT=YES, use the number 1 through the last number used for the PCB.
- For the IMS ODBA access method, the PSB name as well as the PCB name

Note: Before you add a data map to the PowerExchange Navigator, complete the prerequisites for creating a data map. For more information, see “Prerequisites for Creating IMS Data Maps” on page 40.

To add a data map:

1. In the PowerExchange Navigator, click Add > Data Map to start the wizard for creating a data map.
2. Complete the fields on the Name page.
   Table 2-3 describes the Name page fields:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Schema Name</td>
<td>Name to call the schema and include in the data map name “schema.datamapname.” Enter up to 256 characters. Can be a maximum of ten bytes long and must begin with an alphabetic character. At run time, the SQL to select data from this data map will use the “schema” and “data map name” as a prefix to the table name.</td>
</tr>
<tr>
<td>Data Map Name</td>
<td>Name to call the data map and use in the data map name “schema.datamapname.” Enter up to 256 characters. Can be a maximum of ten bytes long and must begin with an alphabetic character. At run time, the SQL to select data from this data map will use the “schema” and “data map name” as a prefix to the table name.</td>
</tr>
</tbody>
</table>
   | Access Method | Select one of the following access methods:  
   - DL/I BATCH to access an IMS database through submitted JCL.
   - IMS ODBA to access an IMS database through direct access by the PowerExchange Listener.
   - SEQ flat files to access an IMS database that you plan to use as a flat file. |
   | Import Record Definitions | Indicates to go directly to the import copybook options after creating the data map. |

3. Click Next.

   If you selected DL/I BATCH as the access method, the DL/I Batch Access Method page appears. Continue to the next step.
   -or-
   If you selected IMS ODBA access method, the IMS ODBA Access Method page appears. Go to Step 5.
4. Complete the fields on the DL/1 Batch Access Method page, and click Next. Table 2-4 describes the DL/1 Batch Access Method fields:

Table 2-4. DL/1 Batch Access Method Fields

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IMS SSID</td>
<td>IMS subsystem ID. This ID can be up to four letters in length.</td>
</tr>
<tr>
<td>DBD Name</td>
<td>Name of the IMS database description (DBD) for the IMS source.</td>
</tr>
<tr>
<td>PCB Number</td>
<td>Number of PSBs if PSBs have been generated CMPAT=YES in the IMSDEF member.</td>
</tr>
<tr>
<td></td>
<td>If you use PSBs that have not been generated CMPAT=YES, use the PCB in the order specified in the PSB member.</td>
</tr>
<tr>
<td>Skip First _ Records from File</td>
<td>If the file has one or more header records that are not required, define the number of header records to skip.</td>
</tr>
</tbody>
</table>

5. Complete the fields on the IMS ODBA Access Method page, and click Next. Table 2-5 describes IMS ODBA Access Method fields:

Table 2-5. IMS ODBA Access Method Fields

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IMS SSID</td>
<td>IMS subsystem ID. This ID can be up to four letters in length.</td>
</tr>
<tr>
<td>DBD Name</td>
<td>Name of the IMS DBD for the IMS source.</td>
</tr>
<tr>
<td>PSB Name</td>
<td>Name of the program specification block. Enter up to eight alphanumeric characters and use no special characters.</td>
</tr>
<tr>
<td>PCB Name</td>
<td>Name associated with the PCB in the PSB.</td>
</tr>
<tr>
<td>Data Codepage</td>
<td>Select the code page that describes the character set for the character data in the database.</td>
</tr>
<tr>
<td>Skip First _ Records from File</td>
<td>If the file has one or more header records that are not required, enter the number of records to skip.</td>
</tr>
</tbody>
</table>

6. Click Finish.

If you selected the option to import record definitions, import the IMS DBD source to the data map. For more information, see “Importing IMS DBD Source to IMS Data Maps” on page 43.

Importing IMS DBD Source for IMS Data Maps

Use the Import Copybook options in the PowerExchange Navigator to access and import the IMS DBD source. You import an IMS DBD source to a data map to define the segments and the hierarchical sequence, generate a hierarchical schema, and add search fields and CCK fields to segment definitions. This provides the relational view of data PowerExchange needs to run SQL-type extracts against the source data.

When you import a DBD source, the actual DBD source hierarchy is preserved in the IMS database. The DBD source representation will be placed in the local PowerExchange library.

Before you import an IMS DBD source, you should know the name of the IMS data set where the DBD source member resides.

PowerExchange will create a relational view based on the import. The relational view will map a table to each IMS segment and columns to each IMS field.

After you import the IMS DBD source, import a copybook for each segment. For more information, see “Importing Copybooks for IMS Data Maps” on page 85.
When the data map is complete, send it to a PowerExchange Listener on the local or remote node. This converts the data map to a platform-independent file. You can also test the data map by performing a row test to return valid data. For more information, see “Testing IMS Data Maps” on page 49.

Before you import a DBD source to a data map, complete the prerequisites for creating a data map. For more information, see “Prerequisites for Creating IMS Data Maps” on page 40.

To import an IMS DBD source:

1. If you selected the option to import record definitions when you added the data map, the Import Copybook - Source Details page appears when you finish the data map. Otherwise, you can display the Import Copybook - Source Details page from the Resources tab by selecting the data map and then selecting File > Import Copybook.

2. Complete the fields on the Import Copybook - Source Details page.

   Table 2-6 describes the Import Copybook - Source Details fields:

   Table 2-6. Import Copybook - Source Details Fields

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source</td>
<td>Indicates the DBD is local which means the data has been downloaded and stored locally, or remote which means the data will be downloaded from a remote location such as MVS.</td>
</tr>
<tr>
<td>Type</td>
<td>Select the type of import to perform: DBD.</td>
</tr>
<tr>
<td>Column Range</td>
<td>Defines the start and end columns to inspect. Typically this range should be 1-72 for a DBD source.</td>
</tr>
<tr>
<td>Last Import</td>
<td>Click to show the specification to which the DBD source will be imported. The current specifications are those of the previous import.</td>
</tr>
</tbody>
</table>

3. If you selected Local as the Source option, complete the fields on the Import Copybook - Local DBD Details page.

   Table 2-7 describes the Import Copybook - Local DBD Details field and button:

   Table 2-7. Import Copybook - Local DBD Details Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>File Name</td>
<td>Name of the DBD to import.</td>
</tr>
<tr>
<td>Preview button</td>
<td>Click to preview the DBD.</td>
</tr>
</tbody>
</table>

4. Complete the fields on the Import Copybook - Remote DBD Details page, and click Next.
Table 2-8 describes the Import Copybook - Remote DBD Details fields:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>File Name</td>
<td>PDS file name and member. Enter by using the following format: HLQ.DTLDEMO(dbdname)</td>
</tr>
<tr>
<td>Location</td>
<td>Remote NODE from the dbmover.cfg file.</td>
</tr>
<tr>
<td>UserID</td>
<td>MVS user ID if security has been implemented.</td>
</tr>
<tr>
<td>Password</td>
<td>MVS password if security has been implemented.</td>
</tr>
<tr>
<td>Name</td>
<td>Name of the file that the DBD source is to be saved as when it is retrieved from the remote platform.</td>
</tr>
<tr>
<td>Name Browse</td>
<td>Click to browse for the file.</td>
</tr>
<tr>
<td>Preview</td>
<td>Click to preview the DBD source.</td>
</tr>
</tbody>
</table>

5. Complete the fields on the Import Copybook - Configuration Details page.

Table 2-9 describes the Import Copybook - Configuration Details fields:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prompt on record import</td>
<td>When a record is created, prompt for a record name showing the first name found.</td>
</tr>
<tr>
<td>Prompt on field import</td>
<td>When a field is created, prompt for a name showing the name found.</td>
</tr>
<tr>
<td>Prompt on table creation</td>
<td>When a new table is created, prompt for a table showing the default name.</td>
</tr>
<tr>
<td>Create table on each record imported</td>
<td>Creates a table if a record is imported.</td>
</tr>
<tr>
<td>Create tables for DL1 hierarchical paths</td>
<td>Select when you work with DBDs to have all tables created for all DL/I hierarchical paths. Only only available for a DBD source.</td>
</tr>
<tr>
<td>Select first data redefinition</td>
<td>Select so that if a REDEFINES clause is found, the first data definition will automatically be used.</td>
</tr>
<tr>
<td>Start import automatically</td>
<td>Select to automatically start the import.</td>
</tr>
<tr>
<td>Action on duplicate record</td>
<td>Select the action to take if a duplicate segment is found: PROMPT, provide a UNIQUE NAME, OVERWRITE it, or SKIP it.</td>
</tr>
<tr>
<td>Action on duplicate field</td>
<td>Select the action to take if a duplicate field is found: PROMPT, provide a UNIQUE NAME, OVERWRITE it, or SKIP it.</td>
</tr>
<tr>
<td>Action on duplicate table</td>
<td>Select if a table with the same name is found: PROMPT, provide a UNIQUE NAME, OVERWRITE it, or SKIP it.</td>
</tr>
</tbody>
</table>

6. Click **Finish**.

7. Review the options that you selected for the import.

8. If the information is correct, click **OK**.

   If you chose to be prompted during the import, continue to “Using the Import Prompts” on page 87.

   -or-

   If you did not select prompts to occur, review the data that is imported.

9. After importing a DBD source to a data map, import a COBOL copybook for each segment that was imported. Importing an IMS DBD source adds segments, key fields and search fields from the IMS database and creates tables for a data map. After you import the IMS DBD source, importing a copybook
for each segment overlays each segment with its COPIYLIB. This redefines the data map while maintaining the hierarchical metadata information for the database. For more information, see “Importing Copybooks for IMS Data Maps” on page 85.

Returning an IMS RBA

You can return an RBA by creating a user-defined field for an IMS data map. Use the Expressions options in the PowerExchange Navigator to return an RBA.

To return an RBA:
1. On the Data Map tab of the Resource Explorer, select the record that represents the IMS segment.
2. Click the Expressions tab, Expr.
   Note: If no expression exists, a zero appears in parenthesis.
3. Right-click in the Expressions tab, and select Add Field at End.
4. Complete the Expressions options. Table 2-10 describes the Expressions options:

Table 2-10. Expressions Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Name to describe the RBA.</td>
</tr>
<tr>
<td>Type</td>
<td>Data type of the field.</td>
</tr>
<tr>
<td>Prec</td>
<td>Number of digits in the number, if applicable.</td>
</tr>
<tr>
<td>Scale</td>
<td>Number of digits to the right of the decimal point, if applicable.</td>
</tr>
<tr>
<td>Length</td>
<td>Length of the field.</td>
</tr>
<tr>
<td>Phase</td>
<td>Select one of the following options: - R (Read) - W (Write) - RW (Read and Write)</td>
</tr>
<tr>
<td>Expression</td>
<td>Expression used to populate the field.</td>
</tr>
<tr>
<td>Validation</td>
<td>Displays a message if there is a problem with the field.</td>
</tr>
</tbody>
</table>

Modifying IMS Data Maps

The PowerExchange Navigator provides options for modifying a data map by selecting segments and tables and manually changing the properties. Instead of manually changing properties, re-import an IMS DBD source or COBOL copybook if you need to update segments and tables.

Warning: Modifying segments and tables manually can result in loss of data in a data map.

For more information, see “Importing IMS DBD Source for IMS Data Maps” on page 43 and “Importing Copybooks for IMS Data Maps” on page 85.

Storing IMS Data Maps

Store each data map that you create so that PowerExchange can extract the related source data. Storing a data map involves sending the data map to the PowerExchange Listener on the same node as your IMS source data. This process converts the data map to a platform-independent file which can be accessed by the PowerExchange extract run-time routines.
To send a data map to the PowerExchange Listener:

1. In the Resource Explorer, select a data map and click **File > Send to Remote Node**.
2. Complete the **Data Map Remote Node options**.

   Table 2-11 describes the **Data Map Remote Node options**:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>User ID</td>
<td>Your user ID if security is set up on the remote MVS node.</td>
</tr>
<tr>
<td>Password</td>
<td>Your password if security is set up on the remote MVS node.</td>
</tr>
<tr>
<td>Location</td>
<td>Location (NODE from the dbmover.cfg file) to send the data map to.</td>
</tr>
<tr>
<td></td>
<td>This is also the platform where the data that will be extracted resides.</td>
</tr>
<tr>
<td>File Password</td>
<td>File password if the file is password protected.</td>
</tr>
<tr>
<td>Save User ID and Password(s)</td>
<td>Saves the user ID and password you entered for the rest of the session.</td>
</tr>
</tbody>
</table>

3. Click **OK**.

   After you store the data map, you can test the data source. For more information, see “Testing IMS Data Maps” on page 49.

**Using Lookup Transformations with IMS Databases**

You can use lookup transformations in PowerCenter to lookup data in IMS databases. You should use a separate netport job for each lookup transformation for PCBs with a “GOx” PROCOPT.

**Note:** Specify the IMS key values of a segment so that the SSA created will perform efficiently within IMS.

For more information about using a lookup transformation for IMS databases, see **PowerExchange Interfaces for PowerCenter**.

**Writing Data to IMS Databases**

You can write source data from a data map to an IMS database. When you write data to IMS, specify the key values of a segment so that all the processing involved in writing data for the segment takes place within IMS.

Use **PowerExchange Client for PowerCenter (PWXPC)** to write data to IMS. For more information about PWXPC, see **PowerExchange Interfaces for PowerCenter**.

When you write data to IMS, use the following guidelines:

- You cannot use SDEP segments.
- You cannot update a segment below an unkeyed segment.
- You can use segments that contain an OCCURS clause where the COPYLIB has been changed to identify each of the fields in the OCCURS clause as a separate field, which eliminates the OCCURS clause in the COPYLIB definition. You cannot generate a separate row for each occurrence of the OCCURS clause.
- When you import a data map to PowerCenter, make sure any CCK fields that you want to use as key fields are marked as key fields.
- In PowerCenter, use a lower Commit frequency to reduce the risk of locking segments.
- A separate NETPORT job is recommended for writing IMS data. This will allow you to access a PSB with write intent and modify the JCL to support updating IMS data, such as updating the IEF RDER log.
- When you write data for one of the following types of fields, indicate the field is optional as you create a data map:
  - When you plan to move spaces for a field to the database
When you plan to move no data for a specific field to the database
If you have several fields in one of these situations, you only need to specify that the first field in the data map is optional, and all fields following that field will be considered optional. If you do not specify the field is optional, errors will appear in the PowerCenter session log.

To write data to IMS databases:
1. Ping the IP address of the MVS system where the target database is located to make sure it is accessible.
2. Add a NODE statement that points to the remote platform in the PowerExchange configuration file dbmover.cfg.
3. Use the following command to test that the remote PowerExchange Listener is started:
   
   dtlrexe loc=xxxx prog=ping uid=<userid> pwd=<pwd>

   Where xxxx is the NODE name from the PowerExchange configuration file dbmover.cfg.
4. Use PowerExchange to create a data map. For information about creating an IMS data map, see “Creating IMS Data Maps” on page 37. To create data maps for other databases, see the appropriate PowerExchange adapter guide.
5. Store the data map.
6. Test the data map. For more information, see “Testing IMS Data Maps” on page 49.
7. Use PowerCenter to execute the request to move the data to IMS. For more information, see PowerExchange Interfaces for PowerCenter.

Viewing or Changing IMS Options

Use the IMS Options for a data map that will be used to write to an IMS database. These options provide default values that are set for optimal performance when working with IMS.

Changing the default settings impacts the amount of data sent to the database and the time required to write the data. Because of these options, PowerExchange regards IMS as if it is a relational target. The defaults are set so that PowerExchange acts like a typical IMS COBOL application program.

Warning: Do not change the Delete Options default settings unless you are familiar with the IMS hierarchy. Changing these settings can result in loss of IMS data.

To view or change IMS options:
1. In the Data Map tab, right-click a table and select Properties.
2. Click the IMS Options tab and complete the IMS Options.

Table 2-12 describes the IMS Options.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Update/Insert Options</td>
<td>Indicates which segment level to apply the update or insert to. Select one of the following options:</td>
</tr>
<tr>
<td></td>
<td>- Lowest segment only. Applies the action to the lowest segment only.</td>
</tr>
<tr>
<td></td>
<td>- All possible levels. Applies the action to all segment levels.</td>
</tr>
<tr>
<td>Update all matching segments</td>
<td>Indicates all non-unique matching segments should be updated.</td>
</tr>
<tr>
<td></td>
<td>Note: You cannot select this option for complex, compound tables.</td>
</tr>
</tbody>
</table>
Testing IMS Data Maps

After you create a data map and store the data map in PowerExchange, you can test the data source by using the Row Test options. You can perform a row test against the data map to review the data that will be retrieved by PowerExchange run-time routines.

This process sources data as one or more columns from the actual source and displays it in a tabular character format with column headings so that you can see what would be retrieved by the run-time routines. If the result is not what you expected, you can change the data map and run the row test again.

**Note:** Before you test a data map, complete the prerequisites for creating a data map. For more information, see “Prerequisites for Creating IMS Data Maps” on page 40.

**To perform a row test:**

1. In the Data Map tab of the Resource Explorer, select a table.
2. Select **File > Database Row Test**.
3. Click **Yes** to run the test by sending the data map to the PowerExchange Listener on the node where the IMS source data is located.
4. Complete the Database Row Test options.

Table 2-13 describes the Row Test options:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB_Type</td>
<td>Select the NRDB DB type.</td>
</tr>
<tr>
<td>Location</td>
<td>The location of the PowerExchange Listener as specified in dbmover.cfg. This must be a Netport location. The port number must correlate to the location in the DBMOVER configuration file on MVS. The NETPORT statement points to a RUNLIB member called IMSJCL when the product is installed. Using this location will cause the IMSJCL job to be submitted on MVS.</td>
</tr>
<tr>
<td>Override File Name</td>
<td>Not applicable to IMS.</td>
</tr>
<tr>
<td>File Password</td>
<td>Not applicable to IMS.</td>
</tr>
<tr>
<td>UserID</td>
<td>User ID if security is activated for the PowerExchange Listener.</td>
</tr>
<tr>
<td>Password</td>
<td>Password if security is activated for the PowerExchange Listener.</td>
</tr>
</tbody>
</table>
5. Click Go.

6. When the row test is complete, view the row test output.

## IMS Data Map Examples

These examples provide a sample IMS DBD source and steps to create a single-segment data map using the sample DBD source.

Before you complete the steps to create the example data map, review “Prerequisites for Creating IMS Data Maps” on page 40.

### IMS DBD Source Example

The following IMS DBD source example is provided with your PowerExchange installation:

```plaintext
DBD   NAME=IMSDEMO,ACCESS=(HDAM,VSAM),X
      RMNAME=(DFSHDC40,50,80),X
      EXIT=(*,KEY,PATH,(CASCADE,KEY,PATH),LOG)
DATASET DD1=IMSDEMO,DEVICE=3380,SIZE=8192,SCAN=3
SEGM NAME=IMSDEMS,BYTES=80FIELD NAME=(DTLKEY,SEQ),BYTES=9,START=3
       FIELD NAME=RECTYPE,BYTES=1,START=12FIELD NAME=AMOUNT,BYTES=4,START=13
       FIELD NAME=BINNO,BYTES=4,START=17FIELD NAME=DECNO,BYTES=3,START=21
       FIELD NAME=DTYY,BYTES=2,START=24FIELD NAME=DTMM,BYTES=2,START=26
       FIELD NAME=DTDD,BYTES=2,START=28FIELD NAME=ACCT1,BYTES=10,START=30
       FIELD NAME=ACCT2,BYTES=10,START=40
       FIELD NAME=ACCT3,BYTES=10,START=50
DBDGEN
FINISH
```

When you install PowerExchange on MVS, a number of libraries are created with the High Level Qualifier specified in the MVS Install Assistant. On MVS, run the following job to create the IMS environment for the example installation:

```plaintext
hlq.DTLDEMO(IMSDEF)
```

### Creating an IMS Data Map Example

Use the following steps to create a basic, single-segment data map that uses the IMS DBD source example. When you use the IMSDEMO example to complete these tasks, you will import a DBD source to define the segments and the hierarchical sequence and generate hierarchical schema. You will also import a COBOL copybook for the segment. The Access Method to use for these steps is DL/1 BATCH.

---

**Table 2-13. Row Test Options**

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fetch</td>
<td>Select one of the following options:</td>
</tr>
<tr>
<td></td>
<td>- Columns – metadata request to retrieve information about columns</td>
</tr>
<tr>
<td></td>
<td>- Data – actually retrieve data rows</td>
</tr>
<tr>
<td></td>
<td>- Foreign keys – metadata request to retrieve information about Foreign keys</td>
</tr>
<tr>
<td></td>
<td>- Primary keys – metadata request to retrieve information about Primary keys</td>
</tr>
<tr>
<td></td>
<td>- Procedure Cols – metadata request to retrieve information about particular stored procedures</td>
</tr>
<tr>
<td></td>
<td>- Procedures – metadata request to retrieve information about available stored procedures</td>
</tr>
<tr>
<td></td>
<td>- Records – metadata request to retrieve information about records</td>
</tr>
<tr>
<td></td>
<td>- Schemas – metadata request to retrieve information about schemas</td>
</tr>
<tr>
<td></td>
<td>- Tables – metadata request to retrieve information about tables</td>
</tr>
</tbody>
</table>

**SQL Statement**

Generated in the format “Select * from Schema.Map_table name (segment)”.
To create a sample IMS data map:

1. In the PowerExchange Navigator, click Add > Data Map.

2. Enter IMSS1 in Schema Name, select DL/1 BATCH in Access Method, and then enter IMSM1 in Data Map Name.

3. Click Next.

4. Select Remote, select DBD in Type, and then click Next.

   Note: The following steps demonstrate a DBD import to define segments and the hierarchical sequence, with data in columns 1-80.

5. Use the Import Copybook Remote DBD Details options to specify the sample DBD source file name and location.

   Enter the PDS name and member by using the following format:

   hlq.DTLDEMO(dbdbname)

   Specify member IMSDBD in PowerExchange DTLDEMO library.

6. Select your MVS location.

7. If this is the first MVS connection you are using for PowerExchange, update the PowerExchange dbmover.cfg configuration file on Windows where the PowerExchange Navigator is located. The format is:

   NODE=(location name,TCPIP,[host name or TCP/IP address],[port number])

   The following location is used for this example:

   NODE=(MP3doc,TCPIP,MP3000,6507)

8. If security has been implemented for the PowerExchange Listener on MVS, enter the MVS user ID and password.

9. Enter the file name.

   The DBD representation will be placed in the local PowerExchange library.

10. Click Next. Use the Import Copybook Configuration Details defaults to confirm the selections you have made so far:

    Note: When you work with DBDs and you want to have all tables created for all DL/I hierarchical paths, select Create tables for DL/I hierarchical paths.

11. Click Finish.

12. Click OK to import the DBD source

13. Use the Record Definition options to complete the import. For this example, select Import. You can skip a record type if required or stop an import, if needed.

   During the import process, use the tools described in Table 2-14 to complete the import:

<table>
<thead>
<tr>
<th>Tool</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Next Redefinition icon</td>
<td>Moves the blue arrow on left to point at the line that you want to select.</td>
</tr>
<tr>
<td>Previous Redefinition icon</td>
<td></td>
</tr>
<tr>
<td>Resume Import icon</td>
<td>Resume the import if prompted.</td>
</tr>
</tbody>
</table>

   Note: The IMSDemo DBD source example imports a single segment. For multiple segments, prompts would be displayed to accept each segment.

14. View the Error/Log Message information for the import process.

15. In the Data Map tab of the PowerExchange Navigator, select the IMS segment and click File > Import Copybook to import a copybook for the segment.
16. Select Remote, and then select COBOL for Type. Click Next.

17. Enter the copybook name and location, and then enter a name to save the file locally. Click Next.

18. Use the default Import Copybook Configuration Details options and click Finish.

19. Review the options you selected for the import.

20. If the information is correct, click OK.

21. Follow the import prompts and make updates, if needed. For more information, see “Using the Import Prompts” on page 87.

22. When the import is complete, review the data.

You can store the data map with PowerExchange and test the data map. For more information, see “Testing IMS Data Maps” on page 49.

Creating VSAM Data Maps

PowerExchange enables access to nonrelational data through the use of SQL, as though the data source was relational. To access VSAM data through PowerExchange, you must create a data map using the PowerExchange Navigator. Use the PowerExchange Navigator to build data maps and personal metadata profiles for VSAM data sets.

PowerExchange can access the following types of VSAM data set:

- Key-sequenced data set (KSDS)
- Entry-sequenced data set (ESDS)
- Fixed-length relative record data set (RRDS)
- Variable-length relative record data set (VRRDS)

These types of VSAM data sets can be basic format, extended-format, or compressed data sets.

In order to create a data map, you must start the PowerExchange Navigator, either from the shortcut created when PowerExchange was installed or by running dtlui.exe from the Windows Start > Run option.

To create VSAM data maps:

1. Select the Add Data Map dialog using one of the following methods:
   - Click the Add Data Map icon on the toolbar.
   - Click Add > Data Map.
   - Right-click Data Maps, and then click Add Data Map.

2. The Name dialog box displays. Enter values for Schema and Data Map Name, and then select the appropriate value for the type of VSAM data set in the Access Method list.

   Note: Select RRDS for both fixed-length and variable length RRDS (VRRDS) data sets.

3. Select the Import Record Definitions box to import the record layout for the VSAM data set.

The **ESDS Access Method** dialog box has the following fields:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>File Name</td>
<td>Specifies the fully-qualified data set name of the VSAM data set.</td>
</tr>
<tr>
<td>Number of Data Buffers</td>
<td>Specifies the number of I/O buffers for data control intervals that PowerExchange requests VSAM to allocate. Default is 2.</td>
</tr>
<tr>
<td>CI ACCESS</td>
<td>If selected, specifies that PowerExchange should access the entire contents of a control interval rather than individual data records when reading VSAM data sets. This option improves read performance. For more information, see “Improving Bulk Read Performance for VSAM Data Sets” on page 55. <strong>Note:</strong> CI ACCESS cannot be used if the data set is compressed.</td>
</tr>
<tr>
<td>Prefix record with XRBA value</td>
<td>If selected, specifies that PowerExchange should return the extended relative byte address (XRBA) value for all records read from the VSAM data set. <strong>Note:</strong> You must include a 8-byte binary field at the beginning of the record to contain the XRBA value.</td>
</tr>
<tr>
<td>File List Processing</td>
<td>If selected, specifies that PowerExchange should use file list processing for this data map. For more information about file list processing, see “File List Processing” on page 97.</td>
</tr>
<tr>
<td>Data Codepage</td>
<td>Specifies the code page for the data. For more information about code pages in data maps, see “Data Map Code Pages” on page 63.</td>
</tr>
<tr>
<td>Skip First Records from File</td>
<td>Requests PowerExchange to skip the number of records specified before returning data. Default is 0.</td>
</tr>
</tbody>
</table>

The **KSDS Access Method** dialog box has the following fields:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>File Name</td>
<td>Specifies the fully-qualified data set name of the VSAM data set.</td>
</tr>
<tr>
<td>Number of Data Buffers</td>
<td>Specifies the number of I/O buffers for data control intervals that PowerExchange requests VSAM to allocate. Default is 2.</td>
</tr>
<tr>
<td>Number of Index Buffers</td>
<td>Specifies the number of I/O buffers for index control intervals that PowerExchange requests VSAM to allocate. Default is 1.</td>
</tr>
</tbody>
</table>
| Offload Processing     | Specifies whether PowerExchange should off load the conversion and merging of data records from the MVS system where the PowerExchange Listener runs to the system initiating the read request. Select one of the following options:  
  - **Default.** The OFFLOADPROCESSING parameter of the DBMOVER configuration file on MVS controls whether off load processing occurs or not.  
  - **Automatic.** PowerExchange uses off load processing if the data map contains variable length fields.  
  - **Yes.** PowerExchange always uses off load processing.  
  - **No.** PowerExchange never uses off load processing.  
  For more information on off load processing, see “Using Offload Processing for VSAM Data Sets” on page 57. |
| CI ACCESS              | If selected, specifies that PowerExchange should access the entire contents of a control interval rather than individual data records when reading VSAM data sets. This option improves read performance. For more information, see “Improving Bulk Read Performance for VSAM Data Sets” on page 55. **Note:** CI ACCESS cannot be used if the data set is compressed. |
The **RRDS Access Method** dialog box has the following fields:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Codepage</td>
<td>Specifies the code page for the data. For more information about code pages in data maps, see “Data Map Code Pages” on page 63.</td>
</tr>
<tr>
<td>Skip First Records from File</td>
<td>Requests PowerExchange to skip the number of records specified before returning data. Default is 0.</td>
</tr>
</tbody>
</table>

In the **File Name** field, enter the data set name for the VSAM data set.

By default, PowerExchange uses the system-defined buffering for VSAM processing. To tune PowerExchange processing, specify additional VSAM buffers.

For KSDS data sets, off load processing can be selected to off load the process of converting and merging data records from MVS to the extraction system. Generally, the extraction system is the PowerCenter Integration Service machine.

For more information about buffer selection, see “Improving Bulk Read Performance for VSAM Data Sets” on page 55. For more information about offload processing, see “Using Offload Processing for VSAM Data Sets” on page 57.

5. **Click Finish.**
The Import Copybook - Source Details dialog box displays. This dialog box has the following fields:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source</td>
<td>Specifies the location of the copybook or metadata information. Select Local if the metadata is available on this Windows machine or Remote if the metadata is remote from this Windows machine.</td>
</tr>
<tr>
<td>Source Type</td>
<td>Specifies the format of the copybook. Select one of the following types: - COBOL. COBOL copybook - PL/1. PL/I copybook Default is COBOL.</td>
</tr>
<tr>
<td>Column Range</td>
<td>Defines the start and end columns that should be inspected. Typically this should be set to 7-72 for COBOL copybooks and 1-72 for PL/I copybooks. Available range is 1-999.</td>
</tr>
</tbody>
</table>

6. Click Next.

The Import Copybook - Remote Details dialog box displays, which has the following fields:

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>File Name</td>
<td>The fully-qualified MVS data set name, including member name in parentheses if a PDS, that contains the copybook source.</td>
</tr>
<tr>
<td>Location</td>
<td>The node name of the MVS system as defined in the dbmover.cfg file.</td>
</tr>
<tr>
<td>UserId/Password</td>
<td>If PowerExchange is configured for security, a valid MVS user ID and password.</td>
</tr>
<tr>
<td>Save File Locally As:</td>
<td>File into which the copybook will be saved locally.</td>
</tr>
<tr>
<td>Preview</td>
<td>Click this button to preview the copybook source.</td>
</tr>
</tbody>
</table>

7. Click Next. The Import Copybook - Configuration Details dialog box displays. Only options related to the copybook type are enabled. Configure the options you want to use and click Finish.

8. The Record Definition dialog box displays with the VSAM record name in the Name field.

9. Click OK. The data map definition is now complete.

10. Close and save the new data map.

Improving Bulk Read Performance for VSAM Data Sets

PowerExchange provides the following options to improve the performance of bulk read operations for VSAM data sets:

- Specify the number of data and index buffers to use when reading data.
- Specify control interval access.
- Use keys in WHERE clauses.

Number of Data Buffers

You can specify the number of I/O buffers VSAM is to use for transmitting data between virtual and auxiliary storage. A buffer is the size of a control interval in the data component. Number of Data Buffers (BUFND) must be a number between 0 and 65535. The minimum number you can specify is 1 plus the number specified for STRNO. If you omit STRNO, BUFND must be at least 2, because the default for STRNO is 1.

Note: The minimum buffer specification does not provide optimum sequential processing performance. Generally, the more data buffers that are specified, the better the performance. Additional data buffers can benefit direct inserts or updates during control area splits and benefits spanned record accessing.
The maximum number of buffers allowed is 255 (254 data buffers and 1 insert buffer).

### Number of Index Buffers

Specifies the number of I/O buffers VSAM is to use for transmitting the contents of index entries between virtual and auxiliary storage for keyed access. A buffer is the size of a control interval in the index. Number of Index Buffers (BUFNI) must be a number between 0 and 65535. The minimum number is the number specified for STRNO (if you omit STRNO, BUFNI must be at least 1, because the default for STRNO is 1). The default is the minimum number required.

Additional index buffers improve performance by providing for the residency of some, or all, of the high-level index, thereby minimizing the number of high-level index records retrieved from DASD for key-direct processing.

The default is the minimum number required. The maximum number of buffers allowed is currently 255 (254 data buffers and 1 insert buffer).

The number of buffers used can be altered at a PowerExchange system level by entering a VSAM parameter line in the DBMOVER configuration PDS member which is documented in the PowerExchange Reference Manual. If this is coded it will override any entry in the data map.

### Using Control Interval Access

Control interval (CI) access is a read option that causes the VSAM access method to use direct control interval reads to retrieve data records. If you extract all of the data from the VSAM data set, this option may improve the sequential read performance. If you specify a WHERE clause when retrieving the data, CI access provides no performance benefit.

To select at the data map level, select the CI ACCESS option on the Access Method tab of the data map. The CI ACCESS option is valid for KSDS, ESDS, and RRDS access methods.

CI access is not supported for compressed VSAM data sets or VSAM data sets with spanned records.

### Using VSAM Keys

PowerExchange reads the key information for VSAM data sets from the catalog when the data set is allocated and opened. Both the key length and the relative key position are retrieved.

For a record layout (keypos=10, keylen=4) similar to

```
01 REC.
  04 HEADER     PIC X(10).
  04 KEY.
    08 KEY1     PIC 99.
    08 KEY2     PIC 99.
  04 REST          PIC X(200).
```

**Note:**

* KEY1 must be in the WHERE clause for any optimization to work.
* KEY2 will be used if KEY1 is also in the WHERE clause.

The key information is used to speed up the processing of WHERE clauses. This is called optimization. It is important to understand the syntax of the WHERE clause, as different formats will produce different results. For example:

Line 1: `SELECT * FROM TEST.VSAM1.TAB WHERE (KEY1 > 50) AND (KEY1 < 90)`

will generate the same results as

Line 2: `SELECT * FROM TEST.VSAM1.TAB WHERE (KEY1 > 50  AND  KEY1 < 90)`

but with different elapsed time.

Line 1 gets translated into `(50 < KEY1 < HIGH-HALUES) AND (LOW-VALUES < KEY1 < 90)` When these two are put together the result is `(KEY1 > LOW-VALUES AND KEY1 < HIGH-HALUES)- the entire file.`
This is still technically optimized, but when the file is processed the read start is positioned at LOW-VALUES and read until HIGH-VALUES - not ideal.

Line 2 is the preferred option. The line 2 syntax will position the file at record 50 and read from there until KEY1 \( \geq 90 \).

Optimization is used for key positioning only, the data is checked against the WHERE clause to ensure the correct records are filtered.

**Note:** If skip records > 0 is specified (using the Skip First 'n' Records from File field on the KSDS Access Method Wizard), the SELECT statement will NOT be optimized even if KEY1 is specified.

### Using Offload Processing for VSAM Data Sets

If the PowerExchange Listener is running on a CPU constrained host, the process of converting and merging data records can be offloaded from the PowerExchange Listener to the client. For certain maps (especially those containing variable fields), the savings can be upwards of 90% with only a small increase in network traffic.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yes</td>
<td>Always offload processing where possible.</td>
</tr>
<tr>
<td>No</td>
<td>Never offload processing.</td>
</tr>
<tr>
<td>Automatic</td>
<td>Allows PowerExchange to optimize the location.</td>
</tr>
<tr>
<td>Default</td>
<td>Defer to the OFFLOADPROCESSING parameter in the dbmover.cfg file.</td>
</tr>
</tbody>
</table>

A message will appear in the log if data map processing has been offloaded.

**Note:** If parameterized SQL is used against this data map, offload processing will be ignored.

### Retrieving the RRN or RBA for Records in VSAM Data Sets

PowerExchange can return the Relative Record Number (RRN) or Relative Byte Address (RBA) for ESDS and RRDS data sets in the following ways:

- Prefix records with the RRN or XRBA value when you select these options in the data map properties
- Include the RRN or RBA value in a new field in the record when you use the GetDatabasekey expression

**Note:** You cannot use the GetDatabasekey expression if you request off load processing.

### Using Data Map Options to Retrieve the RRN or RBA

You can use options in the data map to retrieve the relative record number (RRN) or relative byte address (RBA) for VSAM data sets.

**To retrieve the RRN or RBA:**

1. Open the data map.
2. Right-click on the data map name, and then select **Properties**.
3. The **Data Map Properties** dialog box displays. Click the **Access Method** tab.
4. On the Access Method tab, select **Prefix record with XRBA** to retrieve the RBA for ESDS data sets or **Prefix Record with RRN** to retrieve the RRN for RRDS data sets.
5. If you request that PowerExchange return the RRN or RBA value, you must add a field at the beginning of the record definition to hold this value.
   - Click the record in the data map to open the record layout dialog box.
6. Right-click on the first field in the record, and then select **Add Field Before**.
7. Enter a value for **Field Name**. Select **BIN** for the **Field Type** and **8** for the **Length** field.
Click OK.

8. Click on the table to open the table layout dialog box.

9. Right-click on the first column in the table, and then select Add Column Before.

10. Enter a name for the new column in the Name field, and then select the new field from the list in Base Field.

11. Click File > Send to Remote Node to send the altered data map to the MVS system where the VSAM data set resides.

12. Close the data map.

Using the GetDatabaseKey Expression to Retrieve the RRN or RBA

You can use the GetDatabaseKey expression to retrieve the relative record number (RRN) or relative byte address (RBA) for VSAM data sets. You cannot use the GetDatabaseKey expression if you use off load processing.

You specify expressions in the record definition of the data map and create user-defined fields to hold the results of the expression.

To retrieve the RRN or RBA using expressions:

1. Open the data map and click on the record to open it.

2. Click the Expr tab to enter the required expression.

3. Right-click, and then click Add Field at End.

4. Enter the following information:
   - A name in the Name column
   - BIN in the Type column
   - 8 in the Length column
   - GetDatabaseKey() in the Expression column

   When the expression field is valid, the PowerExchange Navigator displays a green check mark in the leftmost column. Otherwise, the PowerExchange Navigator display a red cross in that column and a message displays in the Validation column.

5. After the field has been added to the record, it needs to be included in the table representation. Click the table to open it.

6. Right-click the table name, and then click Add Column at End.

7. Select the new column from the list in Base Field, specify a name for the column in the Name field, and then click OK.

8. Click File > Send to Remote Node to send the altered data map to the MVS system where the VSAM data set resides.

9. Close the data map.

Testing VSAM Data Maps

To test the definition, use Database Row Test. The database row test options retrieves data from the selected VSAM data set. For more information about Database Row Test, see “Using Database Row Test” on page 107.
Creating C-ISAM Data Maps

To access C-ISAM data using PowerExchange, the structure of the file, tables, and columns must be known, so that you can define a data map.

Step 1. Data Map Name
1. The main data map Name dialog box appears.
2. Enter a Schema Name, such as ISAM, and a Data Map Name, such as map1.
3. The Access Method is important. For C-ISAM access, you need to select ISAM.
4. The Import Record Definitions box should be disabled, as the record definitions will be specified later.
5. Click Next.

Step 2. Access Method
1. The ISAM Access Method dialog box is now displayed.
2. Enter the full path and file name of the C-ISAM data file that you wish to map and the click Finish.

   Notice how a Data Map tab has appeared at the bottom left. Also the name of any data map open is shown in the Resources dialog box.

Step 3. Adding a Record
The next task is to add a record.
1. Right-click the required data map and click Add Record. A blank Add Record dialog box appears. The name is for reference when building tables.
2. Enter the required record name, such as record.
3. Click OK. The Record panel appears.

Step 4. Adding a Field
There are three fields in the sample data file that must be added (seqno, name and gender).
Each field needs to be added using the toolbar Add Field icon, or by clicking Add > Field. Complete the following tasks for each field:
♦ Enter a Field Name.
♦ Select a Field Type.

The details for each field that is to be added are as follows:

Add the ID field (which is a number field):
1. Select a Field Type of NUMCHAR (numeric character).
2. Enter the number of digits of Precision (2) and the Length (2) field. The Length field must be equal or greater than the Precision. This tells PowerExchange the maximum size of the input, allowing for leading blanks or other characters.
3. Click OK. The field is added to the record display.

Add the Reference field (which is a character field):
1. Select a Field Type of CHAR (character).
2. Enter the number of digits of the Length (3) field.
3. Click OK.
Add the Comment field (which is a character field):
1. Select a Field Type of CHAR (character).
2. Enter the number of digits of the Length (100) field.
3. Click OK.
4. After all the fields are complete, the record display looks like the following:

```
<table>
<thead>
<tr>
<th>ID</th>
<th>recordID</th>
<th>Num</th>
<th>2.02</th>
<th>0.640</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ref</td>
<td>Reference</td>
<td>Ch</td>
<td>3</td>
<td>0.2 ± 0.64</td>
</tr>
<tr>
<td>Com</td>
<td>Comment</td>
<td>CHAR</td>
<td>(100)</td>
<td>45 ± 0.00</td>
</tr>
</tbody>
</table>
```

Having completed the record we can now create a table.

Step 5. Adding a Table
1. Right-click the data map name on the left of the dialog box, and then click Add Table. This displays the Add Table dialog box. For this example, enter a Table Name of table1.
   
   In this example all the data from an input record is going to make up a row.
2. Click OK. The table will be created using all the fields in record.

This data map is now complete. It can be used to test our data source immediately. In this example we are not going to connect to a remote PowerExchange Listener, but go directly to the data file locally.

Step 6. Row Test
1. Use the toolbar Row Test icon or the click File > Database Row Test to start the process. A message box appears asking you to confirm that you want to send the data map to a remote location.
2. Click Yes.
3. Make sure the Location is set to local as below:
Creating MQSeries Data Maps

To move around data using MQSeries functionality, you must know the structure of the file, tables, and columns so that you can define a data map.

Typically the MQ data map must be column mapped to the data file that you wish to move with it. For example, we may be moving a VSAM file that has been mapped using a VSAM data map. We must ensure that the VSAM data map and this MQ data map are matched.

Step 1. Data Map Name

1. The main data map Name dialog box appears.
2. Enter a Schema Name, such as mq1 and a Data Map Name, such as map1.
3. The Access Method is important. For MQSeries access you need to select MQSERIES.
4. The Import Record Definitions box should be disabled, as the record definitions will be specified later.
5. Click Next.

Step 2. Access Method

1. The MQSeries Access Method dialog box is now displayed.
2. Enter the name of the MQSeries Queue Manager and the queue itself that you wish to map and the click Finish.

Notice how a Data Map tab has appeared at the bottom left. Also the name of any data map open is shown in the Resources dialog box.

Step 3. Adding the Records

The next task is to add the records that match the data file to be moved.

1. Select Properties.

Certain field formats are supplied including several date and timestamp formats. The timestamp selected above redefines the 26 byte character field. The formats can be edited, however, and if the data is delivered
without hyphens between the year, month and day delete these from the field format to match the field length and format found in the file.

2. Right-click the required data map and click Add Record. A blank Add Record dialog box appears. The name is for reference when building tables.

3. Enter the required record name, such as record.

4. Click OK. The Record panel appears.

Step 4. Adding a Field

For example there may be three fields in the sample data file that must be added (seqno, name and gender).

Each field needs to be added using either the toolbar Add Field icon or through the menu option Add, Field. Complete the following tasks for each field:

- Select a Field Type.

The details for each field that is to be added are as follows:

Add the ID field (which is a number field):

1. Select a Field Type of NUMCHAR (numeric character).

2. Enter the number of digits of Precision (2) and the Length (2) field. The Length field must be equal or greater than the Precision. This tells PowerExchange the maximum size of the input, allowing for leading blanks or other characters.

3. Click OK. The field is added to the record display.

Add the Reference field (which is a character field):

1. Select a Field Type of CHAR (character).

2. Enter the number of digits of the Length (3) field.

3. Click OK.

Add the Comment field (which is a character field):

1. Select a Field Type of CHAR (character).

2. Enter the number of digits of the Length (100) field.

3. Click OK. Having completed the record we can now create a table.

Step 5. Adding a Table

1. Right-click the Data Map name on the left hand side of the dialog box and then click Add Table. This displays the Add Table dialog box. For this example, enter a Table Name of table1.

   In this example all the data from an input record is going to make up a row.

2. Click OK. The table will be created using all the fields in record.

   This data map is now complete. It can be used to move data from its source to an MQ Series queue.

3. It is important now that the data map that you have created is saved. Select File > Close Resource. After closed, you will now see mq.map1 under the Data Maps resource folder.
Data Map Code Pages

PowerExchange supports a wide range of code pages. Extended single and multi-byte support is available for DB2 on MVS and i5/OS, and NRDB access to VSAM and sequential files. To facilitate extended single byte and multi-byte support PowerExchange includes ICU software. Where the code pages pre-supplied with PowerExchange are not sufficient, the facility exists for the user to supply external code pages. Code page can be set at both the data map and field level.

The code page option is only active when the Encoding is explicitly defined as EBCDIC, ASCII HIEND and ASCII LOEND.

To specify a different code page at a field level, select a data map, open the record, then select properties of the required field.

The code page for the selected field can then be specified on the following screen:

Code page specified at the field level will take precedence over anything specified at the data map level.

PowerExchange Code Pages

The standard code pages included within PowerExchange are listed below. Code pages beginning with IBM- are for handling data from MVS and i5/OS systems only:

<table>
<thead>
<tr>
<th>Code Page</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ISO-8859</td>
<td>Typically UNIX and Windows</td>
</tr>
<tr>
<td>IBM-037</td>
<td>USA, Canada, Brazil</td>
</tr>
<tr>
<td>IBM-273</td>
<td>Germany, Austria</td>
</tr>
<tr>
<td>IBM-277</td>
<td>Denmark, Norway</td>
</tr>
</tbody>
</table>
Default code pages are assumed for each platform running PowerExchange as follows:

<table>
<thead>
<tr>
<th>IBM-278</th>
<th>Finland, Sweden</th>
</tr>
</thead>
<tbody>
<tr>
<td>IBM-280</td>
<td>Italy</td>
</tr>
<tr>
<td>IBM-284</td>
<td>Spain, Latin America</td>
</tr>
<tr>
<td>IBM-285</td>
<td>United Kingdom</td>
</tr>
<tr>
<td>IBM-297</td>
<td>France</td>
</tr>
<tr>
<td>IBM-424</td>
<td>Modern Hebrew</td>
</tr>
<tr>
<td>IBM-500</td>
<td>International</td>
</tr>
<tr>
<td>IBM-1047</td>
<td>Latin 1/Open Systems</td>
</tr>
</tbody>
</table>

Up to 10 USRC Pnn can be set up where nn is a value between 00 and 09. User code pages are ones that a user has defined to display or print given hex values.

For a full explanation of using the CODEPAGES, and how to set up user-defined pages, see the PowerExchange Reference Manual.
This chapter includes the following topics:

♦ Importing COBOL Copybooks, 65
♦ Importing COBOL Copybooks Containing Redefines, 71
♦ Importing a PL/1 Copybook, 76
♦ Importing a DDS Copybook, 81
♦ Importing Copybooks for IMS Data Maps, 85
♦ Creating Records from Multiple Copybooks, 89

Importing COBOL Copybooks

This chapter is intended to demonstrate how to import a COBOL Copybook through the PowerExchange Navigator and then source actual data. It contains the complete sequence of dialog boxes with the related text.

The key features demonstrated in this section are as follows:

♦ The data file is fixed-length records, 60 bytes long.
♦ Import of a COBOL copybook.
♦ Multiple record types, 01 for COBOL NAME_REC, 02 for COBOL ACCOUNT_REC.
♦ A hierarchical view of a flat file.
♦ An OCCURS depending on.
♦ The ability to generate multiple output rows from a single input row.
♦ Using date masking of COBOL POLICY_DATE.
♦ Use of a where clause to eliminate Donald Duck record.

Data File Input

File name: train3.dat, fixed 60 bytes long

00501Mickey Mouse  M3apple orange pear
0050200000012345012-31-87
01001Shirley Temple  M1raspberry
01002000000025607-04-57
02001John Wayne  M2pansy daisy
03001Donald Duck  M0
03002000000004501-12-66
04001Goofy  M1fox
Data File Input Viewed via Copybook

<table>
<thead>
<tr>
<th>No.</th>
<th>Name</th>
<th>Sex</th>
<th>Item</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>005</td>
<td>Mickey Mouse</td>
<td>M</td>
<td>3</td>
<td>apple orange</td>
</tr>
<tr>
<td>005</td>
<td>02 0000123450 12-31-87</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>010</td>
<td>Shirley Temple</td>
<td>F</td>
<td>1</td>
<td>raspberry</td>
</tr>
<tr>
<td>010</td>
<td>02 00000000256 07-04-57</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>020</td>
<td>John Wayne</td>
<td>M</td>
<td>2</td>
<td>pansy daisy</td>
</tr>
<tr>
<td>030</td>
<td>Donald Duck</td>
<td>M</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>030</td>
<td>02 0000000045 01-12-66</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>040</td>
<td>Goofy</td>
<td>M</td>
<td>1</td>
<td>fox</td>
</tr>
<tr>
<td>050</td>
<td>Greta Garbo</td>
<td>F</td>
<td>3</td>
<td>dog cat rabbit</td>
</tr>
<tr>
<td>050</td>
<td>02 0000000911 02-29-96</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>060</td>
<td>Ronald Reagan</td>
<td>M</td>
<td>2</td>
<td>horse pony</td>
</tr>
<tr>
<td>060</td>
<td>02 0000100000 01-01-01</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>070</td>
<td>Bette Midler</td>
<td>F</td>
<td>1</td>
<td>wolf</td>
</tr>
<tr>
<td>070</td>
<td>02 0000640001 12-07-41</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

COBOL Copybook with OCCURS

```
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).
```

Output Row to be Constructed:

- account
- name
- sex
- amount
- policy date as a formatted date

Loading a COBOL Copybook

To load a COBOL copybook is a simple process of following the wizard provided. A copybook can be imported into an existing data map or a new one.

**Step 1. Add a Data Map**

1. Use the toolbar Add Data Map icon or right-click Data Maps and click **Add Data Map**.
2. Enter a Schema Name of xxxx and a Data Map Name of map2.
3. Select the Import Record Definitions (IRD) option. (This should be the default value). The process will be invoked automatically as part of the data map build.
Step 2. File Name and Record Format

1. On the SEQ Access Method dialog box, browse through the supplied examples directory for the file name \texttt{train3.dat} and open the file. For example: \texttt{C:\Program Files\Informatica\Informatica PowerExchange\examples\train3.dat}.

2. Select \texttt{Fixed}.

3. Enter a Size of 60.

4. Leave the Encoding as Default.

5. Click \texttt{Finish}.

Step 3. Import Copybook Wizard

The Import Copybook wizard opens automatically.

\textbf{To complete information in the Import Copybook wizard:}

1. In Source, select \texttt{Local}.

2. In Source, make sure that the dialog box shows COBOL in the Type field.

3. In Column Range, define the Start and End columns to be inspected. Typically, set these to 7 and 72 for IBM copybooks, but other ones might have data outside this range. The value entered for the Start column is recognized as the last comment column. The values should show 7 and 72.

4. Click \texttt{Next} to request the copybook file name. Browse for \texttt{train3.cob} in the examples directory and open it. For example: \texttt{C:\Program Files\Informatica\Informatica PowerExchange\examples\train3.cob}.

5. Click the \texttt{Preview} button to view the copybook file being imported. The Field Usage, Level, Original Name, and Picture from the Cobol or PL1 copybook are visible after the import has completed in the Extra Properties tab.

6. Close the Preview dialog box if viewed.

7. To display the Import Copybook - Configuration Details dialog box, click \texttt{Next}.

8. To display the Import Copybook Information dialog box, click \texttt{Finish}.

\textbf{Note:} The Import Copybook Configuration Details dialog box enables you to change the prompt parameters. To skip this dialog box, click \texttt{Finish} on the Import Copybook - Local COBOL Details dialog box. However, view this dialog box to confirm selected options.

The Import Copybook Information dialog box enables you to confirm the selected copybook options. If these are incorrect you can click \texttt{Cancel} to go back and reset them.
9. If the settings are correct, click **OK**. The Record Definition dialog box appears for the two records to be imported. The record names as the same as the ones in the COBOL copybook.

   ♦ Select **Skip** to skip a record type.
   ♦ Select **Stop Import** to terminate the import.

10. In this example, because you want to import both record types, accept the defaults and click **OK** on both dialog boxes.

    The Copybook import dialog box appears.

    An Error/Log appears at the end of the import showing the outcome.

11. After successfully completing this task, close the Copybook Import dialog box.

    **Note:** If there is an error in the Copybook Message Log, you can double-click the message to display the relevant line number in the Cobol Import window.

### Step 4. Change Record ID Value

1. On the Data Map tab, click **NAME_REC**.

2. Right-click the **RECTYPE** field and click **Properties**.

3. Click **Record ID Values** and enter 01.

4. Click **OK**.

5. Repeat the process for **ACCOUNT_REC** with a Record ID Value of 02.

    **Note:** To review the original COBOL or PL1 field attributes from the Cobol copybook click the Extra Properties tab.

6. Click **OK**. Both **RECTYPE** field icons should be green.

### Step 5. Add Table

1. Right-click map2, and then click **Add Table**.

2. Enter a Table Name of both.

3. Right-click **NAME_REC** in the left dialog box and click **Add Record**.

4. Right-click **ACCOUNT_REC** in left dialog box and click **Add Record as Child**.

5. After creating the hierarchical definition, click **OK**.
Step 6. Row Test

To retrieve data from the data file:
1. Click the Row Test icon.
   A message box indicates that the data map needs to be sent to a remote node.
2. Select **Don't warn** so that the prompt is not seen again.
3. Click **OK**.
   The Data Map Remote Node dialog box appears.
4. Make sure the Location is set to local.
5. Click **OK**.
   The Database Row Test dialog box appears.
6. Click **Go**.
   The data is retrieved from the data file and is displayed.

Step 7. Delete Unwanted Fields

1. Close the Database Row Test dialog box.
2. Click the both table and delete the RECTYPE, ACCOUNT_REC_RECTYPE and ACCOUNT_REC_ACCOUNT fields.
3. To delete a field, right-click the appropriate field and click **Delete**.

Step 8. Add Field Format for Policy Date

**Note:** Be aware of the following:
- The POLICY_DATE shows as an unformatted field, which you can format.
- The FILLER fields are normally excluded by default. However, in this example, you must delete them manually.

1. Click the record ACCOUNT_REC.
2. Right-click POLICY_DATE field and click **Properties**.
   The Field Properties dialog box appears.
3. Click the arrow button to see the format options for Field Format.
4. Select Y2-M M-D 2 (note no spaces in mask).
   **Note:** The both table POLICY_DATE field type has changed from CHAR(8) to DATE.
5. Click the both table and then proceed with another Row Test.
6. On the Database Row Test dialog box, click **Go**.
   The following actions occur:
The century has now been added to the date.
The hyphens within the POLICY_DATE data are for display purposes only.
Donald Duck has zero items. Therefore, a where clause can be used to remove it.

Step 9. Use Where Clause to Eliminate Donald Duck

1. Add where itemct > 0 to SQL statement.
   It should read:
   ```sql
   select * from xxxx.map2.both where itemct > 0
   ```

2. On the Database Row Test dialog box, click **Go**.
   The following actions occur:
   - Only seven rows are displayed.
   - There are several NULL values in ITEMS field.
   - See below for what happens if you only want a single ITEM per output row (OCCURS DEPENDING ON).

Step 10. Single Item per Output Row

1. Close the Database Row Test dialog box.
2. Right-click the both table and click **Properties**.
   The Table Properties dialog box appears.
3. Check the ITEMS box as shown below.

4. Click **OK**.
5. Perform a Row Test. On the Database Row Test dialog box, click Go.
   
   Note: There are now three records for Mickey Mouse with a single ITEM per output.

6. Close the Database Row Test dialog box.

7. Click File > Close Resource.

Importing COBOL Copybooks Containing Redefines

This chapter is intended to follow “Importing Copybooks” on page 65, teaching you how to import a more complex COBOL Copybook through the PowerExchange Navigator and then to source the data. The input data source is an EBCDIC file, simulating data sourced from an IBM mainframe.

The key features demonstrated in this section are as follows:
- A fixed-length binary data file of EBCDIC data, 57 byte records, 58320 rows
- COBOL copybook with line numbers in columns 1-6 and comments in columns 73-80
- Redefines in copybook imported
- First record to be ignored
- Date masking on MASTER_DATE field

Data File Input

File name: train5.dat - This contains EBCDIC data after a single ASCII record.

COBOL Copybook

File name: train5.cob

00001 * TRAINS EXAMPLE COBOL COPYBOOK
00002 01 MASTER_REC.
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 * REDefines of binary field to character
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 PIC S99 (8) COMP.  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 ACCT-CODES PIC X(10) OCCURS 3.  COL 73-80

Note: The comment lines in the copybook are treated as such and are ignored by PowerExchange.
Output Row to be Constructed:
- account
- amount
- bin_no
- master date as single date column
- single row per account code

Loading the Copybook

Note: The skip records feature can be used, for example, to skip header records, which do not contain data. Range of this value is 0 to 2147483647.

Step 1. Add Data Map
1. Right-click Data Maps and click Add Data Map.
2. Enter a Schema Name of xxxx and a Data Map Name of map3.
3. Ensure the Import Record Definitions box is checked.
4. Click Next.

Step 2. File Name and Record Format
1. In the File Name box, browse for train5.dat and open it.
2. Select Fixed.
3. Enter a record Size of 57.
4. For Encoding select EBCDIC.
5. Leave the Codepage as Default.
6. Skip First Records from File must be 1 for this example. Your SEQ Access Method dialog box should look like below.

![SEQ Access Method Dialog Box](image)

7. Click Finish.

Step 3. Import Copybook
The Import Copybook dialog box appears.
1. In Source, select Local.
2. In Source, make sure that the dialog box shows COBOL in the Type field.
   The source is the node name where the copybook exists. PowerExchange can for instance go directly to MVS and retrieve the copybook from a PDS.
3. In Column Range define the Start and End columns that should be inspected. Typically these should be set to 7 and 72 for IBM copybooks, but other ones might have data outside this range. It should be stressed that the value entered for the Start column will be deemed to be the last comment column. The values should show 7 and 72.
4. Click Next.
5. In the File Name box, browse for train5.cob and open it.
6. Click Next.
7. Click Finish on the Configuration Details dialog box.
8. Click OK on the Import Copybook Information dialog box.

Step 4. Select Definition
1. On the Record Definition dialog box, click OK on the MASTER_REC. The Copybook Import dialog box appears.
2. Use the Next Redefinition icon and Previous Redefinition icon on the toolbar to move the blue arrow on left to point at the line that you want to select.
3. Point at the PIC S9(8) field and click the Resume Import icon to resume the import. The following Error/Log Message should be displayed.
4. The first two lines in the log refer to record numbers in the Copybook Import dialog box above.
5. Close the Copybook Import dialog box.

**Step 5. Create Single Date Column**

1. Click the Data Map tab. Click MASTER_REC record, and click MASTER_DATE field.
   
   **Note:** The COBOL copybook has a GROUP field and three individual ZONED fields, DATE_YY, DATE_MM and DATE_DD. However, what is required on output is a single DATE column from this GROUP structure.

2. Highlight the three individual date fields. Click the DATE_YY field then hold down CTRL key and select the other two, see below.

3. Click the Delete icon on the toolbar.
4. Click More Details to display field details.

5. Click Yes to the PowerExchange Navigator prompt.

**Step 6. Change Field Properties**

1. Right-click the MASTER_DATE field and click Properties.
2. Change the Field Type to CHAR, and Length to 6.
3. For the Field Format, enter Y2M D2 or select Y2-M M-D2 from the Field Mask and delete the hyphens.
   
   **Note:** Do not select the Field Mask without deleting the hyphens otherwise you will increase the number of characters to 8.
The completed Field Properties dialog box appears.

4. Click OK. The MASTER_REC record will now look like this.

Step 7. Add Column
1. Click MASTER_REC table.
2. Click the Add Column icon.
3. On the Add Column dialog box, enter a Name of new_date.
4. Select MASTER_REC:MASTER_DATE from the Base Field pull-down list.
5. Click OK.
Step 8 - Row Test

1. Click MASTER_REC table and click the Row Test icon.
2. On the Database Row Test dialog box click Go. After performing the Row Test, the results appear:

   ![Database Row Test dialog box]

   - Close the Database Row Test dialog box.
   - Click File > Close Resource.

---

**Importing a PL/1 Copybook**

**Introduction**

In this chapter a PL/1 Copybook will be imported instead of a COBOL Copybook. The Access Method used will be (SEQ). The data file being sourced is train5.dat and it was used in “Importing COBOL Copybooks Containing Redefines” on page 71 so this is intended to purely show the PL/1 import.

**Data File Input**

File name: train5.dat - This contains EBCDIC data after a single ASCII record. The only record that will display properly with a text editor is the ASCII record which reads: This is a bogus first record which should be skipped.

**PL/1 Loading**

File name: train5.pl1

```pl1
DECLARE 1 MASTER_REC,
  2 ACCOUNT                     CHAR(9),
  2 REC_TYPE                    CHAR(1),
  2 AMOUNT                      FIXED DECIMAL(6,2),
  2 BIN_NO UNION,
  3 BIN_NO_N                 FIXED BINARY(16),
  /* REDIFINES OF BINARY FIELD TO CHARACTER */
  3 BIN_NO_X                 CHAR(4),
  2 DECIMAL-NO                  PIC "S999",
  2 MASTER-DATE,
  3 DATE-YY            PIC "999",
```
Procedure

Note: A PL/1 Copybook can be a substitute for a COBOL copybook.

Step 1. Add Data Map

When naming the new Data Map, ensure that the Access Method is SEQ.

Enter a Schema Name of xxxx, and a Data Map Name of map5. The Import Record Definitions box must be checked. The completed dialog box is shown below.

Step 2. File Name

1. Browse for the File name of train5.dat.

2. Select Fixed.

3. Enter a Size of 57.

4. The Encoding should be EBCDIC.

5. Click Finish.
Step 3. Import Copybook

1. The Import Copybook wizard opens automatically.

2. In Source, select Local. This is the node name where the member exists.

3. In Source, select PL/1 from the Type pull-down box. For PL/1, the column range automatically defaults to 1 and 72.

4. Click Next.

5. The file name of the PL/1 copybook will now be requested.


7. Click Next.
Step 4. Configuration Details

1. A dialog box requesting what actions to be taken when loading the copybook appears.

2. Click **Finish** on the Configuration Details dialog box.

3. Click **OK** on the Import Copybook Information dialog box.

Step 5. Record Definition Dialog Box

1. On the Record Definition dialog box, click **OK** for the MASTER_REC. The next screen display is shown below.

   This dialog box could have been suppressed if the Prompt on record import box had been disabled. Again the options should be obvious, accepting the record, skipping this record, which will cause all entries to be skipped until the next Declare 1 statement is found, or the import stopped.
Step 6. Select Redefinition Line

1. There is a redefinition and the screen following appears, with the redefinitions in red.

2. Use Next Redefinition icon and Previous Redefinition icon on the toolbar to move the blue arrow on left to point at the line that you want to select.

3. Point the blue arrow at the line containing the Fixed Binary (16) field, which is line 6 and click the Resume Import icon to resume the import. The following Error/Log Message should be displayed.

![Copybook Redefines]

**Note:** The first two lines in the log refer to record numbers in the PL/1 Import dialog box above.

4. Close the PL/1 Import dialog box.

Step 7. View Record and Table

1. Click MASTER_REC record and table to view the fields and columns that have been imported. It is interesting to compare the field and table structures imported, relative to the COBOL Copybook import.
The contents should be effectively identical. The fact that the COBOL version shows BIN_NO as a group item has no impact.

Thereafter, the data can be sourced in the normal way. After the import is complete, the data map is identical to one created by any other means.


Importing a DDS Copybook

Introduction

In this chapter a DDS Copybook will be imported instead of a COBOL or PL/1 Copybook. The Access Method used will be (SEQ).

Procedure

Note: A DDS Copybook can be a substitute for a COBOL or PL/1 copybook.

Step 1. Add Data Map

1. Right-click Data Maps and click Add Data Map.
2. When naming the new Data Map, ensure that the Access Method is SEQ.
3. Enter a Schema Name, such as DDS, and a Data Map Name, such as map1. The Import Record Definitions box must be checked. The completed dialog box is shown below.

4. Click Next.

**Step 2. File Name**

1. Enter the file name that you want to map, such STQA/IFITEST1 on an i5/OS computer.
2. Select Default.
3. The Encoding should be Default.

4. Click Finish.

**Step 3. Import Copybook**

1. The Import Copybook wizard opens automatically. The Import Copybook - Source Details dialog box appears.
2. In Source, select Remote.
3. In Source, make sure that the dialog box shows DDS in the Type field.
The source is the node name where the DDS file exists.

4. Click Next.

**Step 4. Import Copybook**

1. The Import Copybook - Remote DDS Details dialog box appears.

2. Enter the appropriate Remote File details.

3. Click Next.
Step 5. Configuration Details

1. The Import Copybook - Configuration Details dialog box appears requesting what actions to be taken when loading the copybook.

The options are very straightforward, requesting prompts in various situations and actions to be taken if duplicate record, fields and tables are detected.

2. Click Finish on the Configuration Details dialog box.
3. Click OK on the Import Copybook Information dialog box.

Step 6. Record Definition Dialog box

1. The next screen display is shown below.

This dialog box could have been suppressed if the Prompt on record import box had been disabled.

The options should be obvious, with you accepting the record or skipping this record, which will cause all entries to be skipped until the next option point is found, until the import is stopped or finishes normally.
importing copybooks for IMS data maps

Importing Copybooks for IMS Data Maps

Import a COBOL copybook for each IMS segment added to a data map. Importing an IMS DBD source adds segments, key fields and search fields from the IMS database and creates tables for a data map. After you import the IMS DBD source, importing a copybook for each segment overlays each segment with its COPYLIB. This redefines the data map while maintaining the hierarchical metadata information for the database.

Before you import a copybook to an IMS data map, import the IMS DBD source. For more information, see “Importing IMS DBD Source for IMS Data Maps” on page 43.

To import a copybook to an IMS data map:

1. In the Data Map tab of the PowerExchange Navigator, select an IMS segment and click File > Import Copybook.

2. Complete the fields on the Import Copybook - Source Details page.
Table 3-1 describes these fields:

Table 3-1. Import Copybook - Source Details Fields

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source</td>
<td>Indicates the copybook is local which means the data has been downloaded and stored locally, or remote which means the data will be downloaded from a remote location such as MVS. For instance, PowerExchange can go directly to MVS and retrieve a copybook from a PDS.</td>
</tr>
<tr>
<td>Type</td>
<td>Select the type of import to perform: COBOL.</td>
</tr>
<tr>
<td>Column Range</td>
<td>Defines the start and end columns to inspect.</td>
</tr>
<tr>
<td>Last Import</td>
<td>Click to show the specification to which the copybook will be imported. The current specifications are those of the previous import.</td>
</tr>
</tbody>
</table>

If you select **Local** as the **Source** option, continue to the next step.

-or-

If you select **Remote** as the **Source** option, go to step Step 4.

3. Complete the fields on the Import Copybook - Local Cobol Details page, and continue to Step 5:

Table 3-2 describes the Import Copybook - Local Cobol Details fields:

Table 3-2. Import Copybook - Local Cobol Details Fields

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>File Name</td>
<td>File name of the copybook to import.</td>
</tr>
<tr>
<td>Preview</td>
<td>Click to preview the copybook.</td>
</tr>
</tbody>
</table>

4. Complete the fields on the Import Copybook - Remote Cobol Details page, and click **Next**.

Table 3-3 describes the Import Copybook - Remote Cobol Details fields:

Table 3-3. Remote Cobol Details Fields

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>File Name</td>
<td>Copybook file name and member.</td>
</tr>
<tr>
<td>Location</td>
<td>Location of the copybook file.</td>
</tr>
<tr>
<td>UserID</td>
<td>MVS user ID if security has been implemented.</td>
</tr>
<tr>
<td>Password</td>
<td>MVS password if security has been implemented.</td>
</tr>
<tr>
<td>Name</td>
<td>Name of the file that the copybook is to be saved as when it is retrieved from the remote platform.</td>
</tr>
<tr>
<td>Name Browse</td>
<td>Click to browse for the file.</td>
</tr>
<tr>
<td>Preview</td>
<td>Click to preview the copybook.</td>
</tr>
</tbody>
</table>

5. Complete the fields on the Import Copybook - Configuration Details page.

Table 3-4 describes the Import Copybook - Configuration Details fields:

Table 3-4. Import Copybook - Configuration Details Fields

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prompt on record import</td>
<td>When a record is created, prompt for a record name showing the first name found.</td>
</tr>
<tr>
<td>Prompt on field import</td>
<td>When a field is created, prompt for a name showing the name found.</td>
</tr>
</tbody>
</table>
6. Click Finish.
7. Review the options you selected for the import.
8. If the information is correct, click OK.
   - If you chose to be prompted during the import, continue to “Using the Import Prompts” on page 87.
   - or -
   - If you did not select prompts to occur, review the data that is imported.

### Using the Import Prompts

When you complete the Import Copybook Options, you indicate whether or not prompts should be provided as each item is imported from a DBD source or copybook.

If you choose to be prompted, use the Record Definition options to complete import prompts and step through each item that is being imported. During this process, you determine which segments to import or skip. You can also specify how to manage duplicate data during a copybook import.

#### To specify items to import:

- Complete the Record Definition options to step through each item as a DBD source or COBOL copybook is imported.
- or -
If you are starting the import from the DBD or Copybook Import view, select Import > Start to access the Record Definition options:

![DBD Import screen](image)

<table>
<thead>
<tr>
<th>Line</th>
<th>Record</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>*</td>
</tr>
<tr>
<td>3</td>
<td>IMSDEMO DBD</td>
</tr>
<tr>
<td>4</td>
<td>EXIT=(*,KEY,P0) FOR INS</td>
</tr>
<tr>
<td>5</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>DDD NAI</td>
</tr>
<tr>
<td>7</td>
<td>RNI</td>
</tr>
<tr>
<td>8</td>
<td>EXI</td>
</tr>
<tr>
<td>9</td>
<td>DSEBB1 DATASET D</td>
</tr>
<tr>
<td>10</td>
<td>SECH NAI</td>
</tr>
<tr>
<td>11</td>
<td>FIELD NAI</td>
</tr>
<tr>
<td>12</td>
<td>FIELD NAI</td>
</tr>
<tr>
<td>13</td>
<td>FIELD NAI</td>
</tr>
<tr>
<td>14</td>
<td>FIELD NAI</td>
</tr>
<tr>
<td>15</td>
<td>FIELD NAI</td>
</tr>
<tr>
<td>16</td>
<td>FIELD NAI</td>
</tr>
<tr>
<td>17</td>
<td>FIELD NAI</td>
</tr>
<tr>
<td>18</td>
<td>FIELD NAI</td>
</tr>
<tr>
<td>19</td>
<td>FIELD NAI</td>
</tr>
<tr>
<td>20</td>
<td>FIELD NAI</td>
</tr>
<tr>
<td>21</td>
<td>FIELD NAME=INDCTR,BYTES=10,START=50</td>
</tr>
<tr>
<td>22</td>
<td>OBJGEN</td>
</tr>
<tr>
<td>23</td>
<td>FINISH</td>
</tr>
<tr>
<td>24</td>
<td>END</td>
</tr>
</tbody>
</table>

Ready Data Map Open: IMSDEMO.DMB

---

Chapter 3: Importing Copybooks
Table 3-5 describes the Record Definition options:

Table 3-5. Record Definition Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Import</td>
<td>Select the item to import.</td>
</tr>
<tr>
<td>Name</td>
<td>Name of the item to import.</td>
</tr>
<tr>
<td>Skip</td>
<td>Select to go to the next item.</td>
</tr>
<tr>
<td>Stop Import</td>
<td>Select to stop the import process.</td>
</tr>
<tr>
<td>Record Browse</td>
<td>Specifies the record to view and apply the options to.</td>
</tr>
<tr>
<td>Apply to rest of import session</td>
<td>Applies the options you select to the rest of the items you import.</td>
</tr>
</tbody>
</table>

To replace fields in a segment when you import a copybook:

1. In the Record Definition options, change the name to match the existing segment name, and then click OK.
2. In the Duplicate Record Definition option, select **Overwrite** and then select **Fields Only**:

   ![Duplicate Record Definition](image)

3. Complete the import.
4. On the Data Maps tab of the Resource Explorer, right-click the relevant table and select **Properties**.
5. Right-click the table in the Record Dependencies list and select **Properties**.
6. In the Column Generation list, select **Refresh with missing columns**
7. Click OK.

Creating Records from Multiple Copybooks

This section describes how to import multiple copybooks through the PowerExchange Navigator. For more information, see “Importing Copybooks” on page 65.

It is assumed that you have already created a data map using a single copybook. The following instructions will show you how to import a second copybook and append its details to an existing record.
Adding a Second Copybook to the Record Definition

The starting point is a data map that has been created from a copybook, as shown.

1. Select the record in the Resource Explorer.
2. Select File > Import Copybook.
   The Import Copybook - Source Details dialog box appears.
3. Click Next.
4. In the File Name box, enter the path and filename of the copybook you want to import.
5. Click Next.
6. Click Finish.
7. Click OK.
8. Click OK.
9. Select the MASTER_REC record name from the Record Browse pull-down list.
10. Select the Append option.
    This adds the details to the current record.
11. Click OK.
12. Select the Overwrite option to replace the table with the new definition and click OK.
13. Click OK.
The MASTER_REC record looks like this:
CHAPTER 4

Managing Data Maps

This chapter includes the following topics:
- Data Checking, 93
- Importing Exported Data Maps, 95
- Searching in Data Maps, 96
- Sorting Data Map Records and Tables, 96
- File List Processing, 97

Data Checking

The quality of data to be extracted may not be consistently good, leading to failures during data selection and loading. PowerExchange supplies functionality to resolve bad data issues, providing two levels at which bad data can be handled. Record and field handling options can be defined at the Data Map level.

Note: Data Checking fields on every record will use additional processing effort and should be used carefully.

To initiate Data Checking and cleansing right-click the Data Map name and select Properties.
From the resulting Properties screen select the Data Checking tag to progress to the following screen.

It is on this screen that the datatypes to be checked are selected and the actions to be taken on return of data mapping and field data issues are determined.

If errors are found at the record level, such as OCCURS DEPENDING ON value greater than the maximum permitted, you can either terminate the complete extract or skip the row and a message will be inserted into the PowerExchange log file. It is good practice to inspect the source PowerExchange log file when initially extracting data with the second option, to see if there are any data quality issues.

If you want specific field types checked select the correct box. The choices are as follows: CHAR, DATE, NUMCHAR, PACKED, TIME, TIMESTAMP and ZONED. The following table lists the validation for these fields:

<table>
<thead>
<tr>
<th>Field Type</th>
<th>Validation</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHAR</td>
<td>Standard alphanumeric rules, 0-9, A-Z, a-z and greater than or equal to space (EBCDIC and ASCII)</td>
</tr>
<tr>
<td>DATE</td>
<td>Date format, defined within PowerExchange.</td>
</tr>
<tr>
<td>NUMCHAR</td>
<td>0-9, +,-, decimal point, comma, $, /, *, space.</td>
</tr>
<tr>
<td>PACKED</td>
<td>Standard packed decimal format (S9 COMP-3).</td>
</tr>
<tr>
<td>TIME</td>
<td>Time format defined within PowerExchange.</td>
</tr>
<tr>
<td>TIMESTAMP</td>
<td>Timestamp format defined within PowerExchange.</td>
</tr>
<tr>
<td>ZONED</td>
<td>Zoned numeric allowing sign and 0-9.</td>
</tr>
</tbody>
</table>

There are several options for handling bad field data:

- Terminate the data extract.
- Replace according to data defaults. For CHAR or DATE type fields this replacement value will be taken from the value in the Data Defaults sheet (selected from the same screen the Data Checking screen is selected from). The value can be inserted in every character position in the field or just the first character. Numeric fields will be set to zeros.
- Skip row and write log file msg. Skip the row, insert a message into the log file, and continue.
- Set to NULL. The invalid field will be set to NULL.
The Date, Time and Timestamp validation is carried out against the PowerExchange defined Date, Time and Timestamp datatypes. To define the format of these datatypes the record will need to be edited. Right-click the required column within the record.

♦ Enter a Field Name.

**Importing Exported Data Maps**

It is possible to use data maps that exist on other computers on your network. You can use them on a client by selecting the data maps from a remote computer where they have previously been saved.

1. To import one or more data maps, click File > Exported Data Maps right-click Exported Data Maps.
2. Select the remote node from which to download the data maps.
3. Click OK.
4. On the Exported Data Maps dialog box, select the data maps that you wish to import and click Import. The selected data maps are added to your data maps list and the Exported Data Maps dialog box is removed.
   Delete any data map by selecting the data map and clicking Delete.
Searching in Data Maps

There is a Search Facility for fields within records, or columns within tables. The facility can also be used with the Metadata Display. The facility can be operated in a simple or advanced mode. This is useful when there are a large number of fields, columns or tables in the display. To perform a search, place the cursor on the element to be searched and use the toolbar Binoculars icon or click Edit > Find. For more information about Metadata Display, see “Search Facility for Metadata Display” on page 105.

There are two search dialog boxes: Find Field in record and Find column in table. They both have similar functionality.

Clicking on the right arrow, the current wildcard characters that could be used in a search can be revealed.

A wildcard search change capability is also offered by clicking Advanced. This permits the Wildcard Characters and the Escape Character to be changed.

The Wildcard Characters (? and *) are available in all advanced searches. These characters were chosen, as they are unlikely to be used in table, field and column names. If they are being used, precede them with the Escape Character (~). For example a request for tab* would list all tables beginning with tab, whereas a request for tab~* would only list the table that was named tab*.

A search on item*, such as item + any other character on the records, items, and item_ct finds both of those fields.

However a search using item?, such as item + one other character, only finds items.

Note: View a list of previous searches by clicking on the down arrow beside the Find What box.

Sorting Data Map Records and Tables

When a data map has been created, the resulting data map may contain numerous records and tables. The records and tables will be separated out displayed as records then tables in the order in which they were added. To make it easier to select a particular record or table a sort by name feature is available.

To sort data map records and tables:

1. Select the data map from the list of data maps.
2. In the Resource Explorer, right-click the data map name.
3. Select Sort Items.

This causes the tables and records to be displayed in name order. This sort is temporary and upon selecting the data map the next time the records and tables will have reverted to the original order.

File List Processing

File List Processing provides a facility whereby sequential files can be concatenated and processed as if all the data were in one file.

All the file names to be concatenated will be specified in a special Filelist file. This file resides on the same platform as the data files that are being sourced.

The data map can then be processed in the normal way, such as from an ODBC call.

Restrictions

- File List Processing is available for SEQ(uential) or ESDS files and for TAPE file access.
- Offload processing will be disabled.
- Only Read Mode is supported.

ODBC

File List Processing can be activated by means of an ODBC escape sequence DTLFILELIST and the Filelist definition file specified with DTLD SN .

For more information, see the PowerExchange Reference Manual.

Filelist File

All the file names to be concatenated will be specified in a special Filelist file.

- This file must reside on the same platform as the data files that are being sourced.
- One file should be specified on each line.
- Blank lines ignored.
- Lines beginning /* are treated as comments and are ignored during processing.
- Blanks at the beginning of lines are ignored.
The specified file name in a Filelist file is all the characters between the leading and trailing blanks on a line. Please note that this means that MVS files with sequence numbers set to ON will include the sequence number as part of the file name.

A whole filelist file is checked for syntactical errors ahead of any data source file reading but references to missing files or read errors are handled when that particular file is read. Process terminates at the point the error is found.

Any invalid file names given in the Filelist file will appear in their entirety in hex in the PowerExchange Listener log output to aid in debugging.

Example Filelist Files

Windows/UNIX

c:\pwx\data\filelist1.dat
c:\pwx\data\filelist2.dat
c:\pwx\data\filelist3.dat

MVS

***************************************************************************** Top of Data ****************************************************************************
000001 /* FILE LIST PROCESSING DEFINITION FILE
000002 /*
000003 DTLUSR.V522.V1.DEMO.ESDS1
000004 /*
000005 DTLUSR.V522.V1.DEMO.ESDS2
000006 /*
000007 DTLUSR.V522.V1.DEMO.ESDS3
000008 /*
***************************************************************************** Bottom of Data ****************************************************************************

Example Scenario

To demonstrate the use of File List processing the following simple example scenario uses a file list definition file on the computer that references three locally held sequential data files through the PowerExchange local mode.

Example Filelist Definition File

c:\pwx\data\filelist1.dat
c:\pwx\data\filelist2.dat
c:\pwx\data\filelist3.dat

Data Map

A simple data map was created to map the data. For more information about creating this data map, see “A Simple Example” on page 12.

However, instead of mapping to a single data file, such as dem1.dat, you check the File List Processing check box on the SEQ Access Method dialog box and enter the full file name and path of the Filelist file into the File Name box.

Row Test Results

The data held in these three files is concatenated in the output results.

1, Mickey Mouse, M
2, Shirley Temple, F
3, John Wayne, M
4, Donald Duck, M
5, Greta Garbo, F
6, Mickey Mouse, M
7, Shirley Temple, F
8, John Wayne, M
9, Donald Duck, M
10, Greta Garbo, F
11, Mickey Mouse, M
12, Shirley Temple, F
13, John Wayne, M
14, Donald Duck, M
15, Greta Garbo, F
Overview

In addition to building data maps and command sets you can use the PowerExchange Navigator to access metadata for sources and targets. After retrieving the metadata, you can go directly to the row test feature and view some data. This allows you to discover what metadata is present. As this works for relational and non-relational sources, the facility can be a very powerful tool for viewing remote sources and targets without the need for any documentation.

Creating a Personal Metadata Profile

To use this facility, you define a Personal Metadata Profile. This stores the format of the metadata request, such that you can come back to it at a later moment and re-use or modify it and run it again. An example profile has been shipped with PowerExchange called demo_oracle.

Let us start out by setting up a Metadata Profile to access the demo non-relational data that was also shipped with PowerExchange.

Step 1. Add Personal Metadata Profile

Go to the Resources tab and add Personal Metadata by performing one of the following actions:

- Right-click Personal Metadata.
- Click the toolbar Add Personal Metadata icon.
- Select the Add, Personal Metadata menu option.
The following dialog box will be seen.

- The Name is the resource name.
- The Location is wherever the resource is to be found.
- There is a drop down list for Type and this will impact on the dimmed areas on the dialog box. A number of optional fields are then displayed according to Type.
- The user ID and password may be required depending upon the SECURITY setting in the PowerExchange configuration file. If the password is blank, the Navigator will prompt for the password to be entered as required. This prevents the password being stored and potentially causing password violations at a later date.

For our example:
1. Enter a Name of demo_nrdb.
2. Select a Type of NRBD.
3. Select a Location of local.
4. Click Next.

Step 2. Enter Personal Metadata Filters

The following dialog box is displayed.

1. This dialog box provides the option of putting in the same information as was seen in the Row Test for further qualifying the dtldescribe syntax. As we only want information regarding demo, enter a Schema of demo as above.

**Note:** You should not use an escape character to query database systems that are multibyte enabled.
2. Click Finish. Immediately the request will be run and in the left-hand pane the results will be visible. It should look something like the following.

![Image of Resource Explorer]

**Note:** If no tables appear below the demo_nrdb Metadata entry this may mean the corresponding Data Maps have not been exported while working through previous chapters. Data Maps will be exported when running a Row Test against them. If the tables are present skip the following bullet points, however, if they are absent then to export the data map:

- Double-click Data Maps.
- Double-click an example Map, for example Demo.map2.
- Click the table representation, in this case row_out.
- Click the Row Test icon.
- You will be prompted to export the Data Map and accept the export. To run the Row Test click Go.
- When you go back to the demo_nrdb Personal Metadata item then Data Maps exported should then be seen.

**Step 3. Explore Metadata**

To go down deeper, place the cursor on one of the tables and then right-click. This offers up the following options:

- Properties is a simple informational display.
- Explore will list the columns in the table as shown below for example demo.map1. table1.

![Image of Data Table]
By exploring further, the columnar information for several tables can be seen concurrently:

**Step 4. Perform Row Test**

1. Click the required table, in this case demo.map2.row_out, and the Row Test icon brightens.
2. Click the Row Test icon and you will then go to the Database Row Test dialog box.
3. Click Go. The data is then displayed.

### Row Tests and Non-standard Code Page Data

Non-standard code page data, including multibyte data, is available for selection from DB2 on both MVS and i5/OS, and the PowerExchange Navigator is capable of displaying double-byte characters. The Navigator uses a UTF-8 code page so will, by default, have the ability to display a wide character set. To receive the data in the correct format it is essential that changes are made to the PowerExchange configuration file on the platform from which the data is selected. The relevant parameters to review are DB2CODEPAGE for DB2 on MVS and DB2_BIN_CODEPAGE for DB2 on i5/OS. These parameters and additional information on code page support, is covered in the PowerExchange Reference Manual.

### Other Features of Personal Metadata

#### Metadata Column Display

The Metadata columnar display, reached by exploring the maps, offers several display options. Both the Name and Type columns can be displayed in either alphabetical or reverse alphabetical order.

![Metadata Column Display](image)

To activate this function click the title box heading for either the Name or Type columns. To display in reverse alphabetical order simply click the desired column heading again. In the example above, you may have noticed that the display shown has the Type column in reverse alphabetical order. A small arrow shape pointing up or down will appear in the heading box to indicate in which order the column is being displayed.
Search Facility for Metadata Display

The Search Facility described in “Searching in Data Maps” on page 96 can also be used with Personal Metadata to search for tables.

To access the Search Facility, use the Binoculars icon on the toolbar or click **Edit > Find**.

The following dialog box appears:

![Find Table Dialog Box](image)

The search may be conducted on Schema, Mapname or Table depending on the database. The Advanced facilities use the same Wildcard characters as described before.
CHAPTER 6

Using Database Row Test

This chapter includes the following topics:
♦ Database Row Test Overview, 107
♦ Performing a Database Row Test, 107
♦ Generating Restart Tokens for Change Data Capture, 112

Database Row Test Overview

You can use the database row test to preview data for all data sources and to generate restart tokens for PowerExchange change data capture sources.

Database row test can be used for objects in the following folders in PowerExchange Navigator:
♦ Extraction Groups in the Data Capture folder. You can use database row test with an extraction map to preview captured change data.
♦ Data Maps folder. You can use database row test with data maps validate data maps and to preview source data.
♦ Personal Metadata folder. You can use database row test to preview source data for tables displayed in personal metadata profiles.

You do not need to select an object to use database row test. You can also use database row test in the Resource Explorer window.

Performing a Database Row Test

Use the following procedure to perform a database row test.

To perform a database row test:
1. For data maps, extraction maps, and personal metadata tables, select the table.
2. On the menu bar, click File > Database Row Test. Alternatively, click the database row test icon.
3. The Database Row Test window displays:

![Database Row Test window](image)

Depending on the **DB Type** value, various fields will be available for input. The SQL statement displayed is a metadata request. For more information about PowerExchange nonrelational SQL, see the PowerExchange Reference Manual.

The Fetch list has the following options:

- **Columns.** Metadata request to retrieve information about columns.
- **Data.** Retrieve data rows.
- **Foreign keys.** Metadata request to retrieve information about Foreign keys.
- **Primary keys.** Metadata request to retrieve information about primary keys.
- **Records.** Metadata request to retrieve information about NRDB and NRDB2 records.
- **Schemas.** Metadata request to retrieve schema names for the DB_Type specified.
- **Tables.** Metadata request to retrieve information about tables.

4. To preview data, select **Data** from the **Fetch** list. By default, PowerExchange generates the following **SELECT** statement for data maps, extraction maps, and personal metadata tables:

   ```sql
   select * from table
   ```

   To retrieve data about one or more columns, select the columns in the table display prior to invoking database row test. PowerExchange adds the columns you select to the **SELECT** statement, instead of using asterisk.

5. Click **Go**.

   By default, PowerExchange displays 10 rows of data. You can display up to 999 rows of data by changing the default in **Get Rows**.

### Specifying Access Methods

If **Tables** or **Columns** is selected for NRDB, then the **Access Methods** list is enabled. It is possible to specify a list of valid access methods and the row test result will filter the result set for the access methods specified.

The **Get Rows** box allows you to specify how many rows to acquire on this next request. The default is 10, but you can set it to a maximum of 999. Extracting large amounts of data and adding it to these Windows tabular displays is not rapid. It is recommended that only a manageable number of rows be retrieved in a single request.

### PowerExchange SQL for Metadata Requests

In previous examples, only data was accessed from the source. However, you can also retrieve metadata to describe the tables and columns. There is a special PowerExchange syntax to do this, **DTLDESCRIBE**, which is explained fully in the chapter on Non-Relational SQL Syntax in the PowerExchange Reference Manual.
Note: The DTLDESCRIBE syntax is valid for both relational and non-relational sources.

For the metadata requests, the default value for the Schema name is converted to the case that is standard for the source database, such as uppercase for Oracle. For example, if you enter scott or SCOTT in the Schema field, you get the same results, because scott is converted to uppercase. By checking the Respect Case box, defaulting to the database case will not be done. Hence a Schema name of scott will produce no results, whereas SCOTT will be successful.

The metadata qualifiers support the following wildcards:

- An asterisk (*) represents one or more matching characters.
- A question mark (?) represents one matching character.

These characters were chosen, as they are not likely to be used in table and column names. If they are being used, precede them with the Escape Character ~. For example a request for tab* would list all tables beginning with tab, whereas a request for tab~* would only list the table that was named tab*.

### Specifying Advanced Parameters

The Advanced button on the database row test dialog box appears if you select CAPX, CAPXRT, or EMR from the DB Type pull-down list.

Click this button to display the Advanced Parameters dialog box.

This dialog allows you to set parameters for the Row Test that override the normal settings in the PowerExchange configuration file.

The fields that are displayed and/or enabled depend on whether you have selected CAPX, CAPXRT, or EMR.

### CAPX Advanced Parameters

Use the following dialog to set CAPX parameters:

![CAPX Advanced Parameters](image)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Values</th>
<th>Description</th>
</tr>
</thead>
</table>
| Extract Type  | SL, RS | SL = Extract all data since last extract run  
               |         | RS = Restart previous (or specified) extract |
| Image Type    | BA, AI, TU | BA = Before and after images are delivered as Delete and Insert records for any change.  
               |         | AI = Captures only the data after image (the latest change).  
               |         | TU = Before and After images delivered with an action indicator of change (rather than delete and insert as delivered by BA). The application that processes the row will have to be able to apply changes rather than just delete and insert affected rows. |
### CAPXRT Advanced Parameters

Use the following dialog to set CAPXRT parameters:

#### Parameter | Values | Description
--- | --- | ---
Extract Type | SL,RS | SL = Extract all data since last extract run  
RS = Restart previous (or specified) extract  
Image Type | BA, AI, TU | BA = Before and after images are delivered as Delete and Insert records for any change.  
AI = Captures only the data after image (the latest change).  
TU = Before and after images delivered with an action indicator of change (rather than delete and insert as delivered by BA). The application that processes the row will have to be able to apply changes rather than just delete and insert affected rows.  
Timeout | 0 - 86400 | Sets the maximum approximate time (in seconds) to wait for data on a queue before returning EOF.  
0 indicates that EOF is returned immediately that there is no more data.  
86400 indicates that EOF is never returned. The job will wait forever.  
No Progress Update | | Default is checked. When checked PowerExchange will not treat this as a true application extract.  
AS400 Library/Journal | | The fully qualified library and journal name that is to be used instead of that specified in the PowerExchange Change Capture Registration.  
For example:  
STQA/NK4JOURNAL
Performing a Database Row Test

Use the following dialog to set EMR parameters:

**Figure 6-3. EMR Advanced Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Values</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Oracle Instance</td>
<td></td>
<td>Allows the user to override Oracle instance information for a given Oracle Collection Id. This means that the user can use a single set of registrations to capture data from multiple Oracle instances. This overrides the second value of the ORACLEID statement in the PowerExchange configuration file. For example: ORACLEID=(coll_id,oracle_sid,connect_string,cap_connect_string) Used in conjunction with the Oracle Connection string. The user can specify either or both Instance/Connection string; if one of the keywords is not specified, Oracle Capture will pick up the value of the other from the PowerExchange configuration file.</td>
</tr>
<tr>
<td>Oracle Connection String</td>
<td></td>
<td>Allows the user to override Oracle connection information for a given Oracle Collection Id. This means that the user can use a single set of registrations to capture data from multiple Oracle instances. This overrides the fourth value of the ORACLEID statement in the PowerExchange configuration file. For example: ORACLEID=(coll_id,oracle_sid,connect_string,cap_connect_string) Used in conjunction with the Oracle Instance. The user can specify either or both Instance/Connection string; if one of the keywords is not specified, Oracle Capture will pick up the value of the other from the PowerExchange configuration file.</td>
</tr>
<tr>
<td>Oracle Schema</td>
<td></td>
<td>Allows the user to override the schema name to use for a group of registrations. This means that the user can use a single set of registrations to capture data from multiple schemas that may exist in a given Oracle instance.</td>
</tr>
<tr>
<td>DB2 UDB Database</td>
<td></td>
<td>This parameter overrides the DB2 UDB database connection obtained from the extraction map.</td>
</tr>
</tbody>
</table>

**EMR Advanced Parameters**

Use the following dialog to set EMR parameters:

**Parameter**  | **Values**  | **Description**                                                                                                                                 |
|----------------|-------------|--------------------------------------------------------------------------------------------------------------------------------------------|
| Extract Type   | SL,RS       | SL = Extract all data since last extract run  
RS = Restart previous (or specified) extract                                                                                                                                 |
| Image Type     | BA,AI,TU    | BA = Before and after images are delivered as Delete and Insert records for any change.  
AI = Captures only the data after image (the latest change).  
TU = Before and After images delivered with an action indicator of change (rather than delete and insert as delivered by BA). The application that processes the row will have to be able to apply changes rather than just delete and insert affected rows. |
You can use database row test to generate restart tokens for data sources. You can then use these generated restart tokens to populate the PWXPC restart token file for a PowerCenter session. PowerExchange generates restart tokens using the location and source type you specify. These restart tokens represent the current end of the change stream at the time at which you perform the database row test. The PowerExchange Navigator displays the generated restart tokens in the Database Row Test Output window. You can then copy the output from this window so you can populate the PWXPC restart token file.

The following methods also generate restart tokens:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Values</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Timeout</td>
<td>0 - 86400</td>
<td>Sets the maximum approximate time (in seconds) to wait for data on a queue before returning EOF. 0 indicates that EOF is returned immediately that there is no more data. 86400 indicates that EOF is never returned. The job will wait forever.</td>
</tr>
<tr>
<td>No Progress Update</td>
<td></td>
<td>Default is checked. When checked PowerExchange will not treat this as a true application extract.</td>
</tr>
<tr>
<td>AS400 Library/Journal</td>
<td></td>
<td>The fully qualified library and journal name that is to be used instead of that specified in the PowerExchange Change Capture Registration. For example: STQA/NEWJOURNAL</td>
</tr>
<tr>
<td>Oracle Instance</td>
<td></td>
<td>Allows the user to override Oracle instance information for a given Oracle Collection Id. This means that the user can use a single set of registrations to capture data from multiple Oracle instances. This overrides the second value of the ORACLEID statement in the PowerExchange configuration file. For example: ORACLEID=(coll_id,oracle_sid,connect_string,cap_connection_string) Used in conjunction with the Oracle Connection string. The user can specify either or both Instance/Connection string; if one of the keywords is not specified, Oracle Capture will pick up the value of the other from the PowerExchange configuration file.</td>
</tr>
<tr>
<td>Oracle Connection String</td>
<td></td>
<td>Allows the user to override Oracle connection information for a given Oracle Collection Id. This means that the user can use a single set of registrations to capture data from multiple Oracle instances. This overrides the fourth value of the ORACLEID statement in the PowerExchange configuration file. For example: ORACLEID=(coll_id,oracle_sid,connect_string,cap_connection_string) Used in conjunction with the Oracle Instance. The user can specify either or both Instance/Connection string; if one of the keywords is not specified, Oracle Capture will pick up the value of the other from the PowerExchange configuration file.</td>
</tr>
<tr>
<td>Oracle Collection ID</td>
<td></td>
<td>Allows the PowerExchange Listener to submit multiple simultaneous Oracle Capture processes that can connect to different Oracle instances. Oracle Capture will now use the override to determine to which instance it should connect, as opposed to using the ORACOLL keyword of the CAPI_CONNECTION_TYPE=ORCL statement in the PowerExchange configuration file. This will allow the customer to use a single listener to capture data from as many as 10 Oracle instances simultaneously.</td>
</tr>
<tr>
<td>HP3000 Volume Set</td>
<td>Do not use.</td>
<td></td>
</tr>
<tr>
<td>DB2 UDB Database</td>
<td>This parameter overrides the DB2 UDB database connection obtained from the extraction map.</td>
<td></td>
</tr>
</tbody>
</table>
The special override statement with CURRENT_RESTART in the PWXPC restart token file. PWXPC and PowerExchange generate restart tokens that represent the current end of the change stream at the time the PowerCenter session runs.

The DTLUAPPL utility with the GENERATE_RSTTKN parameters and a valid capture registration. DTLUAPPL generates restart tokens that represent the current end of the change stream at the time the utility runs.

For more information about these other methods, see PowerExchange Interfaces for PowerCenter.

Use the following procedure to generate restart tokens from PowerExchange Navigator database row test.

To generate restart tokens:
1. On the menu bar, click File > Database Row Test. The Database Row Test window appears.
2. In the DB Type list, select CAPXRT.
3. In the Location list, select the node name that represents the location of the data source.
   PowerExchange Navigator populates the Location list with the node names you specify in the dbmover.cfg file on the Windows machine.
4. In the Application Name box, enter a value for the application name.
   You can enter between 1 and 20 characters. PowerExchange does not store this application name so you can enter any value in this box.
5. In the SQL Statement box, enter the PowerExchange SQL statement to generate restart tokens. For more information about the syntax of the SQL statement, see “SQL Statement for Generating Restart Tokens” on page 113.
6. Click Go. The Database Row Test Output window appears and displays the generated restart tokens.
7. To copy the generated restart tokens, right-click in the output window and then click Copy Output.
   You can use the generated restart tokens in the PWXPC restart token file. For more information about PWXPC restart token files, see PowerExchange Interfaces for PowerCenter.

SQL Statement for Generating Restart Tokens
PowerExchange provides a unique SQL statement to generate restart tokens. Use the following syntax to generate restart tokens:

```
SELECT CURRENT_RESTART [WHERE CONNAME=conn_name|CONTYPE=conn_type]
```

This SQL statement has the following parameters:

**SELECT CURRENT_RESTART**
- Specifies that PowerExchange should generate current restart tokens.
  - If you do not specify either CONNAME or CONTYPE, PowerExchange uses the CAPI_CONN_NAME value for the PowerExchange Listener in the source location to determine the data source type.

**CONNAME=conn_name**
- Specifies the CAPI_CONNECTION name at the source location that PowerExchange uses to determine the data source type.

**CONTYPE=conn_type**
- Specifies the connection type that PowerExchange uses to determine the data source type.
  - Restart token formats vary by source type. Specify one of the following values for conn_type:
    - ADA for Adabas sources (MVS)
    - AS4 for DB2 for i5/OS sources
    - DCM for Datacom sources (MVS)
    - DB2 for DB2 for z/OS sources (MVS)
- **IDL** for IDMS log-based sources (MVS)
- **IDM** for IDMS synchronous sources (MVS)
- **IML** for IMS log-based sources (MVS)
- **IMS** for IMS synchronous sources (MVS)
- **ORA** for Oracle sources
- **MSS** for Microsoft SQL Server sources
- **UDB** for DB2 for Linux, UNIX, and Windows sources
- **VSAM** or **VSM** for VSAM sources (MVS)

For MVS sources, all MVS connection types result in restart tokens of the same format because PowerExchange records all change data from MVS sources in the PowerExchange Logger. Restart tokens for MVS sources represent locations within the PowerExchange Logger logs.
CHAPTER 7

Working with Capture Registrations

This chapter includes the following topics:
♦ Capture Registration Overview, 115
♦ Adding Registration Groups, 115
♦ Adding Capture Registrations, 118
♦ Modifying Capture Registrations, 124
♦ Changing Properties of the Registration Definition, 126

Capture Registration Overview

In order for PowerExchange to capture changes, you must register source data sets and tables for change data capture using the PowerExchange Navigator. PowerExchange only collects changes when the source is properly registered. Each creation of a Registration Group automatically creates an Extraction Group and Application Group with the same name.

When defining an actual capture registration for a source, PowerExchange also defines an extraction map for that registration. You can modify these default extraction maps and create new extraction maps based on the same capture registration.

Once a source is registered and the status is set to active, this version of the capture registration can no longer be changed. This is to ensure that the data already collected is not invalidated. To change an existing registration, a new version must be created and activated. For more information about capture registration versions, see "Version Indicator in Extraction Maps" on page 138.

Note: The maximum column length allowed for PowerExchange change data capture is 32,244 bytes.

Adding Registration Groups

Before you can add capture registration, you must create a registration group in the Data Capture folder of the Resource Explorer.

A registration group is a named group of individual capture registrations. It is a name of your choice. The related properties define the location of the PowerExchange Listener, the type of capture source, the name of the source subsystem and information such as user ID and password. All capture registrations within a registration group must access the same source type and instance of that source type.
Registration group information is stored locally on the Windows machine on which it was created and is unique for that machine. Accessing a PowerExchange Listener shows all registrations for the database instance associated with the registration group.

**To add a registration group:**

1. On the toolbar, click the *Add Registration Group* icon. Alternatively, right-click the Registration Group folder, and then click *Add Registration Group*.

2. The Add Registration Group dialog box appears. Enter values for the options in the dialog box.

   The following example shows the options for a DB2 for z/OS source type:

   ![Add Registration Group Dialog Box](image)

   The number and names of some options vary based on the source type selected.

   The Add Registration Group dialog box has the following options:

   **Name**
   
   Specify the registration group name. The registration group name can be between 1 and 16 characters in length and can be any combinations of alphabetic characters and numbers.

   **Location**
   
   Specify a valid PowerExchange location. The location name must be specified in a NODE statement in the dbmover.cfg file on the Windows machine. The NODE statement must point to the location of where PowerExchange change data capture is running or to location local if the source resides on the same Windows machine.

   For example, if you define a registration groups for DB2 for z/OS sources the location should point to the PowerExchange Listener running on the same MVS image as the PowerExchange Agent, PowerExchange Logger, and DB2 ECCR.

   **Type**
   
   Specify a valid source type from one of the following:
   
   - **ADABAS** for Adabas files
   - **AS4** for DB2 for i5/OS
   - **DATACOM** for Datacom tables
   - **DB2** for DB2 for z/OS tables
   - **DB2UDB** for DB2 for Linux, UNIX, and Windows
   - **IDMS** for IDMS tables registered for IDMS synchronous change data capture
   - **IDMS_L** for IDMS tables registered for IDMS log-based change data capture
Adding Registration Groups

- **IMS** for IMS databases
- **MSSql** for Microsoft SQL Server tables
- **ORACLE** for Oracle tables
- **VSAM** for VSAM ESDS, KSDS, RRDS, and VRRDS data sets

**UserID**
Optional. Specify a valid operating system or database user ID.

The user ID is a valid database user ID and is required for DB2 for Linux, UNIX, and Windows, Microsoft SQL Server, and Oracle. For MVS and i5/OS source types, user ID is required if PowerExchange security is active. For more information about PowerExchange security, see the PowerExchange Reference Manual.

**Password**
Optional. Specify the password for the user ID specified in **UserID** option.

Note: You can leave the **Password** blank for MVS and i5/OS source types. This prevents the password being stored and potentially causing password violations at a later date. When blank, PowerExchange Navigator prompts for the password when necessary.

**Collection Identifier**
Specify a unique name for the group of registrations. A collection identifier name is also called an instance name.

The **Collection Identifier** option displays when the following source types are selected:
- **ADABAS**
- **AS4**
  - For DB2 for i5/OS tables, specifies the instance name specified in the INST parameter of the AS4J CAPI_CONNECTION statement in the DBMOVER member of the dtllib/CFG file on the node specified in the **Location** option.
- **ORACLE**
  - For Oracle tables, specifies the instance name specified in the collection_id variable of the ORACLEID statement in the dbmover.cfg file on the node specified in the **Location** option.
- **VSAM**

**CV Name**
For IDMS tables, specifies the name of the IDMS Central Version (CV) address space in which changes for the tables occur.

**Database**
For DB2 for Linux, UNIX, and Windows tables, specifies the database that contains the tables.

**Database Instance**
For DB2 for z/OS tables, specifies the DB2 subsystem ID or the DB2 data-sharing group name that contains the tables. A database instance name is also called an instance name.

**Database Name**
For Microsoft SQL Server tables, specifies the database name on the database server that contains the tables.

**Database Server**
For Microsoft SQL Server tables, you select the database server where the database that contains the tables resides.

**Logsid**
For IDMS tables used in IDMS log-based change data capture, specifies a unique name for the group of registrations. The name selected must be specified in the LOGSID parameter in the DBMOVER configuration parameters on the MVS system where the IDMS tables reside.
MUF Name
For Datacom tables, specifies the name of Multi-User Facility (MUF) in which changes for the tables occur.

Recon Identifier
For IMS synchronous change data capture, specifies the IMS subsystem ID in which changes for the tables occur.
For IMS log-based change data capture, specifies the name specified in the IMSID statement of the DBM OVE R configuration file parameters on the MVS system where the RECON data sets reside.

Source Map Location
For IDMS tables used in IDMS log-based change data capture, specifies the PowerExchange location containing the data maps for the IDMS tables.

Add Registration
If selected, PowerExchange Navigator invokes the Add Capture Registration dialog so that a capture registration for the source type selected in the Type option can be added. By default, this option is selected. If cleared, only a registration group is added.

3. Click Next if Add Registration is selected. Otherwise, click Finish.

Adding Capture Registrations

PowerExchange uses capture registrations to determine which source data sets and tables participate in change data capture. Use the PowerExchange Navigator to register source data sets and tables for which you want to capture and extract changes. Capture registrations are stored in the CCT file at the source location specified in the capture registration group to which the capture registration belongs.

To add a capture registration:
1. When adding a new registration group, select Add Registration on Add Registration Group dialog box.
If the registration group exists, double-click the registration group name to open the group. Then, right-click the registration group name and click Add Capture Registration.
2. The Add Capture Registration - Name and Table Filter dialog box displays:
Enter values for the options in the dialog box to filter the source tables displayed. The conditions are logically associated by the “and” connector.

The Add Capture Registration - Name and Table Filter dialog box has the following options:

**Name**
Specify the name for the capture registration. The capture registration name must be between 1 and 8 characters in length and must begin with an alphabetic character. The remaining characters can be any combination of alphabetic character and numbers. Capture registration names are always lower case.

**Table Filter**
Optional. Specify the filter criteria for the displaying a list of tables from which you chose the one to be registered. You can specify values in any or all of the options. Use the asterisk (*) to represent a character or set of characters at any point in the value. The filter option names varies based on source type.

- **Creator/Schema or Schema or Owner**
  Specify the owner or schema name of the table.

- **DBName/Definer or Mapname**
  Specify the database name for DB2 tables or the IMS data map name for IMS databases.

- **Table**
  Specify the table name.

- **Respect Case**
  For DB2 and Microsoft SQL Server, select Respect Case if you want the case of the value specified in the table filter to be used. Otherwise, PowerExchange ignores case when searching for tables to match the filter criteria.

**Escape Character**
Optional. Specify if any of the values for the Table Filter options contain restricted characters, such as %. Place the escape character value prior to the restricted character. For example, with a table name called my%tab and an escape character of \, enter the table name as my\%tab.

3. Click Next.

4. The Add Capture Registration - Tables and Columns dialog box displays. Select the table and columns and click Next.

On this dialog box, select the source being registered for change data capture. The tables in the display are filtered based on the values specified in the Table Filters options on the Add Capture Registration - Name and Table Filter dialog box.

There can only be one capture registration for a given source. Great care needs to be taken when selecting the individual columns for capture. An extraction process can select a subset of the captured columns, but only for columns which have been captured in the first place.

The display window shows a list of tables on the left. Select a table to display the columns on the right. Columns marked with a 'key' are the index columns for that table. Key columns are used as part of the Condense processing. Care needs to be taken when de-selecting key columns to ensure the “uniqueness” of the row is not compromised. Where a Primary Key is specified on the table then only those key columns are checked by default.
The following example shows a list of DB2 for z/OS tables and the columns for the table selected from that list:

![Add Capture Registration - Tables and Columns dialog box]

Description of the dialog box options follow:

### Table 7-1. Tables

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Creator Name or Schema</td>
<td>Table creator</td>
</tr>
<tr>
<td>Name</td>
<td>Table name</td>
</tr>
<tr>
<td>Type or DataSet Name</td>
<td>Type or data set name</td>
</tr>
</tbody>
</table>

### Table 7-2. Columns

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select Columns</td>
<td>Indicates whether a column is selected or not.</td>
</tr>
<tr>
<td>Name</td>
<td>Column Name</td>
</tr>
</tbody>
</table>

Select one of the following options for the columns to register:

- Select the specific columns that are required for capture. If a change is made to a column that has not been selected then a change is not logged.

- Click **Select All Columns**. Now all columns are selected and a change to any column within a row causes that row to be logged. This option is only available for source types that support selective column capture.

- **Select all and notify changes**. This removes all the check boxes and all columns are selected. Like option 2, any change to a row causes that row to be logged. This option is only available for DB2 for z/OS sources.

The key difference between **Select All Columns** and **Select all and notify changes** is seen when the source table changes. With specific column selection or **Select All Columns**, a column can be added to the source table with no impact. Columns that have been specifically selected and new columns added to the source table are ignored until they are selected. When you use **Select all and notify changes**, PowerExchange CDC fails if you change the table schema.
When you select **Select all and notify changes** for DB2 for z/OS tables and you change the table schema, the DB2 ECCR abends it reads the first change record for that table after the schema change. For more information about schema changes, see the PowerExchange Change Data Capture Guide for z/OS.

5. After you select the table and columns, the Add Capture Registration- Type dialog box displays. Specify the options and click **Finish**. The PowerExchange Listener writes the capture registration to the CCT file on the source platform.

This dialog box defines the type and characteristics for a capture registration.

![Add Capture Registration - Type dialog box](image)

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
</table>
| Type           | Displays the capture type for the data source:  
  - **Log-Based**. PowerExchange captures changes from the data source logs.  
  - **Synchronous**. PowerExchange captures changes in a synchronous manner. No selection is necessary. |
| Status         | This can have one of two values:  
  - **Inactive**. The registration is to be defined but is not to be used by the Collection agent.  
  - **Active**. The registration is to be defined and used by the Collection Agent on the next start. |
| Condense       | Specifies the type of condense processing to do for the capture registration:  
  - **None**. Indicates that no condense processing is done.  
  - **Part**. Indicates that condense processing is done and keeps all committed changes for each source row.  
  - **Full**. Indicates that condense processing is done and amalgamates all changes for each source row, storing only the first before and last after image. Default is None. For more information about the Condense option, see Table 7-3. |
| Database ID    | For Adabas sources, displays the DBID of the table selected.                                                                                   |
| File Number    | For Adabas sources, displays the file number of the table selected.                                                                              |
| Comment        | An appropriate comment to identify the Capture Registration.                                                                                     |

**Defining the Condense Option**

The Condense option defines the type of the condense processing PowerExchange should perform for the capture registration.
Table 7-3 describes the Condense options:

Table 7-3. Condense Options

<table>
<thead>
<tr>
<th>Condense Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Full</td>
<td>Defines full condense processing. This type of condense amalgamates changes to deliver the first before and last after image. This is the option to be chosen for CAPX processing. <strong>Note:</strong> Full can only be selected if the table has a unique key which becomes the key for which a single before image and after image is maintained. As such, Full condense should only be used when the unique key is not updated by anything other than Delete followed by an Insert to reinstate the data on a new key.</td>
</tr>
<tr>
<td>Part</td>
<td>Defines partial condense processing. This type of condense keeps all changes to a given row or record in a source. This is the option to be chosen for CAPX processing.</td>
</tr>
<tr>
<td>None</td>
<td>No condense processing is performed. This is the option to be chosen for CAPXRT processing.</td>
</tr>
</tbody>
</table>

Adding a Capture Registration to an Existing Group

You can add capture registrations to an existing group.

**To add a capture registration to an existing group:**

1. Right-click the required Registration Group, and then click **Open**.
2. On the toolbar, click the Add Capture Registration icon. The Add Capture Registration - Name and Table Filter dialog box appears.
3. Complete the steps in “Adding Capture Registrations” on page 118.

Deleting a Registration Group

You can delete registration groups.

Deleting a Registration Group does not delete any Registration Entries. It only deletes the Group information that is held locally on the workstation.

**To delete a registration group:**

1. Select the appropriate Registration Group and click **Delete**. A confirmation dialog box displays.
2. Click **Yes** to delete. If no active registrations are outstanding for this group, the delete is accepted and the group is removed; otherwise, a further warning is issued stating the registration group has active registrations.
   The second warning is intended to prevent confusion when a group is deleted while an active registration remains that might still be capturing data.

Deleting a Registration Entry

You can delete registration entries.

**To delete a registration entry:**

- Select the appropriate Registration Entry and click the **Delete** button. If the registration is not active you are asked to confirm the deletion, otherwise you are prompted to make the registration history before deleting.
This action deletes the registration entry on the source platform. The next time the Collection Agent is restarted no more changes are collected for that registration.

**Viewing a Registration Group**

You can view registration groups.

**To view a registration group:**

- Select a group in the Resource Explorer. This displays the details for that group in the Resource Inspector. A complete list of registrations appears under the Registration Group in the Resource Explorer.

**Viewing a Capture Registration**

You can view the details of registration.

**To view capture registration details:**

- Double-click a Capture Definition Group from the Resource Explorer, Resource tab.
- Select a Capture Registration from the group list, and then double-click the Registration Name that you want to view. The following example shows a DB2 for z/OS capture registration:

![Resource Explorer](image)

**Resource Explorer**

The Resource Explorer displays a list of capture registration groups and the capture registrations within each group. To select and open a capture registration double click on the capture registration name. All the Registration details are displayed in the capture registration details dialog boxes.

**Capture Registration Details**

Capture registration details display the column information of the table referenced by the capture registration. This only displays details for those columns that are selected for capture.

This includes:
- Name
- Table Name
- Column No
- Type
To make changes to an existing list of columns (for a given source) right-click on the registration name.

**Resource Inspector**

The Resource Inspector displays the registration properties of the capture registration. For example, Type is Synchronous and the Tag contents are used when writing changes to the PowerExchange Logger.

If you make any changes they must be confirmed by clicking on the Apply button. Once confirmed, the capture registration in the Resource Explorer changes color. This is an indication that the source requires to be saved.

**Modifying Capture Registrations**

After a data source has been registered for change capture it is possible that you may be required to alter that registration to take into account changes to your processing or underlying data.

Any change has implications for the PowerExchange change capture process, some more than others.

The major types of allowable changes are:

- Changing the PowerExchange view of the captured data
- Changing the underlying data source
- Changing registration definition properties

**Changing a Capture Registration**

You can change the following attributes of capture registrations:

- With DB2 and Microsoft SQL Server sources, add and remove columns.
- Change the status from inactive to active or active to history.
- Change the condense from none to part or part to none.
- For all MVS and i5/OS sources, change condense for sources with keys from none or part to full.
To change a capture registration to add or remove columns:

1. Right-click on the capture registration, and then click **Open**.

2. Right-click on the capture registration, and then click **Amend Columns**. The following example shows a DB2 for z/OS capture registration:

3. PowerExchange Navigator displays all available columns for the table.

4. Highlight the column(s) required, in this case **PHONE**, and click **Add**. Click **Remove** to remove columns.
5. When the change is accepted the Navigator displays the following information.

The window above now shows a red icon against the capture registration a version number of 2 against the registration. The PHONE column now appears in the capture registration in the middle pane of the window. Next time the registration group is opened, it shows the original Active capture as Version 1 and the new amended Version 2 as inactive.

The change is not effective until this version is set to Active as described previously. For more information, see “Setting to Active” on page 126.

When this new version is set to Active, a confirmation dialog box appears.

Once this has been accepted, the registration now set to history cannot be re-activated. Once set to active, PowerExchange captures the amended data, however, the original version is part of the original extraction definition. For more information about bringing the new definition into effect for extracting data, see “Changing the Capture Registration Associated with an Extraction Map” on page 137.

Changing Properties of the Registration Definition

It may be necessary to change some of the registration properties of the Registration. The following properties can be amended:

- Status
- Condense
- Comments

The Resource Inspector dialog box displays the registration properties of the Capture Registration. If you make any changes they must be confirmed by clicking the Apply button. Once confirmed, the Registration in the Explorer changes color. This is an indication that the source requires to be saved.

For example, you may have created a registration with a status of Inactive and now wish to use it in your environment. You must change it to Active.

Setting to Active

Creating the capture registration db2capc with a status of inactive prevents data from being captured until everything is in place for the data to be useful to apply to the target.

The capture registration usually remains inactive until the target has been materialized. When ready, the registration is set to Active.
To activate a capture registration:
1. Right-click the capture registration, and then click Open.
2. In the Resource Inspector, click Status.
3. Select Active. A small red square appears against the Status literal.
4. Click Apply.
   Note: The icon to the left of the registration turns red.
5. Right-click on the capture registration, and the click Close.
6. Click Yes in the confirmation dialog box, and the change is saved.

Setting to History

If a capture registration is no longer required, you can disable change data capture for that source by setting the capture registration status to history. Open the Registration Group, and then double click or Open the Registration definition. In the following window, select the field that says Active, and then select History from the drop-down list.

When History is selected, a red square appears next to the amended field, which disappears when you click the Apply button.

When this has been carried out, the registration cannot be made active again. A new registration would need to be added to start capture again. If wishing to delete the registration setting to history has the advantage of retaining the capture information to date.
CHAPTER 8

Working with Capture Extraction Maps

This chapter includes the following topics:

- Extraction Maps Overview, 129
- Adding Extraction Groups, 130
- Adding Extraction Maps, 132
- Version Indicator in Extraction Maps, 138
- Using Change Indicator and Before Image Columns, 139

Extraction Maps Overview

Use extraction maps to extract changes from PowerExchange change data capture. The type of extraction is dependant on the type of processing being performed against the specific source.

Batch and continuous extraction mode as well as database row tests using the CAPX DB type read changes from condense files created by PowerExchange Condense. Real-time extraction mode and database row tests using the CAPXRT DB type read changes from the change stream.

When you create a registration group, PowerExchange creates an extraction group with the same name. When you create a capture registration, PowerExchange creates an extraction map with the same name and same columns. You can alter the default extraction maps. You can also create additional extractions maps for the same capture registration using the Add Extract Definition wizard.
The extraction map names have the following format:

\[Xninstance\_regname\_tablename\]

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
</table>
| \(X\)    | The first character defines the type of extraction maps and has the following values:  
|          | - D for default extraction maps  
|          | - U for user-defined extraction maps |
| \(n\)    | The second character defines the source type and has the following values:  
|          | - 1 for DB2 for z/OS  
|          | - 2 for IMS  
|          | - 3 for DB2 for i5/OS  
|          | - 4 for Adabas  
|          | - 5 for IDMS synchronous change data capture  
|          | - 6 for VSAM  
|          | - 7 for Datacom  
|          | - 8 for Oracle  
|          | - 10 for Microsoft SQL Server  
|          | - B for DB2 for Linux, UNIX, and Windows  
|          | - D for IDMS log-based change data capture |
| instance | The instance value varies by source type and contains the value specified when the registration group was created. For more information about instance values, see “Adding Registration Groups” on page 115. |
| regname  | The name assigned when the capture registration was created. |
| tablename | The table name of the relational source table or from the source data map. |

For extraction maps, the variables \(Xninstance\) are called schema. When you open the extraction group, PowerExchange Navigator displays the extraction maps as schema.regname. When you extract changes using an extraction map, specify the full extraction map name of schema.regname_tablename. For example, when using database row test, PowerExchange Navigator generates the following SQL statement to extract the changes:

```
SELECT * FROM schema.regname_tablename
```

You can select all columns or specific columns using extraction maps.

### Adding Extraction Groups

Extraction maps are defined under the Extraction Group in the Resource Explorer. By default, whenever a registration group is defined an extraction group with the same name is created. There is normally no need to define extraction groups manually, but you can create extraction groups.

**To add an Extraction Group:**

1. On the toolbar, click the **Add Extraction Group** icon. Alternatively, right-click **Extraction Group**, and then click **Add Extraction Group**.
2. Complete the Add Extraction Group dialog box.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>The name of the Extraction Group.</td>
</tr>
<tr>
<td>Location</td>
<td>Specifies the name as defined in the dbmover.cfg file where the PowerExchange Listener is located.</td>
</tr>
<tr>
<td>Type</td>
<td>The type of data to be extracted.</td>
</tr>
<tr>
<td>UserID/Password</td>
<td>User ID and password to identify the user and check their authorizations.</td>
</tr>
</tbody>
</table>
### Field | Description
---|---
Collection Identifier | Specifies a unique name for the group of registration. Specify the same value as specified for the matching registration group. This field displays for the following source types:  
- Adabas  
- DB2 for i5/OS  
- Oracle  
- VSAM
CV Name | For IDMS sources only, the name of the IDMS Central Version (CV) address space in which changes occur. Specify the same value as specified for the matching registration group.
Database | For DB2 for Linux, UNIX, and Windows sources only, specifies the database that contains the tables. Specify the same value as specified for the matching registration group.
Database Instance | For DB2 for z/OS sources only, the DB2 subsystem or group name. Specify the same value as specified for the matching registration group.
Database Name and Database Server | For Microsoft SQL Server sources only, the database name on the database server that contains the tables. Specify the same value as specified for the matching registration group.
Logsid | For IDMS log-based sources only, a unique name for the group of registrations. Specify the same value as specified for the matching registration group.
MUF Name | For Datacom sources only, the name of Multi-User Facility (MUF) in which changes occur. Specify the same value as specified for the matching registration group.
Recon ID | For IMS synchronous sources only, the IMS subsystem ID in which changes for the tables occur. For IMS log-based sources, the name specified in the IMSID statement of the DBMOVER configuration file parameters on the MVS system where the RECON data sets reside. Specify the same value as specified for the matching registration group.
Source Map Location | For IDMS log-based sources only, the PowerExchange location containing the data maps for the IDMS tables. Specify the same value as specified for the matching registration group.
Add Extraction Definition | Identifies whether the extraction details are given now or at a later stage. If the box is not checked the process completes at this point and the Next button changes to Finish.

3. Click **Next** or click **Finish**.

All existing extraction maps for the specified Type, Database Instance, and Location are displayed.
Adding Extraction Maps

To add an extraction map:

- Complete the dialog box as shown in the following sample.

The constructed extraction map name has the format:

```
schema.RegName_TableName
```

In this example, the extraction map name is:

```
u1dsn1.DB2MAP.DB2TAB
```

Adding an Extraction Map - Capture Registration Dialog Box

Using this dialog box, the user can add a version of a capture registration to the extraction process definition. The Available Capture Registrations list displays the Capture Registrations that are available for selection. The Current Capture Registrations list displays the name of the Capture Registrations that have been selected for the extraction map.
To add a version of a capture registration to an extraction map:

- Highlight the required capture registration, and then click Add.

### Table 8-1. Available Capture Registrations

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Displays the name of the Capture Registrations that are available for selection.</td>
</tr>
<tr>
<td>Version</td>
<td>Displays the version number of the Capture Registration.</td>
</tr>
<tr>
<td>Condense</td>
<td>Displays the condense state of the Capture Registration (None, Part or Full).</td>
</tr>
<tr>
<td>Status</td>
<td>Displays the status of the Capture Registration. This can have one of two values:</td>
</tr>
<tr>
<td></td>
<td>- I. Inactive. The registration is to be defined but is not to be used by the Collection Agent.</td>
</tr>
<tr>
<td></td>
<td>- A. Active. The registration is to be defined and used by the Collection Agent.</td>
</tr>
</tbody>
</table>

### Table 8-2. Current Capture Registration

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Displays the name of the Capture Registration that has been selected.</td>
</tr>
<tr>
<td>Version</td>
<td>Displays the version number of the Capture Registration.</td>
</tr>
<tr>
<td>Condense</td>
<td>Displays the condense state of the Capture Registration (None, Part or Full).</td>
</tr>
<tr>
<td>Status</td>
<td>Displays the status of the Capture Registration.</td>
</tr>
<tr>
<td>Add</td>
<td>Adds the Capture Registration selected in Available Capture Registrations list to the Current Capture Registrations list.</td>
</tr>
<tr>
<td>Remove</td>
<td>Removes the Capture Registration selected in Current Capture Registrations list.</td>
</tr>
</tbody>
</table>
Specifying Extraction Criteria

There are a variety of values which can be specified when running an extraction process. When using either PWXPC through PowerCenter or the PowerExchange ODBC drivers, these criteria have to be defined as part of the extraction process.

For more information about using PWXPC to extract changes, see PowerExchange Interfaces for PowerCenter. PowerExchange ODBC drivers allows various overrides using escape sequences:

♦ DTLXTYPE=
♦ DTLAPP=
♦ DTLDSN=

<table>
<thead>
<tr>
<th>Type</th>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SL</td>
<td>None</td>
<td>Extract all new captured data since the last extraction.</td>
</tr>
<tr>
<td>RS</td>
<td>-n</td>
<td>Default DTLXTYPE Re-run the last specified extraction process. This is either the last extraction that was run or the one that is specified with the Reset Start Point Wizard in the Application dialog box. For more information about Application Groups, see “Application Groups” on page 141.</td>
</tr>
</tbody>
</table>

For more information, see the PowerExchange Reference Manual.

Adding an Extraction Map to an Existing Extraction Group

1. Right-click the required extraction group, and then click Open.
2. On the toolbar, click the Add Extract Definition icon.
   The Add Extract Definition - Name dialog box appears. For more information about this dialog box, see “Adding Extraction Maps” on page 132.
3. Follow Steps 2 and 3 in “Adding Extraction Groups” on page 130.

Deleting an Extraction Group

To delete an Extraction Group:

1. Select the extraction group, and then click Delete. A confirmation dialog box opens.
2. Click Yes. Deleting the extraction group does not delete the extraction maps. Deleting extraction groups only removes the group from the Windows machine on which the delete is done.

Deleting an Extraction Map

To delete an extraction map:

Select the extraction map, and then click Delete to delete the extraction map. A confirmation dialog box displays.

After the entry is deleted, it is no longer possible to extract any changed data by using that extraction map. Verify that corresponding changes are made to any PowerCenter mappings and sessions.

Viewing an Extraction Map

To view an extraction map:

1. Double-click the extraction group in the Resource Explorer.
2. Select an extraction map, and then double-click the extraction map name to open it.
The following example shows an extraction map for a DB2 for z/OS source:

The following appears:

- **Resource Explorer.** Displays a list of extraction groups and the extraction maps within each group. To select and open an extraction map, double-click on the Extraction Name. All the Extraction details are displayed in the Extract Definition Details dialog boxes.

- **Extract Definition.** It is possible to remove columns from the extraction map by de-selecting them by un-checking the associated box. As a default the automatically generated columns are not displayed. Right click to display these columns. They can be selected as part of the extraction map. Select the File, Save menu option to save any changes.

- **Extraction Details.** Displays the Name, Version and referenced Table of the extraction map.

- **Table Details.** Displays the column information of the table referenced by the extraction map, including: name, type, precision, scale, and length.

You can right-click in the Extract Definition window, and then select **Show Auto Generated Columns.**

The following internal columns display:

<table>
<thead>
<tr>
<th>Column</th>
<th>Description</th>
<th>Data Type</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>DTL__CAPXRESTART1</td>
<td>Coded position of the extracted data in the extraction log.</td>
<td>VARBIN</td>
<td>255</td>
</tr>
<tr>
<td>DTL__CAPXRESTART2</td>
<td>Coded start point for the oldest unit-of-work.</td>
<td>VARBIN</td>
<td>255</td>
</tr>
<tr>
<td>DTL__CAPXUOW</td>
<td>Coded position of the start of the commitment control for the unit-of-work for this update.</td>
<td>VARBIN</td>
<td>255</td>
</tr>
<tr>
<td>DTL__CAPXUSER</td>
<td>The user making the change as recorded by the source software performing the change operation. For DB2 for z/OS sources, the contents of this field varies based upon whether the UIDFMT keyword is specified on the LRAP CAPI_CONNECTION or not. If UIDFMT is not specified, then the field contains only the user ID that performed the operation. For more information about UIDFMT, see CAPI LRAP parameters in PowerExchange Change Data Capture for z/OS.</td>
<td>VARCHAR</td>
<td>255</td>
</tr>
</tbody>
</table>
In addition, null indicator hidden columns are shown adjacent to nullable columns and a length indicator when these are of variable length.

- **Resource Inspector.** The Extraction Group tab displays the connection properties of the extraction map. If you make any changes, click Apply to apply the changes.

## Removing Columns from the Extraction Map

If a field is no longer required by the processing application but it is still present in the source database, the least intrusive change is to remove the field from the extraction map. The view of the underlying data is determined by the capture registration. You can remove columns from the extraction map. PowerExchange capture changes for sources using the capture registration. PowerExchange extracts the captured changes using the extraction map. You do not need to include all captured columns in an extraction map.

After the extraction map has been changed, subsequent extraction processing does not pick up any data from columns that have been deselected.

<table>
<thead>
<tr>
<th>Column</th>
<th>Description</th>
<th>Data Type</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>DTL__CAPXTIMESTAMP</td>
<td>The time the change was made to the source data for the following source types: - DB2 for i5/OS - DB2 for Linux, UNIX, and Windows - DB2 for z/OS - Datacom - IDMS synchronous - IMS synchronous - VSAM - The time the change was read from the source database logs for the following source types: - Adabas - IDMS log-based - IMS log-based - The timestamp has the following format: YYYYMMDDHHMMSSnnnnnn where nnnnnn are microseconds</td>
<td>CHAR</td>
<td>20</td>
</tr>
<tr>
<td>DTL__CAPXACTION</td>
<td>The change capture action. (insert), D(elete) or U(pdate).</td>
<td>CHAR</td>
<td>1</td>
</tr>
<tr>
<td>DTL__CAPXCASDELIND</td>
<td>For DB2 for z/OS sources only, an indicator for delete records generated for tables with references clauses specifying ON DELETE CASCADE. If a dependent row is deleted by DB2 as a result of a cascade delete, this indicator contains Y. Otherwise, the indicator contains N.</td>
<td>CHAR</td>
<td>1</td>
</tr>
<tr>
<td>DTL__BI</td>
<td>The Before Image of the changed field is also retrieved in addition to the normal retrieval of the After Image. For more information about DTL__BI, see “Using Change Indicator and Before Image Columns” on page 139.</td>
<td>Datatype and length of the source column.</td>
<td></td>
</tr>
<tr>
<td>DTL__CI</td>
<td>A change indicator (CI) can be added to each column in which you are interested. For Updates this indicates, on retrieval, whether (Y) or not (N) the column has been changed. Inserts and Deletes always return a null. For more information about DTL__CI, see “Using Change Indicator and Before Image Columns” on page 139.</td>
<td>CHAR</td>
<td>1</td>
</tr>
</tbody>
</table>
To remove columns from an extraction map:

1. Click on the extraction map to highlight it.
2. Right-click on the extraction map, and then click Open.
3. In the Extract Definition window, clear the box beside the column to be removed.

This process does not prevent the data being captured. Selecting the column again at a later date makes that data available again for extraction through this extraction map.

Changing the Capture Registration Associated with an Extraction Map

To change the capture registration associated with an extraction map:

1. Click on the extraction map to highlight it.
2. Right-click on the extraction map, and then click Open.
3. Right-click on the extraction map, and then select Amend Capture Registrations to select a different capture registration.

For information on Amend Change Indicator/Before Image Extensions, see “Using Change Indicator and Before Image Columns” on page 139.

4. Use Amend Capture Registrations if a new version of the capture registration has been created. The following shows how to change the extraction map to use version 2 of the capture registration. Only a single Capture registration can be selected.

Table 8-3 describes Available Capture Registrations list:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Displays the name of the Capture Registrations that are available for selection.</td>
</tr>
<tr>
<td>Version</td>
<td>Displays the version number of the Capture Registration.</td>
</tr>
</tbody>
</table>
Table 8-4 describes Current Capture Registration list:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Displays the name of the Capture Registration that has been selected.</td>
</tr>
<tr>
<td>Version</td>
<td>Displays the version number of the Capture Registration.</td>
</tr>
<tr>
<td>Condense</td>
<td>Displays the condense state of the Capture Registration: None, Part or Full.</td>
</tr>
<tr>
<td>Status</td>
<td>Displays the status of the Capture Registration.</td>
</tr>
<tr>
<td>Add</td>
<td>Adds the Capture Registration selected in Available Capture Registrations list to the Current Capture Registrations list.</td>
</tr>
<tr>
<td>Remove</td>
<td>Removes the Capture Registration selected in Current Capture Registrations list.</td>
</tr>
</tbody>
</table>

Table 8-3. Available Capture Registrations

Now the version 1 capture registration is removed and the version 2 registration added.

5. Click **OK** to save the changed extraction map. The resulting window shows the new version in the middle pane.

Version Indicator in Extraction Maps

The version indicator is used to identify the current version of the extraction map. The extraction map is represented as a data map. The extraction process uses this data map to read and format the data. The qualifier used in the SQL request used by the application to retrieve the changes specifies a data map name, such as the extraction map name.

What happens if the registration changes when a column is added or dropped? The layout of the extracted data changes. If the application retrieves the changes using the old data map, the data might be affected, the results are unpredictable. When a column is added it might just work but this must not be relied upon.

The requesting application is not aware of the version identifier. The way to correct that incompatibility is by using the extraction map process and adjusting the correct version identifier to the extraction map. A wizard guides you through that process.

What other changes are required if the version identifier changes? By making the changes in the extraction process only a part of the changes are completed. If the application is an ETL product the ‘SOURCE’ definition has to be adjusted so that the new columns are recognized or old ones are being dropped. The ‘TARGET’ definitions have to be made aware of the changes in the data layout.

By changing the version identifier a manual server process ensures that there is no more captured data that hasn't been extracted. Using the version identifier it is always possible to extract captured data with the correct data map. This is done by defining an extraction process which specifies the specific version for a given capture file.

Before a registration is changed, all the data for that registration must first have been extracted. The new registration can then be given its new version number and its data can then be extracted.
When defining a new Capture Registration, the PowerExchange Navigator application automatically defines a corresponding extraction map with a version identifier of 1. All further changes to that extraction map have to be made manually for the above-mentioned reasons.

### Using Change Indicator and Before Image Columns

Change indicators enable you to determine if a particular field in the registered table has changed. Use it, for example, in an SQL WHERE clause to retrieve only those columns that have changed rather than the entire table.

To achieve this level of functionality, the extraction data map can be modified to add Change Indicator (CI) and Before Image (BI) columns.

For the After Image (AI), a change indicator (CI) can be added to each column in which you are interested. For Updates this indicates, on retrieval, whether (Y) or not (N) the column has been changed. Inserts and Deletes always return a null.

By adding the Before Image (BI) column, the Before Image of the changed field is also retrieved in addition to the normal retrieval of the After Image.

**Note:** The change indicator field is only set to 'Y' if the update causes the column value to change.

### Selecting CI and BI Columns in an Extraction Map

You must use the PowerExchange Navigator to edit the required extraction map. In the following example, the Change Indicator and Before Image columns are added to the CITY field.

**To select CI and BI columns in an extraction map:**

1. In the PowerExchange Navigator, open the extraction map.
2. Position the cursor in the Extract Definition pane.
3. Right-click the extraction map, and then click **Amend Change Indicator/Before Image Extensions**.
   The Extract Definition - Amend Columns dialog box appears. The Change Indicators tab is active.
4. From the Available Columns list, select the columns that you want to amend (for example, the CITY column) and click **Add**.
   You can also select the CITY column. Right-click and select **Add**.
   The CITY column is moved to the Selected Columns list.
5. Click the **Before Images** tab.
   The Before Images tab is now active:
6. From the Available Columns list, select the columns that you want to amend. For example, select the CITY column and click **Add**.
   You can also select the CITY column. Right-click and select **Add**. The CITY column is moved to the Selected Columns list.
7. Click **OK**.
   The Extract Definition - Amend Columns dialog box is closed.
   **Note:** The new DTL_CI_CITY and DTL_BI_CITY columns are added to the list. The column type of these new columns is CI and BI respectively, and the Real Column field displays the source CITY column. The CITY column displays DTL_CI_CITY in the CI Column field and DTL_BI_CITY in the BI Column field.
8. To display full details of the new fields, click the **Extensions** tab.
Testing the Column Changes Using Database Row Test

Use the PowerExchange Navigator Row Test facility to test the changes that you have made to the extraction map.

In the following example, the record of employee id 3 has been updated to show a new CITY of Shanghai.

To test column changes with row test:

1. In the PowerExchange Navigator, select the extraction map that you require to test.
2. Select **File > Database Row Test**.
3. Ensure that the correct information is entered into the Row Test dialog box.

4. Click **Go**.

The results of the data retrieval are displayed. These represent the changes that have occurred on the table that you registered for data capture.

**Note:** The new Control Indicator (CI) /Before Image (BI) columns are visible in the Row Test results show that:

- The column was changed (**DTL_CI_CITY=Y**)
- The After Image of the record (**CITY=Shanghai**).
- The Before Image of the record (**DTL_BI_CITY=innercity**).
CHAPTER 9

Working with Capture Applications

Application Groups

This function provides the capability for the user to display the information for a given extract application and to allow the user to reset the status of the extractions process to rerun.

This shows an audit trail of all the extractions that have been run, their status and an indication of the captured data that has been moved.

To view the application restart information open the Application Group by right clicking and selecting Open or double clicking the Application Group.

All applications appear in the list below the Application Group and not just those associated with the Registration Group created at the same time. When the applications are open, it shows a history of the change extracts.

The Application Group is based on the same principle as the Extraction and Registration Groups. The Group level identifies the location of the PowerExchange Listener task. There is no need for a database instance type or name. The Resource Inspector shows:

- Name
- Location
- Userid
- Password

The Application Name list is based on the application name on the extraction run.

Clicking on a specific Application Name in the list displays a list of all the extracts (successful or not) in timestamp order, latest one on top. Click the timestamp to display the complete information for that extraction run, including registration name, table name, and timestamps.

Right-clicking on a specific Application Name displays a list of possible actions for that application name.

Note: Informatica recommends that a unique application name be used for each extraction process. Using unique application names for each extraction process ensures that proper restart occurs. Sharing application names with multiple extraction processes can result in errors, extraction of undesired change data, and other unexpected results.

Reset to New Start Point

The Application tracks the extractions against a registration and stores the start point for the next run. There are occasions when one or more extractions need to be rerun and this can be achieved by resetting the start point for the application.
This feature is used to reset the start point of the next extraction run for the selected application. It resets the start point to the start point of the open timestamp.

The application can then either:

♦ Restart at the start of the selected run, or
♦ Be reset again.

This is a powerful feature and cannot be undone. If in doubt, use DTLUAPPL. Tokens can be printed for copy or paste using DTLUCDEP.

The following example uses an application DB2TEST which has had two extractions run against it. This is for demonstration purposes only so the extract times are very close together, however, the application shows how to reset the start point.

**To reset the start point:**

1. Open the required application by double clicking on the application group then the relevant application.

   **Note:** The order of runs shown in the right hand pane. The first and last runs show the time of the two runs and the Current Run is blank. Highlight the run to which the start point is to be reset, in this case the earliest of the two runs is selected.

1. Select the Application Group name, right clicking and selecting **Reset To New Start Point**.

   The icon to the left of the application group turns red. When exiting there a prompt to save appears.

   The restart point is reset to the start point of the first run. When next opening the application it can be seen that the Current Run information is now no longer blank but has been replaced by the run information from the first extraction.
Overview

The purpose of user-defined fields are to add the flexibility of manipulating data from the source and creating value added data maps. You can create fields by using one or more functions defined as expressions:

- Calls to customer written COBOL, PL/1, Assembler, or C programs.
- Calls to PowerExchange internal functions, such as concatenation and field splitting.

Expressions Tab

The Expr(x) tab of the Record window contains the user-defined fields that have been set-up for the current data map. Each row contains a declaration of a field or an assignment of a particular value to a previously declared field. The number x will indicate the number of expressions that have been defined. For example, Expr(0) indicates that none have been defined while Expr(6) indicates that 6 have been defined.
Figure 10-1 shows a user defined field for a Data Map:

**Figure 10-1. Example of User Defined Fields for a Data Map.**

<table>
<thead>
<tr>
<th>Column</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Field status indicator</td>
<td>Field entered correctly and is operable or a problem exists with the field. Read the comment in the Validation column. New field selected (after selecting one of the Add Field … menu options)</td>
</tr>
<tr>
<td>Name</td>
<td>Name given to the user-defined field. This is used to identify the new field.</td>
</tr>
<tr>
<td>Type</td>
<td>Datatype of the new field.</td>
</tr>
<tr>
<td>Prec</td>
<td>Precision of the new field</td>
</tr>
<tr>
<td>Scale</td>
<td>Scale of the new field</td>
</tr>
<tr>
<td>Length</td>
<td>Length of the new field</td>
</tr>
<tr>
<td>Phase</td>
<td>R (Read), W (Write), or RW (Read and Write).</td>
</tr>
<tr>
<td>Expression</td>
<td>You can enter the expression that is used to populate the field.</td>
</tr>
<tr>
<td>Validation</td>
<td>If there is a problem with the field, a message appears.</td>
</tr>
</tbody>
</table>

**Right-Click Menu**

The following menu is available by right-clicking the mouse.

<table>
<thead>
<tr>
<th>Column</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add Field Before …</td>
<td>Adds a new field row above the current item.</td>
</tr>
<tr>
<td>Add Field After …</td>
<td>Adds a new field row below the current item.</td>
</tr>
<tr>
<td>Add Field at End …</td>
<td>Adds a new field row at the bottom of the list.</td>
</tr>
<tr>
<td>Move Field Up</td>
<td>Moves declarations to the top of the list and assignments to the bottom.</td>
</tr>
<tr>
<td>Organise Fields</td>
<td>Moves all completed new fields to the top of the list.</td>
</tr>
<tr>
<td>Delete Field/Cell</td>
<td>Deletes the current cell or field, if the entire row is selected.</td>
</tr>
<tr>
<td>Hide/Show Type Columns</td>
<td>Hides or shows the Type, Pre(cision), Scale, and Length columns.</td>
</tr>
<tr>
<td>Restore Columns</td>
<td>Shows all columns and restores all columns to the default width.</td>
</tr>
</tbody>
</table>

**Declaration**

The row contains the new field name, a datatype with appropriate precision, scale and length values and the phase of database access in which the field will operate (Read, Write or Read/Write).

**Assignment**

The assignment row must contain the name of the previously declared field, the phase of database access in which the field will operate and an expression which assigns a particular value to the field. It is possible to combine the declaration and assignment for a new field by entering an expression on the declaration line.
Expression

The expression determines what action is performed on the field to populate it with data. The expression can be:

- A text or numeric constant, such as 'My text' or 1234.
- A pre-defined PowerExchange function, such as concatenation or field splitting.
- A call to a user-defined external program, such as COBOL, PL/1, Assembler, or C programs.

Phase

The phase is the aspect of database communication for which the user-defined field is appropriate. For example, if a field is defined as having a phase of Read, its expression will not be recalculated when the database is being written to.

Keyboard Shortcuts

The following keyboard shortcuts are available.

<table>
<thead>
<tr>
<th>Key</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Insert</td>
<td>Insert a new row at the bottom</td>
</tr>
<tr>
<td>Delete</td>
<td>Delete field or cell contents depending on whether a cell or row is highlighted</td>
</tr>
<tr>
<td>Enter</td>
<td>Start editing a cell or enter contents from editing cell</td>
</tr>
<tr>
<td>Escape</td>
<td>Leave a cell without making changes</td>
</tr>
<tr>
<td>Ctrl+Up Arrow Key</td>
<td>Moves a field up</td>
</tr>
<tr>
<td>Ctrl+Down Arrow Key</td>
<td>Moves a field down</td>
</tr>
<tr>
<td>Arrow keys</td>
<td>Moves highlighted cell around</td>
</tr>
</tbody>
</table>

Expression Editor Dialog Box

This dialog box allows you to build the expression that is associated with the field. Double-click the appropriate function in the Functions List and then manually edit the parameters to build the expression.

The dialog box is accessed by clicking the Browse button in the Expression column of the Expr(x)s tab on the Record Properties window.
Creating a User-Defined Field

The following procedure enables you to create a new field and associate a custom definition to it.

1. In the Resource Explorer panel select the required data map.
2. Select the record. The Record window appears.
3. Click the Expr(x) tab of the Record window.
4. Right-click and select Add Field at End.
5. Enter a name for the new field in the Name cell.
6. Select the datatype from the Type cell.
7. Enter the Precision, Scale, and Length as appropriate to the datatype selected.
8. Enter the expression that you wish to associate with the field. You can do this by directly typing into the cell or by clicking the Browse button. This displays the Expression Editor dialog box.
   - Double-click the appropriate function and edit the parameters manually in the Expression box as appropriate.
   - Click Validate. You are notified of any error in the expression.
   - Click OK. The expression is associated with the field and the dialog box is closed.

You need to refresh the table view before the user-defined field appears in the Table window.
To do this, select the table in the Resource Explorer, select the Properties... right-menu option and then select the Refresh with missing columns from the Column Generation pull-down list.

Available Functions

The following pre-defined functions are included.

CallProg

Purpose
Calls a user-defined sub-routine or program.

Syntax
CallProg('Module','Routine', 'Linkage', arg1 ..."

Parameters:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Module</td>
<td>Module where the subroutine resides, which is the physical name of the program.</td>
</tr>
<tr>
<td></td>
<td>- On i5/OS, a service program.</td>
</tr>
<tr>
<td></td>
<td>- On MVS, a load module.</td>
</tr>
<tr>
<td></td>
<td>- On Windows, a DLL</td>
</tr>
<tr>
<td></td>
<td>- On UNIX, a shared object</td>
</tr>
<tr>
<td>Routine</td>
<td>Name of entry point.</td>
</tr>
<tr>
<td></td>
<td>- On i5/OS, Windows, and UNIX, this is the name of the subroutine.</td>
</tr>
<tr>
<td></td>
<td>- On MVS linkages for types of Assembler, COBOL, a value must be provided but it is ignored and the default entry point for the load module is used. It is best to repeat the module name.</td>
</tr>
<tr>
<td></td>
<td>- For MVS PLI, it is sometimes possible to have multiple fetchable subroutines in the same load module.</td>
</tr>
<tr>
<td>Linkage</td>
<td>Argument determining the type of linkage which has two factors:</td>
</tr>
<tr>
<td></td>
<td>- the method by which arguments are passed (C uses the stack, other languages pass a list of addresses)</td>
</tr>
<tr>
<td></td>
<td>- what parameters are passed (C and OS gets the failure code from the subroutine return code; other types pass the address of a Failure code integer)</td>
</tr>
<tr>
<td></td>
<td>Supported values are:</td>
</tr>
<tr>
<td></td>
<td>- C on all platforms</td>
</tr>
<tr>
<td></td>
<td>- COBOL on MVS</td>
</tr>
<tr>
<td></td>
<td>- PLI on MVS</td>
</tr>
<tr>
<td></td>
<td>- OS on MVS</td>
</tr>
<tr>
<td></td>
<td>- OS400 on i5/OS</td>
</tr>
</tbody>
</table>

arg1 First argument passed to external subroutine

arg2 Second argument passed to external subroutine

Result:
The optional Result is a NUM 32. 0 indicates success. Non-zero indicates failure.

If no result argument is specified and a non-zero failure code is returned from the external function, then CALLPROG applies default error handling. By default, any non-zero failure code returned from the external sub-routine triggers a map-level response (either; terminate the extract or skip this routine).

Notes
For more information about the programming interface and return code checking for CALLPROG programs, see “Using External Programs with CALLPROG” on page 161.
Check

Purpose
Provides a limited override facility for reporting errors signalled by external programs that have been called using the CALLPROG function.

By default, any non-zero failure code returned from the external sub-routine triggers a map-level response (either; terminate the extract or skip this routine).

The CHECK function can be used to allow specified non-zero return codes to be tolerated.

Syntax
Check(Field1, Message, Comparison, Value1, Value2, ...)

Parameters:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Field1</td>
<td>Field to be checked. It will have received the result from CALLPROG.</td>
</tr>
<tr>
<td>Message</td>
<td>A user-defined message on the log, such as 'Unrecognized name in program XYZ.'</td>
</tr>
<tr>
<td>Comparison</td>
<td>The following types of comparison can be used:</td>
</tr>
<tr>
<td></td>
<td>- EQ. A failure is signalled if the value in Field1 matches any of values in the list.</td>
</tr>
<tr>
<td></td>
<td>- NE. A failure is signalled if the value in Field1 does not match any of values in the list.</td>
</tr>
<tr>
<td>Value1</td>
<td>The first value on the list.</td>
</tr>
<tr>
<td>Value2</td>
<td>The second value on the list.</td>
</tr>
</tbody>
</table>

Any number of further parameters can be added.

Result
No result variable is defined. If the function determines either:

- that the processing for this row should abort or
- continue by setting a system return code.

Example of accepting non-zero return codes
If failure codes of 0, 3, and 4 are required to be acceptable then define:

```plaintext
CallProgRC = CallProg('Program1', 'Program2', 'COBOL', 'Field1', 'Field2')
Check(CallProgRC, 'Error in Program1', NE, 0, 3, 4 )
```

Concat

Purpose
Concatenates two or more fields (real or user-defined). This function could, for example, be used to create a group field from several other fields.

Syntax
Concat(field1, field2, ...)

148 Chapter 10: Working with User-Defined Fields
**Parameters:**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Field1</td>
<td>1st field to be copied. Can be a real or user-defined expression field</td>
</tr>
<tr>
<td>Field2</td>
<td>2nd field to be copied. Can be a real or user-defined expression field</td>
</tr>
</tbody>
</table>

**Example:**

If the fields (field1) contained the text *The quick brown fox* and the field (field2) contained the text *jumped over the lazy dog*, the function `Concat(field1,field2)` would return the text *The quick brown fox jumped over the lazy dog*.

**Result**

The result field is a group field created by copying the input fields together. It must be a User-defined expression field.

**Notes**

- No conversions are done. The number of bytes to copy is determined from the data length.

**Copydata**

**Purpose**

Copies data from the specified field into the new user-defined field.

**Syntax**

CopyData(SourceFieldName)

or

CopyData(SourceFieldName,TargetFieldName)

**Parameters:**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SourceFieldName</td>
<td>Input field to be copied. Can be a real or user-defined expression field</td>
</tr>
<tr>
<td>TargetFieldName</td>
<td>Specifies the target field to copy to (if Result is not specified). The target field can only by a user-defined expression field.</td>
</tr>
</tbody>
</table>

**Result**

The result field is optional. If used, it specifies the target field for the move and the 2nd argument is not given.

**Notes**

- The target field can either be specified in the Result field using prototype the first prototype or it is specified as the second argument as in the second form of syntax shown above.
- When an expression evaluates to a field, a call is made to this function.

**Fragment**

**Purpose**

Splits the contents of an existing field and distributes the result among two or more other existing fields.

**Syntax**

Fragment(SourceField,TargetField1,TargetField2, ...)

AvailableFunctions 149
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SourceField</td>
<td>Field whose data is to be fragmented</td>
</tr>
<tr>
<td>TargetField1</td>
<td>The first target field</td>
</tr>
<tr>
<td>TargetField2</td>
<td>The second target field</td>
</tr>
</tbody>
</table>

Example

Pre-existing fields:

- Field1 (44)
- TargetField1 (15)
- TargetField2 (29)

If the field (Field1) contained the text *The quick brown fox jumped over the lazy dog*, then the function `Fragment(Field1, TargetField1, TargetField2)` would return:

- TargetField1 = "The quick brown"
- TargetField2 = " fox jumped over the lazy dog"

GenVRowKey

Purpose

This function is used with source rows that contain an array. For data rows with an OCCURS clause, a single row is returned for each instance of the OCCURS.

The function returns a generated unique row number for the row which was created from a source row that contained the array.

Syntax

`GenVRowKey()`

Result

NUM 32U

Example

Assume you have a data map with an array defined by a COBOL OCCURS clause as follows:

```
01 RECORD.
  03 FIELD1 PIC X(4).
  03 FIELD2 PIC X(4) OCCURS 3 TIMES.
```

For 2 source rows containing the following data:

- AAAA111122221111
- BBBB555566667777

the following six table rows will be generated:

- AAAA 1111 1
- AAAA 2222 2
- AAAA 1111 3
- BBBB 5555 1
- BBBB 6666 2
- BBBB 7777 3

Note that this allows you to create unique instances of the two rows containing the value 'AAAA 1111'.
**GetDatabaseKey**

**Purpose**
Returns the internal database key for the associated record instance.

**Syntax**

```plaintext
GetDatabaseKey()
```

**Result**

<table>
<thead>
<tr>
<th>Applicable Data Source</th>
<th>Type</th>
<th>Key object returned</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADABAS</td>
<td>NUM32</td>
<td>Internal Sequence Number (ISN)</td>
</tr>
<tr>
<td>ESDS</td>
<td>BIN 8</td>
<td>Extended Relative Byte Address (XRBA)</td>
</tr>
<tr>
<td>i5/OS</td>
<td>NUM32</td>
<td>Relative Record Number (RRN)</td>
</tr>
<tr>
<td>IDMS</td>
<td>NUM32</td>
<td>Database key</td>
</tr>
<tr>
<td>IMS</td>
<td>BIN 8</td>
<td>Relative Byte Address (RBA)</td>
</tr>
<tr>
<td>ODBA</td>
<td>BIN 8</td>
<td>Relative Byte Address (RBA)</td>
</tr>
<tr>
<td>RRDS</td>
<td>NUM32</td>
<td>Relative Record Number (RRN)</td>
</tr>
</tbody>
</table>

**Notes**

GetDbKey can be used as an alias for this function when being used with IDMS.

IMS Log-based Change Capture is not currently supported.

**GetDataFlowType**

**Purpose**

This describes the type of record returned. This function can be used with both PowerExchange Bulk and Capture processing.

When used with GetDbKeyOfOwner a relational view of the IDMS data can be generated, so records will show their owner record key when they are written out. This data could be loaded into a relational database using PowerExchange Bulk processing, and then kept current using PowerExchange Capture.

**Syntax**

```plaintext
GetDataFlowType()
```

**Result**

The type of data included in the returned row. The result argument must be a CHAR 2.

The first character can be:
- **B** = Bulk data extraction
- **L** = Log capture data extraction
- **R** = Real-time capture data extraction.

The second character can be:
- **D** = Normal data record
- **C** = Connect without data (relevant for capture only)
- **U** = Disconnect without data (relevant for capture only)

When the second character returned is a C or U the data portion of the record returned will be filled with null values.

If there is no owner, the value that was returned by GetDbKeyOfOwner is set to high values.
Notes
This function is only relevant for IDMS data access.

This function will only produce results for a single record capture registration or for the base record (lowest level) of a multiple record registration.

For owner key functionality only the database key is available, the page group and radix elements will always be set to zero.

GetDbKey

Alias to GetDatabaseKey

GetDbKeyOfOwner

Purpose
This returns the database key of the specified owner record. This function must be used only where there is no possibility of duplicate database keys because this function will not return either the IDMS page group or radix. If this is not the case you may need to use GetFullDbKeyOfOwner.

Use of the GetDbKeyOfOwner function enables the specified owner record to be identified without physically reading that record.

Syntax
GetDbKeyOfOwner('\<SETNAME\>')</n
Result
The database key. The result argument must be a NUM32U or BIN4.

Notes
This function is only relevant for IDMS data access.

GetFullDbKey

Purpose
This returns the fully qualified database key of the IDMS record. The use of the fully qualified option may be a requirement when duplicate database keys are present in the database. The unique record is identified by the addition of the four byte concatenated page group and radix identifier prefix.

This will add a four byte overhead to the records passed.

Syntax
GetFullDbKey()

Result
The database key. The result argument must be a BIN8.

Notes
This function is only relevant for IDMS data access.
GetFullDbKeyOfOwner

**Purpose**
This returns the fully qualified database key of the owner record. The use of the fully qualified option may be a requirement when duplicate database keys are present in the database. The unique record is identified by the addition of the concatenated four byte page group and radix identifier prefix.

This will add a four byte overhead to the records passed.

**Syntax**
GetFullDbKeyOfOwner('<SETNAME>')

**Result**
The database key. The result argument must be a BIN 8.

**Notes**
This function is only relevant for IDMS data access.
This function cannot be used with a setname which represents a system index.

GetIMSRBAByLevel

**Purpose**
Returns the RBA value of the requested segment's.

**Syntax**
GetIMSRBAByLevel(integer)

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
</table>
| integer | The following types are possible:  
- Not supplied, such as GetIMSRBAByLevel()  
  - If you do not specify a number, the result will be the RBA of the current segment  
- Positive integer, such as GetIMSRBAByLevel(2)  
  - Input is the 4-byte integer level number of the segment in the IMS hierarchy whose RBA you wish to access.  
    - The level number is limited to the level number of the current segment in the hierarchy, or any of that segments ancestors. In other words, if you are dealing with a segment at level 3 of a 5-level hierarchy, you may request the RBA of segments at level 1, 2, or 3.  
    - Attempting to request the RBA of segments in the hierarchy that are below the level of the current segment will result in a run time error.  
- Negative integer, such as GetIMSRBAByLevel(-2)  
  - You may also make a relative request by providing a negative number as input.  
    - For instance, you may request the RBA of the segment two levels above the level of the current segment. You do this by specifying -2.  
    - An attempt to specify a negative number whose absolute value is greater than or equal to the level number of the current segment will result in a run time error. |

**Result**
Output of GetIMSRBAByLevel is an 8-byte binary representation of the requested segment's RBA value. 8 bytes are used to allow for future expansion and for a uniqueness guarantee for FastPath and HALDB databases.

**Notes**
GetIMSRBAByLevel is a standard PWX field, and can be used for Bulk and synchronous Change Capture IMS maps.
IMS Unload files and IMS Log-based Change Capture are not currently supported.
GetPageGroup

Purpose
Returns the Page Group within which the IDMS record exists.

Syntax
GetPageGroup()

Result
The Page Group. The result argument must be a Num16, Num16U, or Bin(2).

Notes
This function is only relevant for IDMS data access.

GetPgGrpAndRdx

Purpose
Returns the Page Group and Radix of the IDMS record.

Syntax
GetPgGrpAndRdx()

Result
The concatenated Page Group and Radix. The result argument must be a Num32, Num32U, or Bin(4).

Notes
This function is only relevant for IDMS data access.

GetPgGrpOfOwner

Purpose
Returns the Page Group of the Owner Record.

Syntax
GetPgGrpOfOwner('<SETNAME>')

Result
The Page Group of the owner record. The result argument must be a Num16, Num16U, or Bin(2).

Notes
This function is only relevant for IDMS data access.

GetRadix

Purpose
Returns the Radix of the IDMS record.

Syntax
GetRadix()

Result
The Radix. The result argument must be a Num16, Num16U, or Bin(2).
GetRadixOfOwner

**Purpose**
Returns the Radix of the Owner Record.

**Syntax**

```
GetRadixOfOwner('<SETNAME>')
```

**Result**
The Radix of the owner record. The result argument must be a Num16, Num16U, or Bin(2).

**Notes**
This function is only relevant for IDMS data access.

GetPgGrpAndRdxOfOwner

**Purpose**
Returns the Page Group and Radix of the Owner record.

**Syntax**

```
GetPgGrpAndRdxOfOwner('<SETNAME>')
```

**Result**
The concatenated Page Group and Radix of the owner record. The result argument must be a Num32, Num32U, or Bin(4).

**Notes**
This function is only relevant for IDMS data access.

GetSeqWithinLevel

**Purpose**
Returns the record sequence number within the current hierarchical level.

**Syntax**

```
GetSeqWithinLevel()
```

**Result**
The record sequence number. The result argument must be a Num32.
Example:

```
  Parent 1
    Child 1
    Child 2
  Parent 2
    Child 3
    Child 4
  Parent 3
  Parent 4
    Child 5
    Child 6
    Child 7
```

For the above scenario the function GetSeqWithinLevel returns:

```
1,2,3,4,0,5,6,7
```

That is:

- Children 1 and 2 from Parent 1
- Children 3 and 4 from Parent 2
- Zero (no children) from Parent 3
- Children 5, 6 and 7 from Parent 4

GetSeqWithinParent

**Purpose**

Returns the record sequence number relative to the current parent record instance.

**Syntax**

```
GetSeqWithinParent()
```

**Result**

The record sequence number. The result argument must be a NUM32.

Example:

```
  Parent 1
    Child 1
    Child 2
  Parent 2
    Child 3
    Child 4
  Parent 3
  Parent 4
    Child 5
    Child 6
    Child 7
```

For the above scenario the function GetSeqWithinParent returns:

```
1,2,1,2,0,1,2,3
```

That is:

- Children 1 and 2 from Parent 1
- Children 1 and 2 from Parent 2
- Zero (no children) from Parent 3
LengthOf

**Purpose**
Returns the length of the given field.

**Syntax**
LengthOf(fieldname)

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fieldname</td>
<td>The NAME Of the field whose length is being retrieved</td>
</tr>
</tbody>
</table>

**Result**
The length of the given field. The result argument must be a NUM32.

**Notes**
This function is only relevant for field types where the data length varies, such as STRING, VARCHAR, or CHAR.

**Example**
If the field (field1) contained the text brown fox, the function LengthOf(field1) would return 9.

LTrim

**Purpose**
Returns a CHAR value which contains the original field value with all character values stripped from the left hand end.

**Syntax**
LTrim (field, character)

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Field</td>
<td>The field to be trimmed.</td>
</tr>
<tr>
<td>Character</td>
<td>The character parameter is a single character which must be enclosed in single quotes. If no character parameter is entered, then a 'space' is the default.</td>
</tr>
</tbody>
</table>

**Result**
CHAR

**Example:**
If the field Text contained the text $$$$$$$The quick brown fox, the function LTrim(Text,'$') would return the text The quick brown fox.

RTrim

**Purpose**
Returns a CHAR value which contains the original field value with all character values stripped from the right hand end.
**Syntax**

\[ \text{RTrim} \left( \text{field}, \text{character} \right) \]

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Field</td>
<td>The field to be trimmed.</td>
</tr>
<tr>
<td>Character</td>
<td>The character parameter is a single character which must be enclosed in single quotes. If no character parameter is entered, then a ‘space’ is the default.</td>
</tr>
</tbody>
</table>

**Result**

CHAR

**Example:**

If the field Text contained the text \text{The quick brown fox***************}, the function \text{LTrim(Text,'*')} would return the text \text{The quick brown fox}.

**Split**

**Purpose**

Returns the portion of the given field that is delimited by the start point and number of characters from the start point.

**Syntax**

\[ \text{Split} \left( \text{Field,Start,Number_Bytes} \right) \]

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Field</td>
<td>Field whose data length is being retrieved</td>
</tr>
<tr>
<td>Start</td>
<td>First byte to copy, where numbering starts at 1</td>
</tr>
<tr>
<td>Number_Bytes</td>
<td>Number of bytes to copy</td>
</tr>
</tbody>
</table>

**Result**

Result argument gets a subset of bytes from Field, starting at the Start byte.

**Notes**

This function can be used, for example, to get a single field from a group field. The numeric arguments must be \text{NUM32} field types.

The Split function does not cater for datatype changes. For example, it may be necessary to split off part of a source character field to create a numeric target field. You should use the Split function on the source field to create a new character field; then use the Copydata function on the newly-created character field to create the target field with a numeric datatype.

**Example**

If the field (field1) contained the text “The quick brown fox jumped over the lazy dog”, then the function Split (field1, 17, 10) would return the text “fox jumped”.

158 Chapter 10: Working with User-Defined Fields
Strip

**Purpose**
Returns a CHAR value which contains the original field value with all character values stripped from either the left hand end, right hand end or both ends depending on the side parameter.

**Syntax**
Strip (field, side, character)

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Field</td>
<td>The field to be trimmed.</td>
</tr>
<tr>
<td>Side</td>
<td>The side parameter is L, R, B, or space which must be enclosed in single quotes. If space is selected, then B is the default.</td>
</tr>
<tr>
<td>Character</td>
<td>The character parameter is a single character which must be enclosed in single quotes. If no character parameter is entered, then a 'space' is the default.</td>
</tr>
</tbody>
</table>

**Result**
CHAR

**Example:**
If the field Text contained the text **************The quick brown fox**************
Strip(Text, 'R', '*') would return the text **************The quick brown fox.
Strip(Text, 'L', '*') would return the text The quick brown fox**************
Strip(Text, 'B', '*') would return the text The quick brown fox.

SetBitA

**Purpose**
Returns a CHAR value of Y or N depending on the bit setting of field at the offset position.
If the bit is set 'on' then Y is returned, if the bit is set 'off' then N is returned.

**Syntax**
SetBitA (field, offset)

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Field</td>
<td>The field to be checked.</td>
</tr>
<tr>
<td>Offset</td>
<td>The position within the field of the bit that is being checked. The offset commences at bit 0 and will end at bit 7 for a 1 byte field, 15 for 2 bytes, etc.</td>
</tr>
</tbody>
</table>

**Result**
CHAR

SetBitN

**Purpose**
Returns a numeric (NUM8) value of 1 (on) or 0 (off) on the bit setting of field at the offset position.
Optionally, you can override the result by setting the optional parameters `on_value` and `off_value`. For example you could set an `on_value` to 3 and an `off_value` to 2. If the bit being tested is set ‘on’ then `on_value` (3) is returned rather than 1; if the bit is set ‘off’ then `off_value` (2) is returned rather than 0.

**Syntax**

```
SetBitN (field, offset, on_value, off_value)
```

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Field</td>
<td>The field to be checked.</td>
</tr>
<tr>
<td>Offset</td>
<td>The position within the field of the bit that is being checked (1 to 8).</td>
</tr>
<tr>
<td>on_value</td>
<td>Set to override the default bit value of 1. Returned if the bit being checked is set ‘on’. Optional.</td>
</tr>
<tr>
<td>off_value</td>
<td>Set to override the default bit value of 0. Returned if the bit being checked is set ‘off’. Optional.</td>
</tr>
</tbody>
</table>

**Result**

NUM8

**ToLower**

**Purpose**

Returns a lower case version of the contents of the field.

**Syntax**

```
ToLower (fieldname)
```

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fieldname</td>
<td>The field to be converted. Valid only for CHAR and VARCHAR fields.</td>
</tr>
</tbody>
</table>

**Result**

CHAR and VARCHAR

**ToUpper**

**Purpose**

Returns an upper case version of the contents of the field.

**Syntax**

```
ToUpper (fieldname)
```

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fieldname</td>
<td>The field to be converted. Valid only for CHAR and VARCHAR fields.</td>
</tr>
</tbody>
</table>

**Result**

CHAR and VARCHAR
Using External Programs with CALLPROG

You can use the CALLPROG function to call user-written subroutines or programs. For example, you can use CALLPROG to invoke user-written programs to decode or supplement data in specific fields.

Use the following syntax for the CALLPROG expression:

```
CallProg('Module', 'Routine', 'Linkage', 'arg1', 'arg2', ...)
```

Invoking External Programs

The function address for the entry point of the external routine is loaded on the first call. The load may fail for the following reasons:

- The program is not found in the standard order of search for programs on the platform.
- The program is not of the required type:
  - i5/OS: service program
  - MVS: load module
  - Windows: DLL
  - Linux and UNIX: a shared object
- The module handle is obtained but the routine is not found within it. This can indicate platform-specific linking or symbol export problems such as:
  - i5/OS: symbol for routine not exported
  - MVS C or PL/I: routine not defined with #pragma fetchable
  - Windows C: routine not exported
  - On some platforms, routines are case-sensitive.

Parameter List Passed to External Programs

PowerExchange passes the following parameters to programs invoked by the CALLPROG function:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NbrFlds</td>
<td>Number of fields. An integer specifying the number occurrences of fields in the arrays ppData[] and ppData[]. Input usage only. If 2 fields are being passed and the expression entered by the Navigator is CallProg('Module', 'Routine', 'Linkage', Field1, Field2) then NbrFlds is set to 2. The usage of fields int NbrFlds, char *ppData[], int * pDataLen[] is similar to the C usage of main(int argc, char *argv[]).</td>
</tr>
<tr>
<td>pMsgBuffer</td>
<td>Message Buffer. A pointer to character string where the external routine can pass back a NULL terminated string to describe an error, which accompanies the setting of the return code to a non-zero value. Output usage only. The message buffer will always be empty on every call to the external function. If no error is met, the external function should leave it untouched.</td>
</tr>
</tbody>
</table>
Result Argument and Error Handling

By default, the result argument from CALLPROG does not need to be defined on the Navigator screen. In this case default error handling applies, that is, any non-zero failure code returned from the external sub-routine triggers a map-level response. The map-level response is either terminate the extract or skip this routine.

The function CHECK provides a limited facility to override the errors that are reported. For example, if failure codes of 0, 3, and 4 are acceptable, then define:

```c
CallProgRC = CallProg('Program1', 'Program2', 'COBOL', 'Field1', 'Field2')
Check(CallProgRC, 'Error in Program1', NE, 0, 3, 4 )
```

Return codes

- If the external routine completes successfully, it should return 0.
- In the event of a failure, it should return a non-zero value that can be positive or negative.

By default, any non-zero failure code returned from the external sub-routine triggers a map-level response (either; terminate the extract or skip this routine).
External C Routines

PowerExchange provides sample C code for a CALLPROG program in the UCPE member of the hlq.SRCLIB library, where hlq is the high-level qualifier that you specified during PowerExchange installation.

Platform Support

Use linkage type C in the third argument to CALLPROG expression to link to a C language program.

PowerExchange supports external C routines on all platforms and has the following requirements:
- On i5/OS, routines must be exported routines in service programs.
- On MVS, routines must be fetchable from a load module. Called programs must be linked with AMODE(31). If access is required to modules linked AMODE(24), the routine must be called through an AMODE(31) program that relocates the parameters below the 16 M B line and then calls the AMODE(24) load module.
- On Windows, routines must be exported routines in the CDECL convention in a DLL.
- On Linux and UNIX, routines must be exported routines in an executable shared object.

C Function Prototype

```c
int CRoutine(int NbrFlds, char *pMsgBuffer, int *pMsgBufferSize, char **pData[], int *pDataLen[]);
```

For more information about the parameters passed to programs invoked from CALLPROG, see “Parameter List Passed to External Programs” on page 161.

External COBOL Routines

PowerExchange provides sample COBOL code for a CALLPROG program in the UCPEC and UCPEC01 members of the SRCLIB library.

Platform Support

Use linkage type COBOL in the third argument to CALLPROG expression to link to a COBOL language program. COBOL programs can be used on MVS only.

COBOL programs should be compiled using the Language Environment (LE) COBOL run-time routines. Results are unpredictable for COBOL programs compiled with non-LE COBOL run-time routines.

COBOL Linkage

In COBOL, the first four arguments are always the same and are mandatory.

The remaining fields vary according to the needs of the external program. The attributes of the user-defined expression fields must match the parameters that the external program expects.

Example of COBOL Linkage

```cobol
003700 LINKAGE SECTION.
003800
003900 01 NUMBER-FIELDS PIC S9(9) COMP.
003901
003902 01 FAILURE-CODE PIC S9(9) COMP.
003903
003904 01 MESSAGE-BUFFER.
003905      05 MESSAGE-BUFFER-BYTE PIC X(1) OCCURS 1 TO 128 DEPENDING ON MESSAGE-BUFFER-LENGTH.
003906          01 MESSAGE-BUFFER-LENGTH PIC S9(9) COMP.
003910
004000 01 TEXT-AREA.
004010      05 TEXT-AREA-BYTE PIC X(1) OCCURS 15.
```
In the sample program UPEC, two fields are passed to the COBOL program.

- The first field is called TEXT and has a maximum size of 15 bytes. Data can be moved from and to it using COBOL field TEXT-AREA. If it is not a fixed-length CHAR field, then the TEXT-AREA-LENGTH must be used to determine the actual length. If the COBOL program wants the length to change, it must store the required value in field TEXT-AREA-LENGTH.

- The second field is numeric, which in the Navigator is defined as NUM32. Data can be moved from and to it using COBOL field NUMBER1. It would be unusual to want to make use of field NUMBER1-LENGTH.

If the field is not nullable, then the field would contain value 4 on entry to the COBOL program which should leave the value untouched. NUMBER1-LENGTH is set to zero if the field is NULL. This might happen if the field was NULL before the program was called. If the program decides to make the field NULL, it move zero to NUMBER1-LENGTH.

**External OS Routines**

PowerExchange provides sample Assembler code for a CALLPROG program in the UCPEA member of the SRCLIB library.

**Platform Support**

Use linkage type OS in the third argument to CALLPROG expression to link to an Assembler language program. Assembler language programs can be used on MVS only.

Assembler programs receive a list of parameter addresses in a parameter list pointed to by Register 1. An Assembler program should place the return code in register 15.

**Assembler Linkage**

In Assembler, the first three arguments are always the same and are mandatory. Usage of the arguments and return code is similar to that used for C and COBOL routines.

The definitions for the fields vary according to the number of fields and their types.

**Example Linkage**

```
L     R3,0(R1)                get address of argument 1
L     R4,0(R3)                get NumberFields value
C     R4,='P'2'               Required value of 2 ?
BNE   BADARGS

*----------------------------------------------------------------------
* Get arguments
*----------------------------------------------------------------------

NBRARGOK DS     0H
L     R3,4(R1)                get address of argument 2
ST     R3,AMSGBBFF            = address of message buffer
L     R3,8(R2)                get address of argument 3
ST     R3,AMSGBBFSZ           = size of message buffer
L     R3,12(R1)               get address of argument 4
ST     R3,ATEXT               = address of text argument
L     R3,16(R1)               get address of argument 5
ST     R3,ATEXTLEN            = length of text argument (15)
L     R3,20(R1)               get address of argument 6
ST     R3,NUMBER              = address of number
L     R3,24(R1)               get address of argument 7
```
Using External Programs with CALLPROG

External OS400 Routines

PowerExchange provides sample RPG code for a CALLPROG program in the UCPERPGLE member of the dtllib/RPGLE, where dtllib is the PowerExchange software library that you specified during PowerExchange installation.

Platform Support

PowerExchange supports C and CL programs on the i5/OS.

Use linkage type OS400 in the third argument to CALLPROG expression to link to an non-C language programs. The OS400 linkage type is required for any language other than C, i5/OS, such as CL, COBOL, PL/I, and RPG. Linkage type OS400 can be used on i5/OS only.

CL Linkage

The first four arguments are always the same and are mandatory. Usage of the arguments and return code is similar to that used in the other languages.

Example CL Linkage

```
PGM PARM(&NBRFLDS &RC &MSGBUFF &MSGBUFFSZ &TEXT &TEXTLEN &NUM &NUMLEN)

DCL        VAR(&NBRFLDS) TYPE(*CHAR) LEN(4)
DCL        VAR(&RC) TYPE(*CHAR) LEN(4)
DCL        VAR(&MSGBUFF) TYPE(*CHAR) LEN(128)
DCL        VAR(&MSGBUFFSZ) TYPE(*CHAR) LEN(4)
DCL        VAR(&TEXT) TYPE(*CHAR) LEN(15)
DCL        VAR(&TEXTLEN) TYPE(*CHAR) LEN(4)
DCL        VAR(&NUM) TYPE(*CHAR) LEN(4)
DCL        VAR(&NUMLEN) TYPE(*CHAR) LEN(4)
```

In the example program, two fields are passed to the CL program.

The first field is called TEXT that has a fixed length of 15. Data can be moved to it and from it using the 5th argument &TEXT. If it is not a fixed-length CHAR field, then the data length must be used to determine the actual length i.e. field &TEXTLEN.

Unfortunately, the CL language does not support integers directly. To get the numeric value, the contents of &TEXTLEN must be moved to a packed decimal using a statement like:

```
CHGVAR     VAR(&DECTEXTLEN) VALUE(%BIN(&TEXTLEN 1 4))
```

If the length is changed by the program then the new length has to be moved into the TEXTLEN field using a statement like:

```
CHGVAR     VAR(%BINARY(&TEXTLEN)) VALUE(&DECTEXTLEN)
```
External PL/I Routines

PowerExchange provides sample PL/I code for a CALLPROG program in the UCPEP member of the SRCLIB library.

Platform Support

Use linkage type PLI in the third argument to CALLPROG expression to link to a PL/I language program. PL/I language programs can be used on MVS only. Use a C interface layer to call PL/I programs on Windows and UNIX. Use linkage type OS400 to call PL/I programs on i5/OS.

PL/I programs should be compiled using the Language Environment (LE) PL/I run-time routines. Results are unpredictable for PL/I programs compiled with non-LE PL/I run-time routines.

PL/I Linkage

The first four arguments are always the same and are mandatory. The definitions for the fields vary according to the number of fields and their types.

Example Linkage

```pli
PROC (NUMBER_ARGUMENTS,
     FAILURE_CODE,
     MESSAGE_BUFFER_PTR,
     MESSAGE_BUFFER_LENGTH,
     TEXT_AREA_PTR,
     TEXT_AREA_LENGTH,
     NUMBER1,
     NUMBER1_LENGTH)
OPTIONS (FETCHABLE)
REORDER;

/*---------------------*/
/* LINKAGE FROM CALLER */
/*---------------------*/
DCL NUMBER_ARGUMENTS FIXED BIN(31);
DCL FAILURE_CODE FIXED BIN(31);

DCL MESSAGE_BUFFER_PTR PTR;
DCL 1 MESSAGEBUFFER_STR BASED (ADDR(MESSAGE_BUFFER_PTR)), CHAR(255);
DCL MESSAGE_BUFFER_LENGTH FIXED BIN(31);

DCL TEXT_AREA_PTR PTR;
DCL 1 TEXTAREA_STR BASED (ADDR(TEXT_AREA_PTR)), CHAR(15);
DCL TEXT_AREA_LENGTH FIXED BIN(31);

DCL NUMBER1 FIXED BIN(31);
DCL NUMBER1_LENGTH FIXED BIN(31);
```

In the example program, two fields are passed to the program.

- The first field is called TEXT and has a maximum size of 15 bytes. Data can be moved from and to it using PL/I field TEXT-AREA. If it is not a fixed-length field, then the TEXT-AREA-LENGTH must be used to determine the actual length. If the program wants the length to change, it must store the required value in field TEXT-AREA-LENGTH.
- The second field is numeric, which in the Navigator is defined as NUM32. Data can be moved from and to it using field NUMBER1. It would be unusual to want to make use of field NUMBER1-LENGTH.

If the field is not null-able, then the field would contain the value 4 on entry to the PL/I program, which should leave the value untouched.

NUMBER1-LENGTH is set to zero if the field is NULL. This might happen if the field was NULL before the program was called. If the program decides to make the field NULL, it move zero to NUMBER1-LENGTH.
Using SQL with User-Defined Fields

You should not use the SQL keyword DISTINCT if the data map includes user-defined fields. Attempting to use DISTINCT with user-defined fields causes the following message in the log file:

```
SELECT DISTINCT not guaranteed with Expressions.
```

If DISTINCT is essential to a query then the best approach is to create another table in the data map that contains only the fields that are actually required for that query.

For more information about using SQL in data maps, see the PowerExchange Reference Manual.
Overview

The custom user access method program has a single entry point and is passed a parameter list with every call. The parameter list contains all the parameters specified in the PowerExchange Navigator's data map. The Access module is called repeatedly to process the file. For more information, see “Explanation of Parameter Structure” on page 171.

To show how simple a custom access program can be, the following notes are meant to show in high-level or pseudo-code how the program will be written to call a user program called DECOMP, which just happens to read records and decompress the data.

PowerExchange will make three calls to the custom access program:

<table>
<thead>
<tr>
<th>Call Type</th>
<th>Custom Access Program Action</th>
</tr>
</thead>
</table>
| OPEN      | Get Storage for row to be returned in  
- connect to and open data file  
- return with success or fail to PowerExchange |
| READ      | Start or continue sequential reading  
- read the data file to retrieve next record  
- make data available to PowerExchange Navigator  
- return to PowerExchange with return code (0, 255 or other) |
| CLOSE     | Release acquired storage  
- close data file  
- return to PowerExchange |
Explanation of Input and Return Parameter

User Program Input Parameter

PowerExchange Navigator provides a single parameter that locates the parameter structure. For more information, see “Explanation of Parameter Structure” on page 171.

Return Codes

The user program returns a function return code. The PowerExchange Navigator expects one of the following return codes:

<table>
<thead>
<tr>
<th>Return Code</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Mode completed successfully, such as OPENed correctly.</td>
</tr>
<tr>
<td>255</td>
<td>No more records available to read, such as end of file.</td>
</tr>
<tr>
<td>other</td>
<td>Mode failed.</td>
</tr>
</tbody>
</table>
# Explanation of Parameter Structure

<table>
<thead>
<tr>
<th>Variable name</th>
<th>Description</th>
<th>PowerExchange Navigator User Interface</th>
<th>Type/Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>LGMODE</td>
<td>Used in connection with LGCODE VALUE 1. and indicates how the data file should be opened. 1 = Read 2 = Write 3 = Update 4 = Append</td>
<td>Not applicable</td>
<td>Binary/Word</td>
</tr>
<tr>
<td>LGCODE</td>
<td>The function to be performed by the USER program. 1 = Open 2 = Close 3 = Read 4 = Write</td>
<td>Not applicable</td>
<td>Binary/Word</td>
</tr>
<tr>
<td>LGUI1</td>
<td>User-defined parameter, for example, record length or number of rows to be retrieved.</td>
<td>Integer 1 (USER Access Method dialog box)</td>
<td>Binary/Word</td>
</tr>
<tr>
<td>LGUI2</td>
<td>User-defined parameter, for example, record length or number of rows to be retrieved.</td>
<td>Integer 2 (USER Access Method dialog box)</td>
<td>Binary/Word</td>
</tr>
<tr>
<td>LGREADCT</td>
<td>Number of records read from the data file.</td>
<td>Not applicable</td>
<td>Binary/Word</td>
</tr>
<tr>
<td>LGWRITCT</td>
<td>Number of records written to the data file.</td>
<td>Not applicable</td>
<td>Binary/Word</td>
</tr>
<tr>
<td>LGMLRECL</td>
<td>Record length of the retrieved data.</td>
<td>Set by User program</td>
<td>Binary/Word</td>
</tr>
<tr>
<td>LGU1LEN</td>
<td>Length of the data entered in String 1</td>
<td>String 1 (USER Access Method dialog box)</td>
<td>Binary/Word</td>
</tr>
<tr>
<td>LGU2LEN</td>
<td>Length of the data entered in String 2</td>
<td>String 2 (USER Access Method dialog box)</td>
<td>Binary/Word</td>
</tr>
<tr>
<td>LGFNLLEN</td>
<td>Length of the data file name</td>
<td>File Name (USER Access Method dialog box)</td>
<td>Binary/Word</td>
</tr>
<tr>
<td>LGUIDLEN</td>
<td>Length of the User ID</td>
<td>User ID (on Data Map Remote Node dialog box)</td>
<td>Binary/Word</td>
</tr>
<tr>
<td>LGPWDLLEN</td>
<td>Length of the Password</td>
<td>Password (on Data Map Remote Node dialog box)</td>
<td>Binary/Word</td>
</tr>
<tr>
<td>LGSQLLEN</td>
<td>Length of SQL string</td>
<td>SQL Statement (on Database Row Test dialog box)</td>
<td>Binary/Word</td>
</tr>
<tr>
<td>LGU1PTR</td>
<td>Pointer to the data stored for String 1</td>
<td>String 1</td>
<td>Pointer/Word</td>
</tr>
<tr>
<td>LGU2PTR</td>
<td>Pointer to the data stored for String 2</td>
<td>String 2</td>
<td>Pointer/Word</td>
</tr>
<tr>
<td>LGFN PTR</td>
<td>Pointer to the data stored for the File Name</td>
<td>File Name</td>
<td>Pointer/Word</td>
</tr>
<tr>
<td>LGUIDPTR</td>
<td>Pointer to the data stored for the User ID</td>
<td>User ID (on Data Map Remote Node dialog box)</td>
<td>Pointer/Word</td>
</tr>
<tr>
<td>LGPWD PTR</td>
<td>Pointer to the data stored for the Password</td>
<td>Password (on Data Map Remote Node dialog box)</td>
<td>Pointer/Word</td>
</tr>
<tr>
<td>LGSQL PTR</td>
<td>Pointer to the data stored for SQL string</td>
<td>SQL Statement (on Database Row Test dialog box)</td>
<td>Pointer/Word</td>
</tr>
</tbody>
</table>
Example Programs

Example programs are supplied for Assembler, C, COBOL and PL/1. Each example demonstrates the following logic.

1. The program receives an OPEN call. For the OPEN call, the custom access module might want to perform some or all of the following actions:
   - Obtain working storage.
   - Validate and save any passed parameters.
   - Initialize counters.
   - Check the file exists for reading and open it.
   - Erase the output file if the target is to be replaced.

2. The program receives a READ or WRITE call.
   - **READ call.** On a read call, the access module passes data back to the calling routine by setting the record pointer to point to the data and set the record_len field to the length of the data being returned. When the last record has been processed, the custom access module returns a return code of 255 to signal end of file. Any return code other than 0 and 255 is treated as an error. It is the responsibility of the access module to read the file or database and build the input record. The custom access module might need to examine the SQL statement passed from PowerExchange or the ETL tool to create the desired record.
   - **WRITE call.** When writing to a file, the data is passed by the record pointer and the length is set in the record_len field. The access module can do whatever it deems necessary with the data. A 0 return code signals a successful write.

3. Finally, the access module receives a CLOSE call. During CLOSE processing, the program might perform some or all of these tasks:
   - Close the file.
   - Release storage.
   - Commit the database.
   - Set a return code. If all tasks complete successfully, the module returns a return code of 0. However, if an error occurs, the module sets a non-zero return code, and places message text in the message buffer. The message buffer holds a 79 character message.
   - Pass control back to the calling program.

<table>
<thead>
<tr>
<th>Variable name</th>
<th>Description</th>
<th>PowerExchange Navigator User Interface</th>
<th>Type/Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>LGPUWK1</td>
<td>Pointer to area of storage for user-defined data</td>
<td>Set by User program</td>
<td>Binary/Word</td>
</tr>
<tr>
<td>LGPUWK2</td>
<td>Pointer to area of storage for user-defined data</td>
<td>Set by User program</td>
<td>Binary/Word</td>
</tr>
<tr>
<td>LGOPENRC</td>
<td>Return code of OPEN function 1 = Successful 0 = Unsuccessful</td>
<td>Set by User program</td>
<td>Binary/Word</td>
</tr>
<tr>
<td>LGRECLEN</td>
<td>Length of the retrieved record</td>
<td>Set by User program</td>
<td>Binary/Word</td>
</tr>
<tr>
<td>LGRECPTR</td>
<td>Pointer to the retrieved record data</td>
<td>Set by User program</td>
<td>Pointer/Word</td>
</tr>
<tr>
<td>LGMSGBUF</td>
<td>Communications message returned to PowerExchange from the User program.</td>
<td>Set by User program</td>
<td>Character/8 0 bytes</td>
</tr>
</tbody>
</table>
4. The access module receives the user ID and password from the PowerExchange Navigator screen or the ETL tool. The access module uses these credentials to check file security.

Assembler Example

***************************************************************************
* USREX002 - MVS Assembler Interface
* This is a sample exit that demonstrates some of the use of
* exit USREX002. It mainly produces diagnostic messages that
* show calling and returned values rather than reading and writing
* real user data.
* N.B. OS calling linkage is used with return code always in R15
* and that the calling program is written in C hence the use
* of EDCPR1G and EDCEP1L at the beginning and end of the
* exit. These macros should be in CER.SCEBMAC or your local
* equivalent.
* 1 Entry point as follows:
* int USREX002 (char *lgif);
* User exit to handle I/O related requests
* Return Codes in R15:
* 255 End of file
*  0 OK
* other non specific error - NULL delimited message in
* LMSGSUBP (Max 79 chars + NULL)
* The details of the request are passed to and from the exit via
* a structure whose address is passed as the single parameter to
* this exit. The DSECT for this is provided separately and may
* be copied in thus:
* COPY USREX02D
* This test exit assumes that a SYSDIAGS dataset is allocated and uses
* this dataset with an LRCL = 120.
* So in your JCL you will need something like
* //SYSDIAGS DD SYSOUT=A
* Two macros are defined inline to invoke the C Prolog and Epilog
* functions.
* \MACRO
&NAME FUNCSTR
&NAME AMODE ANY
&NAME EMODE ANY
&NAME CSECT
EDCPR1G BASEREG=12
MEND
* \MACRO
&NAME FUNCEND
EDCEP1L MEND
* \******** USREX002 Entry Point
* USREX002 FUNCSTR
L R3,0(R1) get structure address
USING LGIF,R3
GETMAIN R,LV=WORKLEN get some work storage
LR R4,R1 save the address
USING WORKAREA,R4
XR R15,R15 clear RC
ST R15,WORKRC set a good one for now
* *** End of prologue ***
* ** set up to use SYSDIAGS dataset for test messages
* MVC WORKOPEN[LOPENLEN],LOPEN copy OPEN prototype
***************************************************************************
*** Format the input structure ***

** PUT WORKDCB,-CL120'00005 THE FILE NAME SPECIFIED IS' **
** MVC WORKLINE,C' ' clear the print **
** MVC WORKLINE+1(119),WORKLINE buffer **
** BCTR R5,0 for the executed move **
** RX R5,LCOMP is it OK ? **
** BZ NOPFILEM no bother. **
** BNZ PRFILEM no **
** L R5,LFLEN get the length of the file name **
** LTR R5,R5 is there a file name ? **
** BZ NOPFILEM no **
** BNC PRFILEM no **
** MVCFILE MVC WORKLINE(4),0(R6) **
** *** Test the input opcode to see what function is requested ***

** DOOPEN DS OH **
** * **
** CLC LGCODE(4),=AL4 (LGOPEN) is this an OPEN ? **
** BE DOOPEN **
** CLC LGCODE(4),=AL4 (LGCLOSE) is this a CLOSE ? **
** BE DOCLOSE **
** CLC LGCODE(4),=AL4 (LGREAD) is this a READ ? **
** BE DOREAD **
** CLC LGCODE(4),=AL4 (LGWRITE) is this a WRITE ? **
** BE DOMWRITE **
** MVC WORKKC,-'F'4' whats going on ? **
** XC LGMSGBUF,LGMSGBUF **
** MVC LGMSGBUF(40),=CL40'Invalid OPCODE specified' **
** B EXIT **
** *** DOOPEN *** **
** * **
** DOOPEN DS OH **
** * **
** CLC LGMODE(4),=AL4 (LGMODE) is this an OPEN for read ? **
** BE OPENREAD **
** CLC LGMODE(4),=AL4 (LGMODE) is this an OPEN for write ? **
** BE OPENWRITE **
** CLC LGMODE(4),=AL4 (LGMODE) is this an OPEN for update ? **
** BE OPENUPDT **
** CLC LGMODE(4),=AL4 (LGMODE) is this an OPEN for append ? **
** BE OPENAPPN **
** MVC WORKKC,-'F'4' whats going on ? **
** XC LGMSGBUF,LGMSGBUF **
** MVC LGMSGBUF(40),=CL40'Invalid OPEN mode specified' **
** B EXIT **
** *** For testing we merely set the file open flag *** **
** * **
** OPENREAD DS OH **
** PUT WORKDCB,-CL120'00011 OPCODE=1 MODE=1 Opening for Read' **
** B OPENEND **
** OPENWRITE DS OH **
** PUT WORKDCB,-CL120'00012 OPCODE=1 MODE=2 Opening for Write' **
** B OPENEND **
** OPENUPDT DS OH **
** PUT WORKDCB,-CL120'00013 OPCODE=1 MODE=3 Opening for Update' **
** B OPENEND **
** OPENAPPN DS OH **
** PUT WORKDCB,-CL120'00014 OPCODE=1 MODE=4 Opening for Append' **
** B OPENEND **
** OPENEND DS OH **
** MVC LGOPENRC,-CL4'OPEN' set file open **
** B EXIT **

174 Chapter 11: Creating User Access Method Programs
*** DOCLOSE ***

* for testing CLOSE we merely reset the file open flag

DOCLOSE DS 0H

PUT WORKDCB,'-CL120'00020 OPCODE=2 Closing the file'

XC LGOPENRC,LGOPENRC set file close

B EXIT

*** DOREAD ***

* for read testing return one of our hard coded records
* index down the array on the basis of the read count
* specified in the input structure. 0 returns the first record

DOREAD DS 0H

PUT WORKDCB,'-CL120'00030 OPCODE=3 Doing a Read'

L R5,LRREADCT Read count from input

C R5,NUMRECS too many ?

BH NOMORE YES

MH R5,-H'80' index down our samples

LA R6,FIRST(R5) no

ST R6,LRRECPTR return the pointer

MVC LGRECLEN,LENGTH and length

B EXIT

NOMORE DS 0H

PUT WORKDCB,'-CL120'00031 End of file on READ'

XC LGRECLEN,LGRECLLEN BOF so clear the length

XC LGRECPTR,LGRECPTR clear the pointer

MVC WORKRC,-F'255' set the flag

B EXIT

*** DOWRITE ***

* for testing just output the first 120 bytes of the record

DOWRITE DS 0H

PUT WORKDCB,'-CL120'00040 OPCODE=4 Doing a Write of...'  

L R5,LGRECLLEN Get the length to be written

L R6,LRRECPTR Get address of data

LTR R5,R5 anything to write ?

BZ EXIT

LTR R6,R6 any buffer ?

BZ EXIT

MVI WORKLINE,C' ' clear the print line

MVC WORKLINE+1(119),WORKLINE

C R5,-F'120' too much to print ?

BNH WRTPRINT no

L R5,-F'120' yes set the maximum

WRTPRINT DS 0H

BCTR R5,0 set up the move

EX R5,MVCFILE do it and

PUT WORKDCB,WORKLINE print

* *** EXIT ***

EXIT DS 0H

*** test returned message facility ***

* Just for test purposes if LGUI is set non-zero
* we return a null delimited mag.

L R5,LGUI1 shall we return a message string ?

LTR R5,R5

BZ NOMSG No msg to return

XC LGMSGBUF,LGMSGBUF

MVC LGMSGBUF(40),-CL40'Just a test message'

MVC WORKRC,-F'4' tell main line there's a msg

NOMSG DS 0H

* ***

* *** format the returned structure in hex ***

* PUT WORKDCB,'-CL120'00003 LEAVING EXIT USREX002'

* *** format the returned structure in hex ***

* PUT WORKDCB,'-CL120'00004 OUTPUT VALUES IN HEX'

MVC WORKLINE(12GPLLEN),LGIF format the

TR WORKLINE(12GPLLEN),DUMPTBHI high hex digits

PUT WORKDCB,WORKLINE and print

MVC WORKLINE(12GPLLEN),LGIF format the low

TR WORKLINE(12GPLLEN),DUMPTBLO hex digits

PUT WORKDCB,WORKLINE and print

CLOSE WORKDCB,MP=-(E,WORKCLOS) close the diagnostic file
C Example

```
#include "dtlamlgi.h"
#include "string.h"
#include "stdio.h"

/**************************************************************************/
/* Contains an example of a user exit program                          */
/* USREX001.C                                                           */
END                                                            02640000
R15      EQU   15                                                      02630000
R14      EQU   14                                                      02620000
R13      EQU   13                                                      02610000
R12      EQU   12                                                      02600000
R11      EQU   11                                                      02590000
R9       EQU   9                                                       02570000
R8       EQU   8                                                       02560000
R7       EQU   7                                                       02550000
R6       EQU   6                                                       02540000
R5       EQU   5                                                       02530000
R3       EQU   3                                                       02510000
R2       EQU   2                                                       02500000
R1       EQU   1                                                       02490000
R0       EQU   0                                                       02480000
WORKLEN  EQU   *-WORKAREA                                             02460001
WORKDCB  DS    CL(LDCBLEN)                                            02450004
WORKOPEN DS    CL(LOPENLEN)                                           02430004
WORKLINE DS    CL133                                                 02410009
WORKRC   DS    F                                                      02400002
WORKAREA DSECT                                                     02390001
LTORG                                                        02380000
DC    16X'C6'                                                 02360009
DC    16X'C5'                                                 02350009
DC    16X'C4'                                                 02340009
DC    16X'C3'                                                 02330009
DC    16X'C2'                                                 02320009
DC    16X'C1'                                                 02310009
DC    16X'C0'                                                 02300009
DUMPTBH1 DC  16X'F0'                                               02290009
DC  16X'F1'                                                 02280009
DC  16X'F2'                                                 02270009
DC  16X'F3'                                                 02260009
DC  16X'F4'                                                 02250009
DC  16X'F5'                                                 02240009
DC  16X'F6'                                                 02230009
DC  16X'F7'                                                 02220009
DC  16X'F8'                                                 02210009
DC  16X'F9'                                                 02200009
DC  16X'C1'                                                 02190009
DC  16X'C2'                                                 02180009
DC  16X'C3'                                                 02170009
DC  16X'C4'                                                 02160009
DC  16X'C5'                                                 02150009
DC  16X'C6'                                                 02140009
DUMPTBLO DC  16X'F01F2F3F4F5F6F7F8F9C1C2C3C4C5C6'               02130009
LDCB DCB DDNAME=SYSDIAGS,DSORG=PS,MACRF=PM,LRECL=120               02120000
LCLOSE EQU   *-LCLOS                                                 02110008
LOPENLEN EQU   *-LOPEN                                                02100008
LDCBLN EQU   *-LDCB                                                 02090008
LATEG DSECT                                                       02080008
LTERM DS    F                                                       02070008
WORKLINE DS   CL133                                                02060008
WORKINF DS   CL133                                                02050008
WORKOPEN DS   CL(LOPENLEN)                                          02040008
WORKCLOS DS   CL(LCLOSLEN)                                         02030008
WORKDCB DS   CL(LDCBLN)                                            02020008
WORKLEN EQU   *-WORKAREA                                           02010008
* R0 EQU   0                                                       02000008
R1 EQU    1                                                       01990008
R2 EQU    2                                                       01980008
R3 EQU    3                                                       01970008
R4 EQU    4                                                       01960008
R5 EQU    5                                                       01950008
R6 EQU    6                                                       01940008
R7 EQU    7                                                       01930008
R8 EQU    8                                                       01920008
R9 EQU    9                                                       01910008
R10 EQU   10                                                      01900008
R11 EQU   11                                                      01890008
R12 EQU   12                                                      01880008
R13 EQU   13                                                      01870008
R14 EQU   14                                                      01860008
R15 EQU   15                                                      01850008
END                                                            01840008
```
#if defined WIN32
    #define DLLEXPORT __declspec(dllexport)
#else
    #define DLLEXPORT
#endif

#if defined DTL_MVS
    #pragma linkage(USREX001, fetchable)
#endif

/**************************************************************************/
/* Internal prototypes                                                   */
/**************************************************************************/
static int F01_Open(pLGIF plgif);
static int F02_Close(pLGIF plgif);
static int F03_Read(pLGIF plgif);
static int F04_Write(pLGIF plgif);
DLLEXPORT int Print_Parms(pLGIF plgif);

/**************************************************************************/
/* Dummy file                                                            */
/**************************************************************************/
#define MAXIND 6
char * dummy_file[MAXIND]=
{
    "Barcelona",
    "Manchester United",
    "Bayer",
    "Juventus",
    "Arsenal",
    "Inter"
};

/*========================================================================*/
/* USREX001                                                               */
/* This is the only external entry point to this module                   */
/*========================================================================*/
#if defined DTL_MVS
DLLEXPORT int USREX001(void *pinterface)
#else
DLLEXPORT int usrex001(void *pinterface)
#endif
{
    pLGIF plgif = (pLGIF) pinterface;
    switch (plgif->opcode)
    {
    case LGIF_OPEN:
        return F01_Open(plgif);
    case LGIF_CLOSE:
        return F02_Close(plgif);
    case LGIF_READ:
        return F03_Read(plgif);
    case LGIF_WRITE:
        return F04_Write(plgif);
    default:
        printf("Invalid Opcode\n");
    }
    return 0;
}
/*========================================================================*/
/* F01_Open                                                               */
/*========================================================================*/
static int F01_Open(pLGIF plgif)
{
    printf("** OPEN being called\n");
    Print_Parms(plgif);
    /* Demonstrate returning an error message if ui1 = 99 */
    if (plgif->ui1 == 99)
    {
        strcpy(plgif->umsgbuf, "Attempt to start USREX001 with invalid ui1");
        return 77; /* User specified return code */
    }
    plgif->file_opened = 1; /* set to !0 if opened OK */
    plgif->puwk1 = (char) 0; /* initialize the file pointer */
    return 0;
}
/*========================================================================*/
/* F02_Close                                                              */
/*========================================================================*/
static int F02_Close(pLGIF plgif)
```c
{
    printf("** CLOSE being called\n");
    Print_Parms(plgif);

    return 0;
}
/*========================================================================*/
/* F03_Read                                                            */
/*========================================================================*/
static int F03_Read(pLGIF plgif)
{
    int i = (int) (plgif->pwk1);
    printf("** READ being called\n");
    Print_Parms(plgif);
    if (i >= MAXIND)        /* if end of file, return 255 */
        return (255);
    else
    {
        plgif->prec=dummy_file[i];
        plgif->reclen = strlen(dummy_file[i]);
        i ++;
        plgif->pwk1 = (char *) i;
    }
    return 0;
}
/*========================================================================*/
/* F04_Write                                                            */
/* Just write the output record.                                         */
/*========================================================================*/
static int F04_Write(pLGIF plgif)
{
    printf("** WRITE being called\n");
    PrintParms(plgif);
    return 0;
}
/*========================================================================*/
/* Print                                                                */
/*========================================================================*/
DLLEXPORT int Print_Parms(pLGIF plgif)
{
    /* Print opcode */
    switch (plgif->opcode)
    {
    case LGIF_OPEN:
        printf("        opcode  = %s\n","OPEN");
        break;
    case LGIF_CLOSE:
        printf("        opcode  = %s\n","CLOSE");
        break;
    case LGIF_READ:
        printf("        opcode  = %s\n","READ");
        break;
    case LGIF_WRITE:
        printf("        opcode  = %s\n","WRITE");
        break;
    default:
        printf("        opcode  = %d\n",plgif->opcode);
        break;
    }

    /* Print openmode */
    switch (plgif->openmode)
    {
    case LGIF_WRITEMODE:
        printf("        openmode= %s\n","WRITEMODE");
        break;
    case LGIF_UPDATEMODE:
        printf("        openmode= %s\n","UPDATEMODE");
        break;
    case LGIF_APPENDMODE:
        printf("        openmode= %s\n","APPENDMODE");
        break;
    default:
        printf("        openmode= %d\n",plgif->openmode);
        break;
    }
    if (plgif->opcode == LGIF_OPEN)
    {
        printf("        ui1     = %d\n",plgif->ui1);
        printf("        ui2     = %d\n",plgif->ui2);
    }
    if (plgif->puser1 != NULL)
        printf("        user1   = %s\n",plgif->puser1);
    if (plgif->puser2 != NULL)
        printf("        user2   = %s\n",plgif->puser2);
```
if (plgif->pfm != NULL)
    printf(" pfn     = %s
",plgif->pfm);
printf(" puwk1   = %d
", (int) plgif->puwk1);
printf(" reclen  = %d
", plgif->reclen);
printf(" rec     = %s
", plgif->rec);
printf( "\n");
return 0;  }

COBOL Example

//UAMCOB JOB (),CLASS=A,MSGCLASS=X,MSGLEVEL=(1,1),REGION=4M,  
//NOTIFY=&SYSUID  
//  
//* SET HLQ=<libname>  
//SET VER=<version>  
//***************************************************************  
//** COMPIL PROGRAM  
//COBCOMP EXEC PROC=IGYWCL,PARM=('XREF(FULL)')  
//COBOL.SYSIN DD *  
IDENTIFICATION DIVISION.
PROGRAM-ID. UAMCOB.
*REMARKS. THIS IS A SAMPLE PROGRAM WHICH EXPECTS TO BE DRIVEN
* BY A DATAMAP WHICH INVOKES PROGRAM UAMCOB.
* THE MAP MUST HAVE IN INTEGER 1 THE LENGTH OF THE
* MAXIMUM RECORD AND IN INTEGER 2 THE NUMBER OF RECS
* TO BE READ.
ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.
77  DEBUG-SW                        PIC X VALUE '0'.
77  HEAP0                           PIC S9(8) COMP VALUE +0.
77  RECCT                           PIC S9(8) COMP VALUE +0.
 01  STOR-C.
    04  STOR-C-RC                  PIC X(4).
    04  STOR-C-8                   PIC X(8).
LINKAGE SECTION.
 01  LGIF.
  * For the OPEN request, the OPEN mode type requested may be one of
  * the following
  * User exit may handle four types I/O related requests:
    04  LGMODE                  PIC S9(8) COMP.
      88 LGMOMODE VALUE 1.
      88 LGMOMODE VALUE 2.
      88 LGMOMODE VALUE 3.
      88 LGMOMODE VALUE 4.
    04  LGCODE                  PIC S9(8) COMP.
      88 LGOPEN  VALUE 1.
      88 LGCLOSE VALUE 2.
      88 LGREAD VALUE 3.
      88 LGWRITE VALUE 4.
    04  LGUI1                   PIC S9(8) COMP.
    04  LGUI2                   PIC S9(8) COMP.
    04  LGREADCT                PIC S9(8) COMP.
    04  LGWRITECT               PIC S9(8) COMP.
    04  LGUIDLEN                PIC S9(8) COMP.
    04  LGUI2LEN                PIC S9(8) COMP.
    04  LGuinlen                PIC S9(8) COMP.
    04  LGUIDPTR                POINTER.
    04  LGUI2PTR                POINTER.
    04  LGSQLPTR                POINTER.
    04  LGWDPTR                 POINTER.
    04  LGPWDPTR                POINTER.
    04  LGWQPST                 PIC S9(8) COMP.
    04  LGWQPST                 PIC S9(8) COMP.
    04  LGWQPST                 PIC S9(8) COMP.
    04  LGWQPST                 PIC S9(8) COMP.
  01  RECORD-OUT.
    04  DBSC                   PIC X(9).
    04  RECCT1                 PIC Z(6)9.
    04  RECCT2                 PIC S9(4) COMP.
    04  RECCT3                 PIC S9(8) COMP.
PROCEDURE DIVISION USING LGIF RETURNING RC.

S0001-MAIN SECTION.
   IF LGOPEN
      PERFORM S1000-OPEN
      GO TO S0001-P99.
   IF LGCLOSE
      PERFORM S3000-CLOSE
      GO TO S0001-P99.
   IF LGREAD
      PERFORM S2000-READ
      GO TO S0001-P99.
   MOVE 4 TO RC.
   MOVE 'Invalid OPCODE specified' TO LGMSGBUF.
S0001-P99.
   GOBACK.

****************************************************************
S1000-OPEN SECTION.
   IF DEBUG-SW = 1 DISPLAY 'UAMCOB IN OPEN'.
   MOVE 1 TO LGOPENRC.
   MOVE 0 TO RC.
   MOVE SPACES TO LGMSGBUF.
   CALL 'CEEGTST' USING HEAP0,
      LGUI1,
      LGRECPTR,
      STOR-C.
   IF STOR-C-RC NOT = LOW-VALUES
      MOVE 4 TO RC
   ELSE
      SET ADDRESS OF RECORD-OUT TO LGRECPTR.
      EXIT.
****************************************************************
S2000-READ SECTION.
   IF DEBUG-SW = 1 DISPLAY 'UAMCOB IN READ'.
   IF LGREADCT >= LGUI2
      MOVE 0 TO LGRECLEN
      MOVE 255 TO RC
      DISPLAY 'END OF FILE SET'
      GO TO S2000-P99.
   MOVE 80 TO LGRECLEN.
   MOVE 0 TO RC.
   MOVE 'RECORD ' TO DESC.
   ADD 1 TO RECCT.
   MOVE RECCT       TO RECCT1
      RECCT2
      RECCT3
      RECCT4.
S2000-P99.
   EXIT.
****************************************************************
S3000-CLOSE SECTION.
   IF DEBUG-SW = 1 DISPLAY 'UAMCOB IN CLOSE'.
   IF LGRECPTR NOT = NULL
      CALL 'CEEFRST' USING LGRECPTR,
         STOR-C.
   MOVE 0 TO LGOPENRC.
   MOVE 0 TO RC.
   MOVE SPACES TO LGMSGBUF.
   EXIT.
****************************************************************
END PROGRAM UAMCOB.
PROC(LGIF_PTR) OPTIONS(FETCHABLE) RETURNS(FIXED BIN(31));

/* INPUT PARAMETER AND DEFINITION OF */
DCL LGIF_PTR
PTR;
DCL 1 LGIF
BASED(ADDR(LGIF_PTR)),
 4 LGMODE
FIXED BIN(31),
 4 LGCODE
FIXED BIN(32),
 4 LGU1
FIXED BIN(31),
 4 LGU2
FIXED BIN(31),
 4 LGREADCT
FIXED BIN(31),
 4 LGWRITECT
FIXED BIN(31),
 4 LGMLRECL
FIXED BIN(31),
 4 LGU1LEN
FIXED BIN(31),
 4 LGU2LEN
FIXED BIN(31),
 4 LGFNPTR
PTR,
 4 LGU1PTR
PTR,
 4 LGU2PTR
PTR,
 4 LGSQLPTR
PTR,
 4 LGFUNK1
FIXED BIN(31),
 4 LGFUNK2
FIXED BIN(31),
 4 LGOPENRC
FIXED BIN(31),
 4 LGRECL
FIXED BIN(31),
 4 LGRECFPTR
PTR,
 4 LGMSGBUF
CHAR(80);

/* GENERAL DECLARATIONS AND BUILTIN FUNCTIONS. */
DCL RC
FIXED BIN(31) INIT(0);

/* OUTPUT RECORD AREA (FIXED FOR EXAMPLE PURPOSE) */
DCL 1 REC_OVERLAY
BASED(ADDR(TEMP_STRING)),
 4 CHAR_ID
CHAR(7),
 4 PIC_NBR
PIC'(6)Z9',
 4 FB_2_NBR
FIXED BIN(15),
 4 FB_4_NBR
FIXED BIN(31),
 4 PACK_NBR
FIXED(7);
DCL (ADDR,
  MIN,
  VERIFY)
BUILTIN;

/* MAINLINE PROGRAM - DETERMINE ACTION AND PROCESS ACCORDINGLY */
SELECT(LGIF.LGCODE);
WHEN(1) DO;
  /* OPEN FILE */
  REC_OVERLAY = "'",
  REC_OVERLAY.CHAR_ID = 'RECORD',
  LGIF.LGOPENRC = 1,
  LGIF.LGRECL = LGIF.LGUI1,
  LGIF.LGRECFPTR = ADDR(TEMP_STRING);
END;
WHEN(2) DO;
  /* CLOSE FILE */
END;
WHEN(3) DO;
  /* READ RECORD */
  REC_OVERLAY.PIC_NBR = REC_OVERLAY.PIC_NBR +1;
  REC_OVERLAY.PACK_NBR = REC_OVERLAY.PACK_NBR +1;
  REC_OVERLAY.FB_2_NBR = REC_OVERLAY.FB_2_NBR +1;
  REC_OVERLAY.FB_4_NBR = REC_OVERLAY.FB_4_NBR +1;
  LGIF.LGRECL = LGIF.LGUI1;
  IF LGIF.LGREADCT >= LGIF.LGUI2
  THEN DO;
    RC = 255;
    LGIF.LGRECL = 0;
  END;
END;
OTHER ;
END;
RETURN(RC);
END UAMPL1;
INDEX

A
access methods, user
   Assembler example 173
   C example 176
   COBOL example 179
   example programs 172
   overview 169
accessing, unload data from multiple tables 31
acquiring, IMS source information 40
activating
capture registration 127
Adabas keys, importing 23
   add
capture registration to an existing group 122
Add Extract Definition Wizard 129
add extraction definitions
capture registrations 132
   Condense 133
   name 133
   status 133
   version 133
add extraction group
   add extraction definition 131
database instance 131
   location 130
   name 130
   password 130
   type 130
   user ID 130
adding
capture registration 118
capture registration to an existing group 122
   expressions 37
   IMS data maps 33
   IMS data maps using PowerExchange Navigator 42
   owner and set name 36
   registration group 116
   registration groups 115
table to an IMS data map 37
Amend Control Indicator/Before Image Extensions 139
AS400 Instance
   CAPX Advanced parameters 110
   AS400 Library/Journal
   CAPXRT Advanced Parameters 110
   EMR Advanced Parameters 112
auto-generated fields
   DTL__CAPXACTION 136
   DTL__CAPXCASDELIND 136
   DTL__CAPXRESTART1 135
   DTL__CAPXRESTART2 135
   DTL__CAPXTIMESTAMP 136
   DTL__CAPXUOW 135
   DTL__CAPXUSER 135

B
Before and After Images 109, 110, 111
Before Image 139
BI 139
BIN 8 151

C
Calc Element Names 36
CALLPROG
   Assembler Linkage 164
   CL linkage 165
   COBOL linkage 163
   external C routines 163
   external COBOL routines 163
   external OS routines 164
   external OS400 routines 165
   external PL/1 routines 166
   external programs overview 161
   overview 161
   PL/1 linkage 166
capture registration
details 123
   overview 115
   CAPX 129
   CAPXRT 129
   Change Indicator 139
changing
capture registration 124
capture registration to add or remove columns 125
   IMS data maps 33
   IMS record properties 35
   IMS options 48
   properties of the registration definition 126
   CI 139
   Click 48
   COBOL Copybook 66
   COBOL copybook 71
   COBOL copybooks 2
   Codepage Support 104
   Condense 129
   control interval access
      using 56
creating
a datacom data map 24
a sample IMS data map 51
Adabas data maps 20
an Adabas data map 20
data maps using DB2 unload files 29
data maps using the DB2 catalog 27
Datacom data maps 24
DB2 data maps 26
IDMS data maps 32
IMS data map example 50
IMS data maps 37
VSAM data maps 52
Current Capture Registration
Name 133
current capture registration
Add 133
Condense 133
Remove 133
Status 133
Version 133

D
Data Checking 93
data map
code pages 63
concepts 11
structure 38
tasks 37
data map options to retrieve the RRN or RBA
using 57
database row test 34
Datacom data maps, creating 24
DB2 data maps, creating 26
DB2 UDB Database
CAPXRT Advanced parameters 111
EMR Advanced parameters 112
DB2-BIN_CODEPAGE
Configuration Parameter 104
DB2CODEPAGE
Configuration Parameter 104
DDS Copybook 81
defining
condense option 121
Delete unwanted Fields 69
deleting
registration entry 122
registration group 122
deleting, Calc Element Name 37
displaying, element information 25
DTL_BI 136
DTL_CI 136
DTLAPP 134
DTLDNSN 134
DTLXTYPE 134

E
EBCDIC data 71
Expression Editor 145
Expressions 143
  Call 147
Check 148
Concat 148
Copydata 149
Fragment 149
GenVRowKey 150
GetDatabaseKey 151
GetDbKey 152
GetDbKeyOwner 152
GetFullDbKey 152
GetFullDbKeyOwner 153
GetSeqWithinLevel 155
GetSeqWithinParent 156
LengthOf 157
LTRIM 157
RT RIM 157
SetBitA 159
SetBitN 159
Split 158
Strip 159
ToLower 160
ToUpper 160
Extended Relative Byte Address 151
Extensions tab 139
Extra Properties Tab
  Cobol or PL1 Field Attributes 68
Extract Type
  CAPX Advanced parameters 109
  CAPXRT Advanced parameters 110
  EMR Advanced parameters 111
extraction criteria 134
extraction definitions 129
extraction group 130
extraction process 130
extraction type
  RS - Restart 134
  SL - Since Last 134

F
Field Level Change Capture 139

G
GenVRowKey 150
getdatabasekey expression to retrieve the RRN or RBA
using 58
Index 185

GetDatabaseFlowType 151
GetDbKeyOfOwner 152
GetFullDbKey 150, 152
GetIMSRBAByLevel 153
GetPageGroup 154
GetPgGrpAndRdx 154
GetPgGrpAndRdxOfOwner 155
GetPgGrpOfOwner 154
GetRadix 154
GetRadixOfOwner 155

H

HP3000 Volume Set
EMR Advanced parameters 112

I

IDMS data maps, creating 32
IDMSMJCL 5
Image Type
CAPX Advanced parameters 109
CAPXRT Advanced parameters 110
Import 52
Import Copybook Wizard 67
importing
Adabas database FDT 23
Adabas keys 23
an IMS DBD source 44
IMS DBD source for IMS data maps 43
improving, bulk read performance for VSAM data sets 55
IMS data map examples 50
IMS data maps, creating 37
IMS DBD source example 50
IMS DBD source for IMS data maps, importing 43
Internal Sequence Number 151
ISN 151

K

key 151
Key object 151

L

LISTENER 4, 5, 6, 7, 8
LOADJOBFILE 5
lookup transformations with IMS databases using 47

M

Metadata Browsing 101
modifying capture registrations 124
modifying, IMS data maps 46
moving data with PowerExchange 1
Multibyte Data
Row Test 104

N

No Progress Update
CAPX Advanced parameters 110
CAPXRT Advanced parameters 110
EMR Advanced parameters 112
NODE 4
nonrelational view of data 39
NUM 32 151

O

ODBC interface to PowerExchange 134
offload processing for VSAM data sets using 57
Oracle Collection ID
EMR Advanced parameters 112
Oracle Connection String
CAPXRT Advanced parameters 111
EMR Advanced parameters 112
Oracle Instance
CAPXRT Advanced parameters 111
EMR Advanced parameters 112
Oracle Schema
CAPXRT Advanced parameters 111

P

performing, Row Test 49
Personal Metadata 4, 101
PL/1 Copybook 76
PowerExchange code pages 63
PowerExchange configuration on MVS updating 41
PowerExchange configuration on Windows updating 41
Pre-requisite 2

R

RBA 151
record definition 90
Redefines 71
Relative Byte Address 151
Relative Record Number 151
Resource Configuration 8
resource explorer 123
resource inspector 124
retrieving
RRN or RBA 57
RRN or RBA using expressions 58
returning, an RBA 46
Row Test 69, 140
row testing
DB2 data map 28
DB2 unload file data map 31

Index 185
second copybook 90
sending, data map to the PowerExchange Listener 47
Sorting a Data Map 96
SQL Request Format 130
storing, IMS data maps 46
structure and content, of data map 12

testing
   Adabas data maps 23
   Datacom data maps 25
   IMS data maps 49
   VSAM data maps 58
Testing column changes 140
Testing the Windows LISTENER 3
Timeout
   CAPX Advanced parameters 110
   CAPXRT Advanced parameters 110
   EMR Advanced parameters 112

Un-condensed 129
updating
   PowerExchange configuration on MVS 41
   PowerExchange configuration on Windows 41
Use "WHERE" clause 70
user access method example
   Assembler 173
   C 176
   COBOL 179
User-defined fields 143
using
   control interval access 56
   data map options to retrieve the RRN or RBA 57
   getdatabasekey expression to retrieve the RRN or RBA 58
   lookup transformations with IMS databases 47
   offload processing for VSAM data sets 57
   VSAM keys 56

views
version indicator 138
view, of IMS hierarchy 40
viewing
   capture registration 123
   IMS hierarchy 40
   IMS options 48
   or changing IMS options 48
   registration group 123
viewing capture registration details 123
Views 9
VSAM data maps, creating 52
VSAM keys
   using 56
NOTICES

This Informatica product (the "Software") includes certain drivers (the "DataDirect Drivers") from DataDirect Technologies, an operating company of Progress Software Corporation ("DataDirect") which are subject to the following terms and conditions:

1. THE DATADIRECT DRIVERS ARE PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NON-INFRINGEMENT.

2. IN NO EVENT WILL DATADIRECT OR ITS THIRD PARTY SUPPLIERS BE LIABLE TO THE END-USER CUSTOMER FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, CONSEQUENTIAL OR OTHER DAMAGES ARISING OUT OF THE USE OF THE ODBC DRIVERS, WHETHER OR NOT INFORMED OF THE POSSIBILITIES OF DAMAGES IN ADVANCE. THESE LIMITATIONS APPLY TO ALL CAUSES OF ACTION, INCLUDING, WITHOUT LIMITATION, BREACH OF CONTRACT, BREACH OF WARRANTY, NEGLIGENCE, STRICT LIABILITY, MISREPRESENTATION AND OTHER TORTS.