

# Pervasive PSQL v9

---

## *Active Connector for AcuCOBOL*

### **Developer's Guide**

Pervasive Software Inc.  
12365 Riata Trace Parkway  
Building B  
Austin, TX 78727 USA

Telephone: 512 231 6000 or 800 287 4383

Fax: 512 231 6010

Email: [info@pervasive.com](mailto:info@pervasive.com)

Web: <http://www.pervasive.com>



## *disclaimer*

PERVASIVE SOFTWARE INC. LICENSES THE SOFTWARE AND DOCUMENTATION PRODUCT TO YOU OR YOUR COMPANY SOLELY ON AN “AS IS” BASIS AND SOLELY IN ACCORDANCE WITH THE TERMS AND CONDITIONS OF THE ACCOMPANYING LICENSE AGREEMENT. PERVASIVE SOFTWARE INC. MAKES NO OTHER WARRANTIES WHATSOEVER, EITHER EXPRESS OR IMPLIED, REGARDING THE SOFTWARE OR THE CONTENT OF THE DOCUMENTATION; PERVASIVE SOFTWARE INC. HEREBY EXPRESSLY STATES AND YOU OR YOUR COMPANY ACKNOWLEDGES THAT PERVASIVE SOFTWARE INC. DOES NOT MAKE ANY WARRANTIES, INCLUDING, FOR EXAMPLE, WITH RESPECT TO MERCHANTABILITY, TITLE, OR FITNESS FOR ANY PARTICULAR PURPOSE OR ARISING FROM COURSE OF DEALING OR USAGE OF TRADE, AMONG OTHERS.

## *trademarks*

Btrieve, Client/Server in a Box, Pervasive, Pervasive Software, and the Pervasive Software logo are registered trademarks of Pervasive Software Inc.

Built on Pervasive Software, DataExchange, MicroKernel Database Engine, MicroKernel Database Architecture, Pervasive.SQL, Pervasive PSQL, Solution Network, Ultralight, and ZDBA are trademarks of Pervasive Software Inc.

Microsoft, MS-DOS, Windows, Windows 95, Windows 98, Windows NT, Windows Millennium, Windows 2000, Windows XP, Win32, Win32s, and Visual Basic are registered trademarks of Microsoft Corporation.

NetWare and Novell are registered trademarks of Novell, Inc.

NetWare Loadable Module, NLM, Novell DOS, Transaction Tracking System, and TTS are trademarks of Novell, Inc.

All other company and product names are the trademarks or registered trademarks of their respective companies.

© Copyright 2007 Pervasive Software Inc. All rights reserved. Reproduction, photocopying, or transmittal of this publication, or portions of this publication, is prohibited without the express prior written consent of the publisher.

This product includes software developed by Powerdog Industries. © Copyright 1994 Powerdog Industries. All rights reserved. This product includes software developed by KeyWorks Software. © Copyright 2002 KeyWorks Software. All rights reserved. This product includes software developed by DUNDAS SOFTWARE. © Copyright 1997-2000 DUNDAS SOFTWARE LTD., all rights reserved. This product includes software developed by the Apache Software Foundation (<http://www.apache.org/>).

The ODBC Driver Manager for NetWare (ODBC.NLM) included in this product is based on the GNU iODBC software © Copyright 1995 by Ke Jin <[kejin@empres.com](mailto:kejin@empres.com)> and was modified by Simba Technologies Inc. in June 1999.

This library is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

A copy of the GNU Lesser General Public License is included in your installation of Pervasive PSQL at \program files\common files\Pervasive Software Shared\doc\lesser.htm. If you cannot find this license, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA. You may contact Pervasive Software Inc. using the contact information on the back cover of this manual.

**Active Connector Developer's Guide**

**January 2007**

**100-004277-001**

# Contents

<b>About This Manual . . . . .</b>	<b>vii</b>
Who Should Read This Manual . . . . .	viii
Manual Organization . . . . .	ix
Conventions . . . . .	x
<b>1 Introducing Pervasive PSQL Active Connector . . . . .</b>	<b>1-1</b>
<i>Understanding Pervasive PSQL Active Connector and its Components</i>	
Overview . . . . .	1-2
Components . . . . .	1-2
Constraints . . . . .	1-2
Tasks . . . . .	1-3
<b>2 Application Development . . . . .</b>	<b>2-1</b>
<i>Tasks for Application Development with Pervasive PSQL Active Connector</i>	
Development Tasks . . . . .	2-2
Step 1: Installing a Pervasive PSQL Database Engine . . . . .	2-3
Step 2: Installing the SDK and Runtime Components . . . . .	2-4
Step 3: Setting AcuCOBOL Compiler Directives . . . . .	2-5
Step 4: Creating an XML Schema of Your Data . . . . .	2-7
Examples . . . . .	2-7
Plan Generator Command Format . . . . .	2-7
Step 5: Editing the XML Schema. . . . .	2-9
Examples . . . . .	2-12
Step 6: Creating a Pervasive PSQL Database . . . . .	2-14
Example . . . . .	2-14
Serverdsn Command Format . . . . .	2-14
Step 7: Installing and Configuring the Runtime Components . . . . .	2-17
Installation . . . . .	2-17
Configuration . . . . .	2-18
Step 8: Creating the Pervasive PSQL Metadata . . . . .	2-23
Example . . . . .	2-23
Plan Executor Command Format. . . . .	2-23
Errors Resulting from Fields That Contain Occurs or Redefines. . . . .	2-25
Error Resulting from Too Many Fields. . . . .	2-27
Step 9: Converting COBOL Metadata Files to System Tables . . . . .	2-28

<b>3</b>	<b>Deployment . . . . .</b>	<b>3-1</b>
	<i>Tasks for Deploying Pervasive PSQL Active Connector</i>	
	Deployment Tasks . . . . .	3-2
	Step 1: Installing a Pervasive PSQL Database Engine . . . . .	3-3
	Step 2: Creating a Pervasive PSQL Database . . . . .	3-4
	Step 3: Installing and Configuring the Runtime Components . . . . .	3-5
	Step 4: Creating the Pervasive PSQL Metadata . . . . .	3-6
	Step 5: Converting COBOL Metadata to System Tables . . . . .	3-7
	Step 6: Migrating Existing Data Into a Pervasive PSQL Database . . . . .	3-8
	Creating a Migration Configuration File . . . . .	3-8
	Copying the List File of Data Files to Migrate. . . . .	3-8
	Migrating the Data . . . . .	3-8
<b>A</b>	<b>Log Messages . . . . .</b>	<b>A-1</b>
	<i>Messages Logged by SDK Utilities</i>	
	Plan Generator Log Messages . . . . .	A-2
	Plan Executor Log Messages. . . . .	A-4
<b>B</b>	<b>Special Data Type Mapping . . . . .</b>	<b>B-1</b>
	<i>Situations That May Require Data Type Mapping</i>	
	Overview . . . . .	B-2
	Creating a Custom Data Conversion Routine . . . . .	B-3
	Exporting Functions from Conversion.dll . . . . .	B-3
	Adding Your Customer Routines to Conv.cpp . . . . .	B-3
	Modify Header Files . . . . .	B-4
	Conv.h - ACConvertor . . . . .	B-4
	DbToNative.h - DBToNativeFunction . . . . .	B-4
	NativeToDb.h - NativeToDBFunction . . . . .	B-4
	Compiling the Conversion Library . . . . .	B-5
<b>C</b>	<b>Clients and Data Source Names . . . . .</b>	<b>C-1</b>
	<i>Conceptual Information About Data Source Names (DSNs)</i>	
	Data Source Names . . . . .	C-2
	Pervasive PSQL and DSNs . . . . .	C-2

# Tables

2-1	Options for SDK Plan Generator Utility . . . . .	2-7
2-2	Edits that May Be Required to an XML Schema. . . . .	2-9
2-3	Options for Serverdsn Utility . . . . .	2-15
2-4	Configuration Options . . . . .	2-19
2-5	Options for SDK Plan Executor Utility. . . . .	2-24
A-1	Plan Generator Log Messages . . . . .	A-2
A-2	Plan Executor Log Messages. . . . .	A-4



# *About This Manual*

---

This manual introduces you to the Pervasive PSQL Active Connector for AcuCOBOL and explains the tasks required to use that product.

## **Who Should Read This Manual**

This manual provides information for COBOL application developers who want to interface their application to a Pervasive PSQL database engine.

Pervasive Software Inc. would appreciate your comments and suggestions about this manual. As a user of our documentation, you are in a unique position to provide ideas that can have a direct impact on future releases of this and other manuals. If you have comments or suggestions for the product documentation, post your request at <http://www.pervasive.com/devtalk>.

## Manual Organization

This manual is divided into the following chapters:

- Chapter 1—“Introducing Pervasive PSQL Active Connector”  
Provides an overview of the active connector.
- Chapter 2—“Application Development”  
Explains the steps required to install and configure the Active Connector SDK, and to test the interaction between your COBOL application and a Pervasive PSQL database.
- Chapter 3—“Deployment”  
Explains the steps required to install and configure the active connector for deployment of an application.
- Appendix A—“Log Messages”  
Explains the messages that may be written to logs by the plan generator utility and the plan executor utility.
- Appendix B—“Special Data Type Mapping”  
Explains considerations if your application uses special data types.
- Appendix C—“Clients and Data Source Names”  
Explains Data Source Names (DSNs) and connecting to a Pervasive PSQL database.

The manual also includes an index.

## Conventions

Unless otherwise noted, command syntax, code, and examples use the following conventions:

CASE	Commands and reserved words typically appear in uppercase letters. You can enter these items using uppercase, lowercase, or both. For example, you can type MYPROG, myprog, or MYprog.
<b>Bold</b>	Words appearing in bold include the following: menu names, dialog box names, commands, options, buttons, statements, etc.
Monospaced font	Monospaced font is reserved for words you enter, such as command syntax.
[ ]	Square brackets enclose optional information, as in [ <i>log_name</i> ]. If information is not enclosed in square brackets, it is required.
	A vertical bar indicates a choice of information to enter, as in [ <i>file name</i>   @ <i>file name</i> ].
< >	Angle brackets enclose multiple choices for a required item, as in /D=<5   6   7>.
<i>variable</i>	Words appearing in italics are variables that you must replace with appropriate values, as in <i>file name</i> .
...	An ellipsis following information indicates you can repeat the information more than one time, as in [ <i>parameter</i> ...].
::=	The symbol ::= means one item is defined in terms of another. For example, a::=b means the item <i>a</i> is defined in terms of <i>b</i> .

# *Introducing Pervasive PSQL Active Connector*

# *1*

---

*Understanding Pervasive PSQL Active Connector and its Components*

This chapter contains the following sections:

- “Overview” on page 1-2
- “Tasks” on page 1-3

## Overview

Pervasive PSQL Active Connector for AcuCOBOL is a runtime interface to the Pervasive PSQL database engine. Active Connector allows an AcuCOBOL application to use the AcuCOBOL runtime environment as a front-end with a Pervasive PSQL database as a back-end. An AcuCOBOL application reads from and writes to Pervasive PSQL database files as if it were working with ISAM files. Active Connector functions transparently to the users of the COBOL application.

Pervasive PSQL databases support a variety of interfaces, including a relational interface through ODBC. A third-party application, such as a reporting program, can query the COBOL application's data through ODBC using relational access methods.

## Components

Active Connector comprises the following components:

- Software Development Kit (SDK) files and runtime files for the application developer's environment
- Runtime files to deploy to the application user's environment

## Constraints

The following constraints apply to this version of Pervasive PSQL Active Connector for AcuCOBOL.

- Indexes

Pervasive PSQL allows up to 119 segments per table. You can have up to 119 keys, but this number drops as more segments are added to each of the keys because of the segment limitation.

Indexes may not be within Redefine statements. Indexes may be within Occurs statements only if the Occurs statements are not normalized into separate Pervasive PSQL tables. (See “Step 5: Editing the XML Schema” on page 2-9 and “Step 8: Creating the Pervasive PSQL Metadata” on page 2-23 for more information.)

- XFD format

XFD file format must be 4 or 5.

## Tasks

The tasks required by a developer to install, configure, and use Active Connector fall into two categories: development and deployment. This guide discusses each category in detail in the following chapters.



# *Application Development*

*chapter*

**2**

---

*Tasks for Application Development with Pervasive PSQL Active Connector*

The topics in this chapter include:

- “Development Tasks” on page 2-2
- “Step 1: Installing a Pervasive PSQL Database Engine” on page 2-3
- “Step 2: Installing the SDK and Runtime Components” on page 2-4
- “Step 3: Setting AcuCOBOL Compiler Directives” on page 2-5
- “Step 4: Creating an XML Schema of Your Data” on page 2-7
- “Step 5: Editing the XML Schema” on page 2-9
- “Step 6: Creating a Pervasive PSQL Database” on page 2-14
- “Step 7: Installing and Configuring the Runtime Components” on page 2-17
- “Step 8: Creating the Pervasive PSQL Metadata” on page 2-23
- “Step 9: Converting COBOL Metadata Files to System Tables” on page 2-28

## Development Tasks

Development tasks take place within the application development environment, which typically requires only a single computer. As a developer, you need to install and configure the Active Connector SDK. With it, you then create all files required to test the interaction between your COBOL application and a Pervasive PSQL database.

The following tasks are required for the development environment:

- Install a Pervasive PSQL database engine and any required patches
- Install the SDK components for Pervasive PSQL Active Connector for AcuCOBOL
- Set AcuCOBOL compiler directives
- Create an XML schema of your data
- Edit the XML schema, if required
- Create a Pervasive PSQL database
- Install and configure the runtime components for the Active Connector
- Create the Pervasive PSQL metadata
- Convert COBOL metadata files to system tables

## Step 1: Installing a Pervasive PSQL Database Engine

A Pervasive PSQL database engine is required to use the SDK and runtime components of Active Connector. You must install the Pervasive PSQL Server product. The Pervasive PSQL database product version must be 9.0 or later.



**Note** Any hotfixes required by the Pervasive PSQL database product must also be applied.

You may obtain the hotfix executable and its companion readme file from the Web at [//www.pervasive.com/support/](http://www.pervasive.com/support/). For the Windows version of Pervasive PSQL Sever, install both the MKDE and the SRDE hotfixes.

---

Refer to the Pervasive PSQL *Getting Started With Pervasive PSQL* manual for detailed instructions on installing the product, and to the product README file for late-breaking information.

## Step 2: Installing the SDK and Runtime Components

The Active Connector SDK consists of a set of files that you copy to your development machine.

### ► To install the Active Connector SDK for Windows

- 1 Insert the CD for Pervasive PSQL Active Connector for AcuCOBOL into a CD drive.
- 2 Run the install script `install.bat`.

The install script is provided on the CD. The `install.bat` script performs the following:

- Copies the following files from the SDK folder to the folder to the `c:\pvs\bin` directory:
  - `cblrb.dll`
  - `mkc3alch.dll`
  - `conversion.dll`
  - `cddftool.exe`
  - `planexec.exe`
  - `plangen.exe`
  - `visionpsql_v5.obj`
  - `visionpsql_v6.obj`
  - `serverdsn.exe`
  - `clientdsn.exe` (see “Clients and Data Source Names” on page C-1)
- Copies `wrun32.dll` from the appropriate AcuGT folder to the AcuCOBOL bin directory.
- Registers `mkc3alch.dll` and `cblrb.dll` using `c:\pvs\bin\psregsvr.exe`.

---

## Step 3: Setting AcuCOBOL Compiler Directives

Active Connector requires an XFD file for each indexed and relative file to be included in the Pervasive PSQL database. (In the next step, Step 4, you use an Active Connector utility to process the XFD files.)



**Note** Only indexed and relative files may be stored in a Pervasive PSQL database. Sequential files remain the same and are routed by Active Connector to the Vision file system.

---

If your application lacks XFD files, you must set compiler directives to create them, then compile your application.

➤ **To set AcuCOBOL compiler directives required for a Pervasive PSQL database**

- 1 Open the COBOL source code file.
- 2 Add the following directive above each indexed or relative file definition (FD) that will be included in a Pervasive PSQL database:

```
$XFD FILE=filename
```

where *filename* is a file name of your choice.



**Tip** Match the XFD name to the table name. Also, data migration to Pervasive PSQL is easier if you limit the file name to 20 characters or less.

---

- 3 Optionally, if you want an SQL table for the redefine or variant record, add the following directive above the redefine or variant record:

```
$XFD WHEN <some expression>
```

See the *AcuCOBOL User's Guide* for details and examples about the \$XFD WHEN directive.

- 4 Compile your application by using the following compiler options:
  - For AcuGT 5.21, use compiler option -Fxa

- For AcuGT 6.00 or AcuGT 6.10, use compiler option -Fa



---

**Note** The -Fx option allows AcuCOBOL to build older file formats (where x is the format version). Plangen can process only format 4 and format 5 XFD files. Do **not** specify an older format such as 3 (-F3). By default, AcuGT 5.21 builds format 4 XFDs; AcuGT 6.00 and 6.10 build format 5 XFDs.

---

## Step 4: Creating an XML Schema of Your Data

Once you have XFD files for your application, you need to convert them into extensible markup language (XML). The Active Connector SDK includes a command-line utility called the Plan Generator that parses XFD files into an XML plan. The utility executable name is `plangen.exe`, and is referred to as `plangen`. The XML plan contains all of the necessary information to create the Pervasive PSQL metadata to support your application.

### Examples

To generate an XML plan named `myxmlplan` for an XFD file named `myxfd`, and write processing messages to the default log file:

```
plangen myxfd myxmlplan
```

To generate an XML plan named `myxmlplan` for all XFD files located in directory `C:\myappsdir` and write processing messages to log file `C:\output\mylog.log`:

```
plangen -l c:\output\mylog.log -i c:\myappsdir
myxmlplan
```

### Plan Generator Command Format

For an individual XFD file:

```
plangen [-b] [-e] [-v] [-h] [-n] [-l log-file]
xfd-file map-file
```

For multiple XFD files:

```
plangen [-b] [-e] [-v] [-h] [-n] [-l log-file] [-x
xfd-ext] -i xfd-dir map-file
```

Table 2-1 Options for SDK Plan Generator Utility

Option	Meaning
-b	Do not display banner.
-v	Run in verbose mode.
-e	Continue processing even after errors occur.
-h or -?	Display command usage. Ignore all other options.
-n	Populate XFD definitions with <code>normalize="no"</code> instead of the default "yes" value. Use of the <code>-n</code> option disables normalization of Occurs into separate tables. See "Step 5: Editing the XML Schema" on page 2-9 and "Step 8: Creating the Pervasive PSQL Metadata" on page 2-23 for more information.

Table 2-1 Options for SDK Plan Generator Utility

Option	Meaning
<i>-l log-file</i>	Log file to use for messages produced during processing of XFD file(s). The default log file is <b>plangen.log</b> and is located in the same directory from where you execute <b>plangen</b> . Each time <b>plangen</b> runs, new log information appends to the existing log. See also "Log Messages" on page A-1.
<i>-x xdf-ext</i>	File name extension of XFD files. The default extension is <b>xdf</b> .
<i>-i xfd-dir</i>	Process all XFD files in the specified directory. If <b>xdf-file</b> is specified, <b>xdf-dir</b> is ignored.
<i>xdf-file</i>	The file name of the input XFD file to be processed.
<i>map-file</i>	The file name for the output XML plan file. If you re-use the same XML file name for subsequent executions of the Plan Generator, new content overwrites the existing content.




---

**Note** If your application references ISAM files in multiple directories, you may need to create a separate XML schema for each directory since data in each directory is converted to separate Pervasive PSQL databases. For example, if you have a Support database and a Sales database, they probably have different types of files and therefore need different XML schemas. However, if the format of the files in each directory is the same, you can use the same XML schema for all of the conversions. For example, if every sales representative has his or her own database but each database contains the same types of files, one XML schema will describe all of the file types.

---

## Step 5: Editing the XML Schema

You may need to edit the XML schema generated by the Plan Generator depending on the names of the XFD files, data files, and the columns. A text editor may be used to edit the plan.

Table 2-2 Edits that May Be Required to an XML Schema

Condition	Type of Edit Required
XFD file name contains more than 20 characters	<p>Plangen adds an alias for the file name in the form <code>alias = "PSQL_TBL_n,"</code> where <i>n</i> starts with "1" and increments by one for each table alias added during the execution of plangen.</p> <p>You may prefer to change the aliases from "PSQL_TBL_n" to something more meaningful, especially if used in SQL queries.<sup>1</sup></p>
Your data files have a file extension and your application references them with that extension	<p>Add the file name extension to the XFD file name.</p> <p><b>Note:</b> Alternatively, you may add the file name extensions when you <b>execute</b> the XML plan by providing a option to the Plan Executor utility. See "Step 8: Creating the Pervasive PSQL Metadata" on page 2-23.</p> <p>AcuCOBOL removes file extensions when creating the XFD files. However, the COBOL application includes file extensions when it opens files. Therefore, the file extensions must be included in the XML file.</p> <p><b>Tip:</b> Suppose that your data files have an extension of .dat. If your XML plan contains no file name aliases, search for ' fileType=' and replace with '.DAT' fileType='. For example, the following XML structure would change from <code>&lt;XFD id="INDEX-FILE" file="PH35INX" fileType="12" &gt;</code> to <code>&lt;XFD id="INDEX-FILE" file="PH35INX.DAT" fileType="12" &gt;</code>.</p>
Data file names use different file name extensions	Manually add the file name extensions to the data file names. Also see previous condition.

Table 2-2 Edits that May Be Required to an XML Schema

Condition	Type of Edit Required
Group name contains more than 20 characters	<p>Plangen adds an alias for the group name in the form <code>alias = "PSQL_GROUP_mn,"</code> where <i>m</i> is the current table alias number and <i>n</i> starts with "1" for each table processed, then increments by 1 for each column within the table.</p> <p>You may prefer to change the aliases from "PSQL_GROUP_mn" to something more meaningful, especially if used in SQL queries.<sup>1</sup></p>
Column name contains more than 20 characters	<p>Plangen adds an alias for the column name in the form <code>alias = "PSQL_COL_n,"</code> where <i>n</i> starts with "1" for each table processed, then increments by 1 for each column within the table.</p> <p>You may prefer to change the aliases from "PSQL_COL_n" to something more meaningful, especially if used in SQL queries.<sup>1</sup></p>
Column name or table name contains one or more hyphens (-)	<p>No edits are required if the column name is 20 characters or less. Be aware, however, that plangen adds an alias for the column name or table name and replaces each hyphen with an underscore.</p> <p>Replacing the hyphens with underscores prevents your having to double quote column names when you access them through SQL.</p> <p>For example, a column named I-ALT-KEY-1 is assigned the alias I_ALT_KEY_1. If the column name exceeds 20 characters, the previous condition also applies.</p>

Table 2-2 Edits that May Be Required to an XML Schema

Condition	Type of Edit Required
<p>Occurs clauses should not be normalized into individual tables</p>	<p>By default, Plan Executor normalizes fields within an Occurs clause into a separate table.</p> <p>To disable normalization for <i>all</i> tables, see the Plan Executor information in “Step 8: Creating the Pervasive PSQL Metadata” on page 2-23. To disable normalization for a single file (table), change the <code>normalize="yes"</code> option in the XFD statement for that file to <code>normalize="no"</code>. The following example shows a modified option (boldfacing used for emphasis):</p> <pre data-bbox="727 557 1200 638">&lt;XFD id="MYTABLE" file=" MYTABLE " fileType="12" maxRecSz="54" minRecSz="54" <b>normalize="no"</b> &gt;</pre> <p>Reasons to normalize fields:</p> <ul style="list-style-type: none"> <li>◆ allows the easy access to like data via SQL</li> <li>◆ allows for a large number of Occurs instances (Pervasive PSQL has a limit of 1536 columns per table.)</li> </ul> <p>Reasons <b>not</b> to normalize fields:</p> <ul style="list-style-type: none"> <li>◆ faster performance</li> <li>◆ keys are not allowed on Occurs fields if normalized.</li> </ul>
<p><sup>1</sup>The alias name must be 20 characters or less, and may contain only the following characters:</p> <p>a through z  A through Z  0 through 9  _ (underscore)  ^ (caret)  ~ (tilde)  \$ (dollar sign)</p> <p>The name must begin with a letter.</p> <p><b>Note:</b> Certain words are reserved by Pervasive PSQL and cannot be used as column names unless enclosed by double quotes. For a list, refer to "SQL Reserved Words" in the Pervasive PSQL <i>SQL Engine Reference</i>.</p>	

Column names and XFD file names that exceed the maximum lengths allowed by Pervasive PSQL result in alias names being generated in the XML plan. The alias name may not accurately

describe your data when the data is accessed via SQL. Check the log file for the Plan Generator for listing of any alias names created as well as other issues found while creating the plan.

## Examples

Your preexisting source code contains the following compiler directive for a particular file definition:

```
$XFD FILE=A_VERY_VERY_VERY_VERY_LONG_FILE_NAME
```

The XML schema contains something similar to the following:

```
<XFD id="INDEX-FILE"  
file="A_VERY_VERY_VERY_VERY_LONG_FILE_NAME"  
alias="PSQL_TBL_1" fileType="12" maxRecSz="194"  
minRecSz="194" >
```

If you prefer a different file name, change PSQL\_TBL\_1 as described in Table 2-2.

Your preexisting source code contains the following compiler directive for a particular file description:

```
$XFD FILE=TABLE_X
```

The XML schema contains something similar to the following:

```
<XFD id="INDEX-FILE" file="TABLE_X" fileType="12"  
maxRecSz="194" minRecSz="194" >
```

Assume that your application actually opens file TABLE\_X.dat. You elect to change the XML plan manually rather than use the -x option with Plan Executor. Change the line to this (boldface added for emphasis):

```
<XFD id="INDEX-FILE" file="TABLE_X.dat"  
fileType="12" maxRecSz="194" minRecSz="194" >
```

A file description block contains a column named MIGHTY-LONG-COLUMN-NAME.

The XML schema contains something similar to the following:

```
<FIELD id="MIGHTY-LONG-COLUMN-NAME"  
alias="PSQL_COL_1" offset="0" size="2"  
sourceType="10" precision="4" scale="0"  
convertID="10" />
```

If you prefer a different column name, change PSQL\_COL\_1 as described in Table 2-2.



---

**Note** You may also need to edit the XML schema if you have special data types that require custom mapping. Generally, such mapping is not required. For further information, see “Special Data Type Mapping” on page B-1.

---

## Step 6: Creating a Pervasive PSQL Database

You now need to create a Pervasive PSQL database for your application. The Active Connector SDK contains a command-line utility for this purpose. The utility executable name is `serverdsn.exe`, and is referred to as `serverdsn`. (You may also create a database with the Create Database wizard provided with the Pervasive PSQL product. This document discusses only the command-line utility because the other SDK utilities are also command-line programs.)

The conceptual aspects of Pervasive PSQL databases, such as security, referential integrity, and so forth, are beyond the scope of the document. If interested, refer to the documentation provided with the Pervasive PSQL database product, particularly the *Pervasive PSQL User's Guide* and *Advanced Operations Guide*.



**Note** If your application references ISAM files in multiple directories, create a separate Pervasive PSQL database for each directory.

---

### **Example**

You want to create a Pervasive PSQL database named `mydbase` and place the dictionary files in directory `C:\myapp\data`. Enter the following at the command prompt:

```
serverdsn -serverdsn=mydbase -dictpath=c:\myapp\  
data -ri=off -createddf=on
```

### **Serverdsn Command Format**

```
serverdsn -serverdsn=svrdsn -dictpath=ddfpath  
[-dbname=dname] [-datapath=dpath] [-server=<localhost |  
sname | IP_address>] [-security=<classic | database |  
mixed>] -ri=<on | off> [-bound=<on | off>]  
-createddf=<on | off> [-user=uname] [-password=password]
```

Table 2-3 Options for Serverdsn Utility

Option	Meaning
-serverdsn= <i>svrdsn</i>	<i>Svrdsn</i> specifies the server data source name (DSN) of the database to create ( <b>required</b> ).  A (DSN) provides the operating system with information about a specific database that an Open Database Connectivity (ODBC) driver needs to connect to the database. For further information, see “Clients and Data Source Names” on page C-1.
-dictpath= <i>ddfpath</i>	<i>Ddfpath</i> specifies the path to the data dictionary files (DDFs) ( <b>required</b> ). DDFs are the schema representation of a Pervasive PSQL database.
-dbname= <i>dname</i>	The database name ( <i>dname</i> ) associated with the DSN (optional). Default: same as <i>svrdsn</i> .  A DSN may have the same name as its associated database. Although this is not mandatory, it is often convenient for reference to have them named the same. The serverdsn utility applies the same name to both if ?dbname is omitted.
-datapath= <i>dpath</i>	The path ( <i>dpath</i> ) to the data files of the database (optional). Default: same as <i>ddfpath</i> .
-server=<localhost   <i>sname</i>   <i>IP_address</i> >	Specifies the server that houses the database (optional). Default: localhost (meaning the machine on which serverdsn is running)
-security=<classic   database   mixed>	Security level of the database (optional). Default: classic
-ri=<on   off>	Relational Integrity (on, off). The -ri option is <b>required</b> , but can specify either “on” or “off.” Default: on
-bound=<on   off>	Bound Database (on, off) (optional). Default: off

Table 2-3 Options for Serverdsn Utility

Option	Meaning
-createddf=<on   off>	Automatically create DDFs (on, off). The -createddf option is <b>required</b> and must specify "on." Default: off
-user= <i>uname</i>	User login name ( <i>uname</i> ) on remote server (optional).
-password= <i>pwd</i>	The user's password ( <i>pwd</i> ), if any, required to access the remote database. This option is required if the -user option is used. If the user has no password, use "null" as the password.

## Step 7: Installing and Configuring the Runtime Components

After you create a Pervasive PSQL database, you are ready to install the runtime components and configure them.

### **Installation**

Active Connector supports two versions of runtime components for AcuCOBOL: 5.21 and 6.00/6.10. The Active Connector distribution CD has a separate directory for each version because the COBOL runtime file differs for each version.

#### ► **To install the runtime components for Active Connector for Windows**

- 1 Close all AcuCOBOL applications that may be running.
- 2 Insert the distribution CD for Pervasive PSQL Active Connector for AcuCOBOL into a CD drive.
- 3 Open the directory appropriate for your version of AcuCOBOL:
  - AcuGT5.21 for version 5.21
  - AcuGT6.00 for version 6.00
  - AcuGT6.10 for version 6.10

- 4 Copy the following file from the appropriate directory to the working directory of the application (the directory from where wrun32.exe is run):

cblconfig.psql

- 5 Copy the following files from the appropriate directory to the "bin" directory of AcuCOBOL (AcuGT\bin).

wrun32.dll (this file replaces the existing one)



**Note** Since you are replacing wrun32.dll, you may want to backup this file with a file copy before replacing it with the one from the SDK.

---

- 6 Click "Yes" to any copy messages about replacing wrun32.dll. Your existing wrun32.dll file must be replaced by the one supplied with Active Connector.
- 7 Copy the following files from the appropriate directory to the "bin" directory of Pervasive PSQL (c:\pvsw\bin).

conversion.dll

mkc3alch.dll

- 8 Change directory to where you copied conversion.dll and mkc3alch.dll (c:\pvsw\bin).
- 9 Register mkc3alch.dll to the operating system by entering the following command at the command prompt:

```
psregsvr.exe mkc3alch.dll
```

After registration completes, a screen message informs you that it succeeded. You do not need to register the other libraries.



---

**Note** Registration is a process by which you make the operating system aware of libraries required by a software product. Psregsvr.exe is a registration program provided with the Pervasive PSQL database product. The mkc3alch.dll library must be registered for Active Connector to function correctly.

---

## Configuration

You need to edit the configuration file that you just installed, **cblconfig.psql**, to provide information to the Active Connector.



---

**Caution** Active Connector requires its own configuration file, which must be located in the same directory as the COBOL runtime. Do **not** combine the contents of cblconfig.psql with your application's configuration file.

---

### ➤ To edit the cblconfig.psql configuration file

- 1 Start a text editor.
- 2 Open the file cblconfig.psql.

### 3 Provide the following information for the configuration options:

Table 2-4 Configuration Options

Configuration Option	Discussion
<p><code>A_PSQL_SERVER_NAME</code> <i>sname</i></p>	<p>You may leave <i>sname</i> blank if the database engine is local. That is, if the engine is running on the same machine as the COBOL application.</p> <p>If the database engine is running on a remote machine, then specify for <i>sname</i> the name or IP address of the server running the Pervasive PSQL database engine.</p> <p>Example</p> <p>The database is running on a remote server named <code>myremotesvr</code>:</p> <pre>A_PSQL_SERVER_NAME myremotesvr</pre>
<p><code>A_PSQL_DATABASE_NAME</code> <i>dbname</i></p>	<p><i>Dbname</i> is the name of the Pervasive PSQL database. (For example, the name of the database that you created in “Step 6: Creating a Pervasive PSQL Database” on page 2-14.)</p> <p>This option is required.</p> <p>Example</p> <p>The name of the database used by your application is <code>mydbase</code>:</p> <pre>A_PSQL_DATABASE_NAME mydbase</pre>
<p><code>A_PSQL_RELATIVE_FILES_USE_VISION</code> 0</p>	<p>The valid values for this option are 0 and 1 (zero and one), indicating false and true, respectively. The default is 0.</p> <p>The value 0 directs Active Connector to use a Pervasive PSQL database for storage and retrieval of data in relative files. (Note that sequential files remain the same and are routed by Active Connector to the Vision file system.)</p> <p>The value 1 directs Active Connector to route all processing of relative files to Vision.</p>

Table 2-4 Configuration Options

Configuration Option	Discussion
A_PSQL_INDEXED_FILES_USE_VISION 0	<p>The valid values for this option are 0 and 1 (zero and one), indicating false and true, respectively. The default is 0.</p> <p>The value 0 directs Active Connector to use a Pervasive PSQL database for storage and retrieval of data in indexed files. (Note that sequential files remain the same and are routed by Active Connector to the Vision file system.)</p> <p>The value 1 directs Active Connector to route all processing of indexed files to Vision.</p>
A_PSQL_ALT_x_SERVER_NAME <i>altsname</i>	<p>The value of <i>x</i> is a number starting with 1 and incrementing by one for each <b>set</b> of the following:</p> <ul style="list-style-type: none"> <li>◆ A_PSQL_ALT_x_SERVER_NAME</li> <li>◆ A_PSQL_ALT_x_DATABASE_NAME</li> <li>◆ A_PSQL_ALT_x_PATH</li> </ul> <p>The A_PSQL_ALT_x_SERVER_NAME option identifies the Pervasive PSQL server for the database specified in this set.</p> <p>You may leave <i>altsname</i> blank if the database engine is local, that is, if the engine is running on the same machine as the COBOL application.</p> <p>If the database engine is running on a remote machine, then specify for <i>altsname</i> the name or IP address of the alternate server running the Pervasive PSQL database engine.</p> <p>Example</p> <p>You have an alternate set of options and your alternate database is running on a remote server named myserver2:</p> <pre>A_PSQL_ALT_1_SERVER_NAME myserver2</pre>

Table 2-4 Configuration Options

Configuration Option	Discussion
<p>A_PSQL_ALT_x_DATABASE_NAME <i>altdbname</i></p>	<p>The value of <i>x</i> is a number starting with 1 and incrementing by one for each <b>set</b> of the following:</p> <ul style="list-style-type: none"> <li>◆ A_PSQL_ALT_x_SERVER_NAME</li> <li>◆ A_PSQL_ALT_x_DATABASE_NAME</li> <li>◆ A_PSQL_ALT_x_PATH</li> </ul> <p>The A_PSQL_ALT_x_DATABASE_NAME option is required for each set to uniquely identify the database.</p> <p><i>AltDbname</i> is the name of the alternate Pervasive PSQL database.</p> <p>Example</p> <p>The name of the alternate database used by your application is mydbase2:</p> <p>A_PSQL_ALT_1_DATABASE_NAME mydbase2</p>
<p>A_PSQL_ALT_x_PATH <i>altpath</i></p>	<p>The value of <i>x</i> is a number starting with 1 and incrementing by one for each <b>set</b> of the following:</p> <ul style="list-style-type: none"> <li>◆ A_PSQL_ALT_x_SERVER_NAME</li> <li>◆ A_PSQL_ALT_x_DATABASE_NAME</li> <li>◆ A_PSQL_ALT_x_PATH</li> </ul> <p>The <b>A_PSQL_ALT_x_PATH</b> option is required for each set to relate the path information supplied by the COBOL application to the appropriate database.</p> <p>Example</p> <p>Your COBOL application references myaltdirectory\myaltfile.dat:</p> <p>A_PSQL_ALT_1_PATH myaltdirectory</p>

Table 2-4 Configuration Options

Configuration Option	Discussion
A_PSQL_TEMP_FILES_USE_VISION 0	<p>The valid values for this option are 0 and 1 (zero and one), indicating false and true, respectively. The default is 0.</p> <p>The value 0 directs Active Connector to route temporary files to Pervasive PSQL.</p> <p>The value 1 directs Active Connector to scan a file path for temporary files when files are opened. The file path is determined by A_PSQL_WORK_FILES. If you set the value to 1, you must also specify a path for A_PSQL_WORK_FILES.</p>
A_PSQL_WORK_FILES <i>temppath</i>	<p>The value of <i>temppath</i> specifies a path that is prepended to all temporary files, like the WORK-PATH AcuGT environment variable. Specify a value for <i>temppath</i> only if A_PSQL_TEMP_FILES_USE_VISION is set to true.</p> <p>Example</p> <p>Vision temporary files are written to c:\tmp:</p> <pre>A_PSQL_TEMP_FILES_USE_VISION 1 A_PSQL_WORK_FILES C:\TMP</pre> <p>All files opened using C:\TMP\* are routed to Vision instead of the Pervasive PSQL engine since there is no point in creating a temporary file for SQL access.</p>




---

**Note** Alternate Database sets must start with 1 and must be sequential with no numbers omitted. Active Connector processes the information in the configuration file until it is unable to find the next sequential set. If there are other sets after an omitted number, they are ignored.

---

- 4 Save the cblconfig.psql file and exit the text editor.

## Step 8: Creating the Pervasive PSQL Metadata

The Active Connector SDK includes a command line utility called the Plan Executor. The utility executable is named `planexec.exe` and is referred to as `planexec`. Plan Executor performs the following actions:

- parses the XML plan you created in Step 4
- creates the internal metadata based on the XML plan
- creates the customer tables for a Pervasive PSQL database based on the XML plan.

For a specified database, Plan Executor generates the following files (SQL tables) for the COBOL metadata:

- `c$file.mkd`
- `c$field.mkd`
- `c$key.mkd`
- `c$cond.mkd`

The MKD files contain the necessary working data to translate between COBOL tables, fields and indexes and Pervasive PSQL tables, fields and indexes. The MKD files are created in the data dictionary directory that you specified in Step 6 for the Pervasive PSQL database.




---

**Note** If your application references ISAM files in multiple directories, execute `planexec` once for the files in **each** directory. In other words, you must create the metadata for each database that you created in “Step 6: Creating a Pervasive PSQL Database” on page 2-14.

---

### Example

You have an XML plan named `myplan.xml` and your Pervasive PSQL database is named `mydbase`. Enter the following at a command prompt:

```
planexec myplan.xml -dbn mydbase
```

### Plan Executor Command Format

```
planexec planname -dbn dbname [-h] [-svr servername]  
[-recreate] [-x ext] [-n] [-l log-file]
```

Table 2-5 Options for SDK Plan Executor Utility

Option	Meaning
<i>planname</i>	The file name of the XML schema plan (the XML plan created in “Step 4: Creating an XML Schema of Your Data” on page 2-7) ( <b>required</b> ).
-dbn <i>dbname</i>	The name of the Pervasive PSQL database for your application ( <b>required</b> ). (For example, the name of the database that you created in “Step 6: Creating a Pervasive PSQL Database” on page 2-14.)
-h or -?	Display command usage (optional). Ignore all other options.
-svr <i>servername</i>	Specifies the name or IP address of a remote Pervasive PSQL server (optional). Use this option if the Pervasive PSQL database engine is running on a different machine than where the COBOL application is running.
-recreate	Deletes all COBOL metadata files and COBOL data files from the Pervasive PSQL database, then recreates them during the processing of the XML schema plan (optional).  <b>Note:</b> All data in the files is lost so use this option with caution.
-x <i>ext</i>	File name extension to add to the file names specified in the XML schema plan (optional). Planexec adds the extension to the file names in the XML plan before executing the plan.  You do <b>not</b> need this option if <ul style="list-style-type: none"> <li>◆ the ISAM data files do not have file name extensions.</li> <li>◆ you manually added the file name extensions to the XML plan as discussed in “Step 5: Editing the XML Schema” on page 2-9. If the data files use different file name extensions, you must manually modify the XML plan.</li> </ul>

Table 2-5 Options for SDK Plan Executor Utility

Option	Meaning
-n	<p>By default, Planexec normalizes fields within each Occurs clauses into a separate table. To disable normalization for specific tables, see Table 2-2, “Edits that May Be Required to an XML Schema”, on page 2-9.</p> <p>Reasons to normalize fields:</p> <ul style="list-style-type: none"> <li>◆ allows the easy access to like data via SQL</li> <li>◆ allows for a large number of Occurs instances (Pervasive PSQL has a limit of 1536 columns per table.)</li> </ul> <p>Reasons not to normalize fields:</p> <ul style="list-style-type: none"> <li>◆ faster performance</li> <li>◆ keys are not allowed on Occurs fields if normalized.</li> </ul>
-l <i>log-file</i>	<p>Log file to use for messages produced during processing of the plan file (optional). The log file also contains the SQL statements that were executed to create the metadata for the files.</p> <p>The default log file is <i>planexec-nnn.log</i> and is located in the same directory from where you execute <i>planexec</i>. The <i>nnn</i> is a unique number assigned during the execution of Plan Executor.</p> <p>Check the log before migrating your data (See “Step 6: Migrating Existing Data Into a Pervasive PSQL Database” on page 3-8) to ensure that your data will be represented properly.</p> <p>See also “Log Messages” on page A-1.</p>

### **Errors Resulting from Fields That Contain Occurs or Redefines**

Plan Executor attempts to create a separate Pervasive PSQL table for each field name in the XML plan that contains an occurs or a redefine. If more than one file description contains the same named field, and that field also contains an occurs or a redefine, duplicate field names result in the XML plan.

Plan Executor reports an error when it attempts to create the Pervasive PSQL table because of the duplicate field names. Plan Executor cannot create a table if one with that same name already exists. Therefore, Plan Executor stops and the following error message appears on the screen:

```
[LNA] [Pervasive] [ODBC Engine Interface]Table or
view already exists
```

This error requires that you edit the XML plan, then re-run Plan Executor.

➤ **To correct the error resulting from fields with the same name**

- 1 Open the planexec-*nnn*.log file in a text editor.
- 2 In the log file, locate the occurrence of the error.

For example, assume that the following lines occur in your log file:

```
03/26/04 15:29:21 INFORM: Create Table "NW_REGION"  
(PrimaryKey IDENTITY NOT NULL, ...)  
03/26/04 15:29:22 INFORM: Create Table "CONTACT"  
(ForeignKey INTEGER NOT NULL, OccursOrder INTEGER  
NOT NULL, "FIRST" CHAR(20) NOT NULL, "LAST"  
CHAR(20) NOT NULL )  
03/26/04 15:29:22 ERROR: [LNA][Pervasive][ODBC  
Engine Interface]Table or view already exists.
```

The table name listed just prior to the error message, in this case "CONTACT," is the table that has a duplicate name.

- 3 Open the XML plan in a text editor.
- 4 Search for all fields or groups with an ID of the duplicate name. For example, you would search for id="CONTACT".

Assume that the following lines occur in your XML plan:

```
<GROUP occurs="10" id="CONTACT" offset="100"  
size="40" >  
<FIELD id="FIRST" offset="100" size="20"  
sourceType="16" precision="20" scale="0"  
convertID="16" />  
<FIELD id="LAST" offset="120" size="20"  
sourceType="16" precision="20" scale="0"  
convertID="16" />  
</GROUP>
```

- 5 Add a unique alias for the ID (observing the naming restrictions explained in “Step 5: Editing the XML Schema” on page 2-9).

Assume that you choose an alias of CONTACT2. The edited line would show the following:

```
<GROUP occurs="10" id="CONTACT" alias="CONTACT2"  
offset="100" size="40" >
```

- 6 Repeat steps 4 and 5 until all duplicate named fields have unique aliases.
- 7 Save the XML plan and exit the text editor.
- 8 At a command prompt, run Plan Executor with the `-recreate` option.

For example, assume that your XML plan is named `myplan.xml` and your Pervasive PSQL database is named `mydbase`:

```
planexec myplan.xml -dbn mydbase -recreate
```




---

**Note** Plan Executor stops whenever it attempts to create a file with a name that already exists. If the XML plan contains several instances of duplicate fields, the same error displays again until the instances for all duplicated fields contain unique aliases. For example, suppose that your XML file contains `id=FIELDA`, `id=FIELDDB`, and `id=FIELDDC`, all three of which contain duplicate names. If you add aliases for all of the duplicates for `FIELDA` then re-run Plan Executor, the utility displays the error when it encountered the first duplicate of `FIELDDB`. If you add aliases for `FIELDDB`, the utility displays the error when it encounters the first duplicate of `FIELDDC`. Therefore, you can either add aliases for all of the duplicates at once, or iteratively add aliases as Plan Executor discovers the duplicates.

---

### **Error Resulting from Too Many Fields**

If the Pervasive PSQL table contains more than 1536 fields, Plan Executor reports the following error:

```
[LNA][Pervasive][ODBC Engine Interface]Too many
columns in create statement.
```

One reason this error may happen is if normalization has been disabled (by use of the `-n` option) and the table contains several occurs statements.

#### ➤ **To correct the error resulting from too many fields**

- 1 Run Plan Executor again and omit the `-n` option.
- 2 If you do **not** want to normalize specific tables, change the `normalize="yes"` option in the XML schema file to `normalize="no"` as described in “Step 5: Editing the XML Schema” on page 2-9.

## Step 9: Converting COBOL Metadata Files to System Tables

This is the final step in the development process. You now need to convert the COBOL metadata files (tables) that were created in “Step 8: Creating the Pervasive PSQl Metadata” on page 2-23 to system tables. System tables cannot be modified by a regular application or by an application user.

The Active Connector SDK provides a utility that creates the following system tables for the COBOL application. The utility executable name is `cddftool.exe` and is referred to as `cddftool`. `Cddftool` creates the following system tables:

- `cfile.ddf`
- `cfield.ddf`
- `ckey.ddf`
- `ccond.ddf`



---

**Note** Perform the following actions before converting the COBOL metadata to system tables:

- Close all Pervasive PSQl utilities, such as Pervasive Control Center (PCC).
  - Ensure that no applications are accessing the COBOL application's data.
  - For the database that you created in “Step 6: Creating a Pervasive PSQl Database”, make a backup copy of the database directory, all data dictionary files (files with file extension `.DDF`), and all data files (if the data files reside in a directory other than the database directory). A file copy suffices for the backup method.
- 

### ➤ To convert COBOL metadata to system tables

- 1 Enter the following command at a command prompt:

```
cddftool md_path
```

where `md_path` is the path to the location of the COBOL metadata files. That is, to the directory that you specified with the `-dictpath` option in “Step 6: Creating a Pervasive PSQl Database” on page 2-14. That directory contains `c$file.mkd`, `c$field.mkd`, `c$key.mkd`, and `c$cond.mkd`.

For example, suppose that your COBOL metadata files reside in `c:\myapp\data`. Enter the following command:

```
cddftool c:\myapp\data
```

The Cddftool utility completes successfully if no errors or warnings display on the screen.



# Deployment

chapter

3

---

## *Tasks for Deploying Pervasive PSQL Active Connector*

The topics in this chapter include:

- “Deployment Tasks” on page 3-2
- “Step 1: Installing a Pervasive PSQL Database Engine” on page 3-3
- “Step 2: Creating a Pervasive PSQL Database” on page 3-4
- “Step 3: Installing and Configuring the Runtime Components” on page 3-5
- “Step 4: Creating the Pervasive PSQL Metadata” on page 3-6
- “Step 5: Converting COBOL Metadata to System Tables” on page 3-7
- “Step 6: Migrating Existing Data Into a Pervasive PSQL Database” on page 3-8

---

## Deployment Tasks

Deployment takes place in the application user's environment, and requires installing and configuring Pervasive PSQL Active Connector. The assumption is that the environment contains a server and one or more clients.

Deployment consists of the following tasks:

Task	Perform on Server <sup>1</sup>	Perform on Client <sup>1</sup>
Step 1: Installing a Pervasive PSQL Database Engine	✓	
Step 2: Creating a Pervasive PSQL Database	✓	
Step 3: Installing and Configuring the Runtime Components	✓	✓
Step 4: Creating the Pervasive PSQL Metadata	✓	
Step 5: Converting COBOL Metadata to System Tables	✓	
Step 6: Migrating Existing Data Into a Pervasive PSQL Database	✓	
<sup>1</sup> "Server" refers to the machine running the Pervasive PSQL database engine. "Client" refers to any machine running your COBOL application that sends data to or receives data from the Pervasive PSQL database on the server.		

## **Step 1: Installing a Pervasive PSQl Database Engine**

A Pervasive PSQl database engine is required to use the runtime components of Active Connector. Refer to “Step 1: Installing a Pervasive PSQl Database Engine” on page 2-3 for information about the required version of Pervasive PSQl.

## Step 2: Creating a Pervasive PSQL Database

This step assumes that you need to create a Pervasive PSQL database for your application. (That is, an existing Pervasive PSQL database does not already exist. If one does exist, however, you may use it and skip this step.)

You may use the command-line utility `serverdsn`, which is provided with the Active Connector SDK. See “Step 6: Creating a Pervasive PSQL Database” on page 2-14 for instructions on using `serverdsn`. You may also create a database with the Create Database wizard provided with the Pervasive PSQL product. Refer to the chapter “Using the Pervasive Control Center” in *Pervasive PSQL User's Guide*.

## **Step 3: Installing and Configuring the Runtime Components**

Follow the instructions for “Step 7: Installing and Configuring the Runtime Components” on page 2-17. Provide the configuration information in `cblconfig.psql` that applies to the application user's environment. You must register `mkc3alch.dll` to the operating system on the server and on **each** client machine. The server and **each** client machine must also have a `cblconfig.psql` configuration file.

## Step 4: Creating the Pervasive PSQL Metadata

For this step, you need the following:

- planexec (provided with the Active Connector SDK)
- cblrd.dll or libcblrb.so.1.xxx (provided with the Active Connector SDK)
- The XML plan that you developed in Steps 4 and 5 under "Development Tasks."

You must register cblrd.dll or libcblrb.so.1.xxx to the operating system as you did in "Step 2: Installing the SDK and Runtime Components" on page 2-4. Once registered, execute the XML plan as explained in "Step 8: Creating the Pervasive PSQL Metadata" on page 2-23. Use the database name that applies to the application user's environment.

## **Step 5: Converting COBOL Metadata to System Tables**

For this step, you use the `cddftool` utility, `cddftool.exe`, which is provided with the Active Connector SDK. Follow the instructions as explained in “Step 9: Converting COBOL Metadata Files to System Tables” on page 2-28.

## Step 6: Migrating Existing Data Into a Pervasive PSQl Database

To migrate existing data, you perform the following actions:

- Create a migration configuration file
- Copy the list file of Vision data files to migrate
- Migrate the data

### **Creating a Migration Configuration File**

The migration configuration file is a text file that contains only one line:

```
FILE-PREFIX = path_to_files_to_migrate
```

where *path\_to\_files\_to\_migrate* is a path to the existing Vision data files. For example, if the Vision data files reside in directory `c:\mycobolapp\mydata`, the configuration file contains the following line:

```
FILE-PREFIX = c:\mycobolapp\mydata
```

You may name the configuration file whatever you choose. We suggest `cblconfig.migrate` so that you can easily identify the file by name.

### **Copying the List File of Data Files to Migrate**

In “Step 4: Creating the Pervasive PSQl Metadata” on page 3-6, Plan Executor automatically creates a list file of the Vision data files to migrate. The list file is named `infile.txt`. Copy `infile.txt` to the directory where the Vision data files are located. Do **not** rename `infile.txt`.

For example, continuing with the example, you would execute the following at a command prompt:

```
copy infile.txt c:\mycobolapp\mydata
```

Notice that `c:\mycobolapp\mydata` is the same file prefix information that you included in the migration configuration file.

### **Migrating the Data**

The Active Connector SDK provides COBOL programs that perform the data migration:

- `visionpsql_v5.obj` (for AcuCOBOL version 5.21)
- `visionpsql_v6.obj` (for AcuCOBOL version 6.00 or 6.10)

Perform the following steps to migrate data:

- 1 Change directory to where the COBOL runtime is located (AcuGT\bin, for example).
- 2 Copy the migration program for your version of AcuCOBOL, visionpsql\_v5.obj or visionpsql\_v6.obj, from the Active Connector CD to the directory where the COBOL runtime resides.
- 3 Execute the command appropriate for your operating system at a command prompt:

Windows:

```
wrun32 -c migration_config_file visionpsql_vN.obj
```

where *migration\_config\_file* is the configuration file that you created for "Creating a Migration Configuration File," and *N* is either 5 or 6 depending on your version of AcuCOBOL. For example, suppose that your configuration file is named *cblconfig.migrate* and you are using version 6 of AcuCOBOL. You would execute the following command:

```
wrun32 -c cblconfig.migrate visionpsql_v6.obj
```

A message window appears on the screen and informs you of each file that is being migrated. After all files have been migrated, the message window closes and the command prompt becomes active again.



---

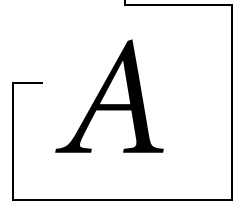
**Note** If you created multiple databases, perform the data migration steps for each of the databases. *Visionpsql\_v5.obj* and *visionpsql\_v6.obj* operate on a single directory. Therefore, for each database, take the following actions before performing the migration steps:

- edit the *migration\_config\_file* to specify the data to be migrated.
  - edit the *cblconfig.psql* configuration file to specify the *A\_PSQL\_SERVER\_NAME* option and the *A\_PSQL\_DATABASE\_NAME* option for the desired database. The options that you need to modify in *cblconfig.psql* are not the ones for Alternate descriptions, but the main descriptions.
- 

You are now ready to run your COBOL application.



# *Log Messages*



---

## *Messages Logged by SDK Utilities*

This appendix lists the messages that may be written to a log file by the SDK utilities and contains the following topics:

- “Plan Generator Log Messages” on page A-2
- “Plan Executor Log Messages” on page A-4

## Plan Generator Log Messages

By default, the log file for Plan Generator is **plangen.log**.

Table A-1 Plan Generator Log Messages

Message	Meaning
Error: cannot open log file, xxxx, #	Log file xxxx cannot be accessed.
Error: cannot open input file, xxxx, #	Input file xxxx either doesn't exist or read access is denied.
Error: cannot open output file, xxxx, #	Write access is denied on Output file xxxx.
Error: cannot access directory, xxxx	This message displays when the -i option is used and the directory given does not exist or its status cannot be read.
Error: xxxx is not a directory	This message displays when the -i option is used and the parameter given is not a directory.
Error: cannot read directory, xxxx	This message displays when the -i option is used and the directory given does not exist or its status cannot be read.
XFD: Unknown error	Unrecognized syntax element found
Error: failed processing, filename (reason)	Parser failure on the file as explained by (reason).
Error: in [Identification Section]	Parser cannot continue in [Identification section].
Error: in [Key Section]	Parser cannot continue in [Key section].
Error: in [Condition Section]	Parser cannot continue in [Condition section].
Error: in <FIELDS>	Parser cannot continue in [Field section].
Error: <XFDS>	Unrecognized syntax element found when processing XFD files.
Error: Unknown content	Unrecognized syntax element found.

Table A-1 Plan Generator Log Messages

Message	Meaning
Error: XFD: Unknown error	Unrecognized syntax element within XFD file.
Error: The system cannot find the file specified, xxxx	This message displays when the -i option is used and system cannot find the specified XFD files
Error: failed processing, xxxx (yyy)	This message displays when the -i option is used and processing failed on a specific XFD file
Warning file xxxx is Pervasive PSQL keyword. Please name an alias	This message displays when an XFD file name is the same as one of the Pervasive PSQL reserved words. This is a warning, reminding you to give the file an alias.
Warning group xxxx is Pervasive PSQL keyword. Please name an alias	This message displays when a group name in XFD is the same as one of the Pervasive PSQL reserved words. This is a warning, reminding you to give the group an alias.
Warning field xxxx is Pervasive PSQL keyword. Please name an alias	This message displays when a field name in XFD is the same as one of the Pervasive PSQL reserved words. This is a warning, reminding you to give the field an alias.

## Plan Executor Log Messages

The log file for Plan Executor is **planexec-*nnn*.log**, where *nnn* is the process ID for each execution of Plan Executor.

Table A-2 Plan Executor Log Messages

Message	Meaning
100: PlanExecutor command line usage:planexec xmlfile.xml -dbn databaseName [-svr serverName] [-recreate] [-x ext]Default serverName=localhost	The planexec command line contained -?, -h, or had an error in the options. The xmlfile.xml and -dbn databaseName options are required. Validate the command line is formatted correctly.
101: PlanExecutor reported x errors and y warnings while parsing the XML file.Database files were not created.Check planexec.log.	The utility could not create the metadata for the Pervasive PSQL database. This message is reported to the screen. Check the log file for more information on the x number of errors and y number of warnings.
102: PlanExecutor reported x warnings while parsing the XML file.Creating database files anyway.Check planexec.log.	This message is reported to the screen. Check the log file for more information on the x number of warnings.
103: PlanExecutor Error, check command line options for valid XML file and database names. Verify the Pervasive PSQL engine is started.	Plan Executor was unable to complete the request. Follow the instructions in the error message.
104: xxxxxxxx: PlanExecutor was unable access the MKC3ALCH COM object. Make sure the MKC3ALCH component is installed and it has been registered with PSREGSVR.	Follow the instructions in the message.
110: <?> tag is before <XFD> tag. This entry will not be processed.	The planexec utility encountered an error parsing the XML file: "?" is a tag name. Run plangen again to re-create the XML file.

Table A-2 Plan Executor Log Messages *continued*

Message	Meaning
<p>111: &lt;FIELD&gt; id (?) is too long. Pervasive PSQL requires 255 characters or less.</p> <p>111: &lt;FIELD&gt; alias (?) is too long. Pervasive PSQL requires 20 characters or less.</p> <p>111: &lt;GROUP&gt; id (?) is too long. Pervasive PSQL requires 255 characters or less.</p> <p>111: &lt;GROUP&gt; alias (?) is too long. Pervasive PSQL requires 20 characters or less.</p> <p>111: &lt;XFD&gt; file name (?) is too long. Pervasive PSQL requires 255 characters or less.</p> <p>111: &lt;XFD&gt; alias (?) is too long. Pervasive PSQL requires 20 characters or less.</p>	<p>"?" is an id , alias or XFD name. Modify the name in the XML file to be within the specified range.</p>
<p>112: &lt;FIELD&gt; id (?) is too long, no alias specified. Pervasive PSQL requires 20 characters or less.</p> <p>112: &lt;GROUP&gt; id (?) is too long, no alias specified. Pervasive PSQL requires 20 characters or less.</p> <p>112: &lt;XFD&gt; file name (?) is too long, no alias specified. Pervasive PSQL requires 20 characters or less.</p>	<p>"?" is an id , alias or XFD name. Modify the name in the XML file to be within the specified range or add an alias to the file within the specified range.</p>
<p>113: &lt;FIELD&gt; id not specified.</p> <p>113: &lt;GROUP&gt; id not specified.</p> <p>113: &lt;XFD&gt; file name not specified.</p>	<p>The specified tag attribute is not present in the XML file. Edit the XML file to add the name or run plangen again to create a new XML file.</p>
<p>114: &lt;FIELD&gt; size (?) must be greater than zero for (??).</p> <p>114: &lt;FIELD&gt; convertID (?) must be greater than zero for (??).</p>	<p>"?" is the size found in the XML file and "??" is the field name. Edit the XML file to correct the value(s) or run plangen again to create a new XML file.</p>

Table A-2 Plan Executor Log Messages *continued*

Message	Meaning												
<p>115: &lt;If&gt; num (?) must be greater than zero.</p> <p>115: &lt;Else&gt; num (?) must be greater than zero.</p> <p>115: &lt;Logical&gt; num (?) must be greater than zero.</p> <p>115: &lt;Logical&gt; cond1 (?) must be greater than zero.</p> <p>115: &lt;Logical&gt; cond2 (?) must be greater than zero.</p>	<p>"?" is the value found in the XML. Edit the XML file to correct the value or run plangen again to create a new XML file.</p>												
<p>116: &lt;If&gt; condition has an invalid op attribute (?).</p>	<p>"?" is the operation specified. Op must be one of the following:</p> <table> <tr> <td>EQUAL</td> <td>1</td> </tr> <tr> <td>GREATER</td> <td>4</td> </tr> <tr> <td>GREATER_OR_EQUAL</td> <td>5</td> </tr> <tr> <td>LESS</td> <td>6</td> </tr> <tr> <td>LESS_OR_EQUAL</td> <td>7</td> </tr> <tr> <td>NOT_EQUAL</td> <td>8</td> </tr> </table>	EQUAL	1	GREATER	4	GREATER_OR_EQUAL	5	LESS	6	LESS_OR_EQUAL	7	NOT_EQUAL	8
EQUAL	1												
GREATER	4												
GREATER_OR_EQUAL	5												
LESS	6												
LESS_OR_EQUAL	7												
NOT_EQUAL	8												
<p>117: &lt;If&gt; tag missing field attribute.</p> <p>117: &lt;If&gt; tag missing value attribute.</p> <p>117: &lt;Else&gt; tag missing field attribute.</p>	<p>The specified attribute is not present. Edit the XML file and add the value or run plangen again to create a new XML file.</p>												
<p>118: &lt;Part&gt; characters missing. The data should be the corresponding field's name.</p>	<p>The Part's character data is not present. Edit the XML file to add the data or run plangen again to create a new XML file.</p>												
<p>119: &lt;Logical&gt; condition has an invalid op attribute (?). It must be 2 (AND) or 9 (OR).</p>	<p>"?" is the op specified. Edit the XML file to specify either 2 or 9.</p>												
<p>120: &lt;XFD&gt; fileType (?) must be 8 (relative) or 12 (indexed).</p>	<p>"?" is the fileType specified. Planexec does not convert sequential files or any other file types other an relative and indexed. Edit the XML file to remove this XFD definition or change the fileType to the correct value.</p>												
<p>121: &lt;XFDS&gt; tag is after &lt;XFD&gt; tag.</p>	<p>The XML file is corrupt. Edit the XML file to correct or run plangen again to create a new XML file.</p>												

Table A-2 Plan Executor Log Messages continued

Message	Meaning
122: Unknown tag (?).	"?" is the errant tag. Edit the XML file to correct or run plangen again to create a new XML file.
123: PlanExecutor received a basic XML error. Check tags or run PlanGen to create a new XML file.	Edit the XML file to correct the error or run plangen again to create a new XML file.
130: More <Part> tags (?) are processed than were specified in the <Key> numParts (??) attribute.	"?" is the number of part tags that have been processed and "??" is the numParts value specified for its key. Planexec creates the key with all of the parts regardless of the numParts value.
131: More <Key> tags (?) are processed than were specified in the <Keys> numKeys (??) attribute.	"?" is the number of key tags that have been processed and ?? is the numkeys value specified for its keys tag. Planexec creates all of the keys regardless of the numkeys value.
132: More <If>, <Else>, or <Logical> tags (?) are processed than were specified in the <Conditions> numConditions (??) attribute.	"?" is the number of condition tags that have been processed and "??" is the numconditions value specified for its Conditions tag. Planexec creates all of the conditions regardless of the numconditions value.
133: <Else> condition has an invalid op1 attribute (?), it will be ignored. The value should be 3 (OTHER).	"?" is the op1 value specified. Planexec ignores the value and creates the condition with an operator value of OTHER.

Table A-2 Plan Executor Log Messages *continued*

Message	Meaning
<p>140: xxxxxxxx: Error initializing MetaData generator.</p>	<p>An error occurred when planexec was preparing the metadata for the P.SQL information. "xxxxxxx" is the error code returned. The code may be a Btrieve error in the form 8004xxxx, where xxxx is the Btrieve error.</p> <p>(Refer to <i>Status Codes and Messages</i>, provided with the Pervasive PSQL database documentation, for an explanation of the Btrieve error codes.)</p> <p>This error may be appended with the following messages.</p> <p><b>The record has a key field containing a duplicate key value.</b> Perhaps the metadata already exists. Try using the -recreate option when running planexec to remove all of the old information.</p> <p><b>A Key specifies a field that does not exist.</b> In the XML file, find the table information for the last table listed on the screen or in planexec.log. Verify that the key information contains valid field names. Also verify that the field does not have an Occurs clause nor is it within a Group that has an Occurs clause. Planexec does not allow keys on fields within Occurs clauses. See Step 5 and Step 8 for ways to disable normalization to prevent this error.</p> <p><b>The number of keys specified is invalid.</b> Pervasive PSQL allows a maximum of 119 segments.</p> <p><b>A condition specifies a field that does not exist.</b> In the XML file, find the table information for the last table listed on the screen or in planexec.log. Verify that the condition information contains valid field names.</p> <p><b>The key length is invalid.</b> Pervasive PSQL allows a maximum key size of 255 bytes.</p>

Table A-2 Plan Executor Log Messages *continued*

Message	Meaning
141: xxxxxxxx: Error persisting COBOL MetaData data.	<p>An error occurred when planexec was persisting the metadata for the P.SQL information to the metadata files. "xxxxxxx" is the error code returned. The code may be a Btrieve error in the form 8004xxxx, where xxxx is the Btrieve error.</p> <p>(Refer to <i>Status Codes and Messages</i>, provided with the Pervasive PSQL database documentation, for an explanation of the Btrieve error codes.)</p> <p>This error may be appended with the same messages as mentioned for "140: xxxxxxxx: Error initializing MetaData generator."</p>
142: xxxxxxxx: An error occurred while creating SQL statements to generate the Btrieve metadata for the COBOL file.	An internal error occurred. Contact Pervasive Software support.
143: xxxxxxxx: PlanExecutor was unable to set the URI for the specified database and server names.	Verify the database has been created on the server specified and that the engine has been started on the server.



# *Special Data Type Mapping*

# *B*

---

## *Situations That May Require Data Type Mapping*

This appendix contains the following topics:

- “Overview” on page B-2
- “Creating a Custom Data Conversion Routine” on page B-3
- “Modify Header Files” on page B-4
- “Compiling the Conversion Library” on page B-5

## **Overview**

The Pervasive PSQL Active Connector for AcuCOBOL handles all data mapping between COBOL and Pervasive PSQL for all standard COBOL data types. The conversion.dll library provided with the SDK contains a set of conversion routines for all of the standard AcuCOBOL data types. You may modify this library if required to add your own data conversion routines. This appendix explains how.

If your application uses special data types—a COMP-6 COBOL type used to store a date, for example—you need to write custom data conversion routines to handle the special data. For instance, suppose that you want to access the COMP-6 data type via SQL as an SQL DATE field. You would need to provide two custom conversion routines in conversion.dll. One routine to convert from COMP-6 to DATE and one to convert from DATE to COMP-6. These routines would allow the same data to be shared by the COBOL application and by SQL applications.

The SDK provides the source code with which you can write custom data conversion routines to handle special data.

---

## Creating a Custom Data Conversion Routine

Complete the following tasks to create a custom conversion routine:

- Export functions from conversion.dll
- Add your custom routines to the C++ source file
- Modify the appropriate header files
- Compile conversion.dll

### ***Exporting Functions from Conversion.dll***

Export the following functions from the version of conversion.dll provided with the SDK:

```
EXPORTS
    getInstance    @1
    getSQLType    @2
    NativeToDB    @3
    DBToNative    @4
```

You export these functions for information purposes only so that you know the entry points of these functions in conversion.dll. Do **not** change these entry points.

### ***Adding Your Customer Routines to Conv.cpp***

The base class for the conversion routines is ACConvertor in file conv.cpp. Add your ordinal to the DBToNative method and the appropriate method call into DBToNativeFunction and NativeToDBFunction objects. (You use this same ordinal for the special COBOL type in the XML plan as explained next in "Modify Header Files").

Active Connector receives field information from the metadata produced from the XML plan. It then provides the offset and length of the data to be converted to the conversion.dll library. The library processes the data with the conversion routines.

## Modify Header Files

You need to modify the following header files:

- Conv.h
- DbToNative.h
- NativeToDb.h

### **Conv.h - ACConverter**

Add your method at the end of the ORDINAL enumeration. The position in this enumeration is the ordinal of your conversion routine used by the Pervasive PSQL database. You must also add this ordinal to the XML plan as a convertID field attribute so that Plan Executor can correctly generate the COBOL/SQL metadata.

For example, suppose that you create your own routine to replace the standard COMP-6 conversion routine. You assign an ordinal of 38 to your routine at the end of conv.h. You would change the convertID in the following line in the XML plan:

```
<FIELD id="IE-COMPUTATIONAL-6-TYPE"  
alias="PSQL_COL_12" offset="180" size="4"  
sourceType="10" precision="8" scale="2"  
convertID="10" />
```

The changed line would read as follows:

```
<FIELD id="IE-COMPUTATIONAL-6-TYPE"  
alias="PSQL_COL_12" offset="180" size="4"  
sourceType="10" precision="8" scale="2"  
convertID="38" />
```

This change creates the proper metadata to call your routine instead of the standard COMP-6 conversion routine.

### **DbToNative.h - DBToNativeFun ction**

This class is for COBOL-to-Pervasive PSQL conversion routines. Add your custom conversion routine here. The "from" buffer is owned by the COBOL runtime so do **not** modify it during your conversions. Use temporary variables where appropriate.

### **NativeToDb.h - NativeToDBFun ction**

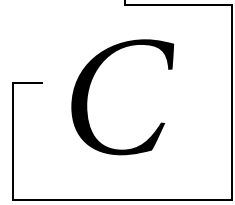
This class is for Pervasive PSQL-to-COBOL conversion routines. Add your custom conversion routine here.

## **Compiling the Conversion Library**

Compile the conversion library with Microsoft Visual C++ 6.0 or later. The project name is **conversion.dsp** and is located on the Active Connector distribution CD in the SKD\conversion directory.



# *Clients and Data Source Names*



---

*Conceptual Information About Data Source Names (DSNs)*

This appendix contains the following topics:

- “Data Source Names” on page C-2

## Data Source Names

A data source name (DSN) provides the operating system with information about a specific database that an Open Database Connectivity (ODBC) driver needs to connect to the database. The DSN information includes the name, directory and driver of the database, and, depending on the type of DSN, the ID and password of the user.

A developer creates a separate DSN for each database. A DSN may have the same name as the database, but this is not mandatory. To connect to a particular database, the developer specifies its DSN within an application program.

### ***Pervasive PSQL and DSNs***

A Pervasive PSQL database on a server requires a DSN. By default, the utilities provided with Pervasive PSQL used to create a database also assign the database name to the DSN. For example, if you use a Pervasive PSQL utility to create a database named **mydbase**, its DSN is also named **mydbase**.

A client DSN is **not** required to run a COBOL application from a client in a client/server environment. "Server" refers to the machine running the Pervasive PSQL database engine. "Client" refers to any machine running your COBOL application that sends data to or receives data from the Pervasive PSQL database on the server.

A client DSN is required on the client machine if your application users want to run SQL queries against the Pervasive PSQL database on the server. The client DSN must specify all of the following:

- The name of the server machine on which the Pervasive PSQL database engine runs.
- The DSN of the server machine that contains the Pervasive PSQL database to which the application on the client machine connects.
- The DSN of the client machine itself.

To create a client DSN, use the **clientdsn.exe** utility provided with the Active Connector SDK. For example, suppose that your server name is **mycblsvr** and the server DSN is **myserver**. You want to name your client DSN **client1**. Execute the following at a command prompt:

```
clientdsn.exe -server=mycblsvr -serverdsn=myserver  
-clientdsn=client1
```



---

**Note** The **clientdsn.exe** utility is available only for the Windows platform.

---



# Index

## Numerics

- 100: PlanExecutor command line usage: planexec xmlfile.xml -dbn databaseName [-svr serverName] [-recreate] [-x ext] Default serverName=localhost A-4
- 101: PlanExecutor reported x errors and y warnings while parsing the XML file. Database files were not created. Check planexec.log. A-4
- 102: PlanExecutor reported x warnings while parsing the XML file. Creating database files anyway. Check planexec.log. A-4
- 103: PlanExecutor Error, check command line options for valid XML file and database names. Verify the Pervasive PSQL engine is started. A-4
- 104: xxxxxxxx: PlanExecutor was unable access the MKC3ALCH COM object. Make sure the MKC3ALCH component is installed and it has been registered with PSREGSVR. A-4
- 110: <?> tag is before <XFD> tag. This entry will not be processed. A-4
- 111: <FIELD> alias (?) is too long. Pervasive PSQL requires 20 characters or less. A-5
- 111: <FIELD> id (?) is too long. Pervasive PSQL requires 255 characters or less. A-5
- 111: <GROUP> alias (?) is too long. Pervasive PSQL requires 20 characters or less. A-5
- 111: <GROUP> id (?) is too long. Pervasive PSQL requires 255 characters or less. A-5
- 111: <XFD> alias (?) is too long. Pervasive PSQL requires 20 characters or less. A-5
- 111: <XFD> file name (?) is too long. Pervasive PSQL requires 255 characters or less. A-5
- 112: <FIELD> id (?) is too long, no alias specified. Pervasive PSQL requires 20 characters or less. A-5
- 112: <GROUP> id (?) is too long, no alias specified. Pervasive PSQL requires 20 characters or less. A-5
- 112: <XFD> file name (?) is too long, no alias specified. Pervasive PSQL requires 20 characters or less. A-5
- 113: <FIELD> id not specified. A-5
- 113: <GROUP> id not specified. A-5
- 113: <XFD> file name not specified. A-5
- 114: <FIELD> convertID (?) must be greater than zero for (?). A-5
- 114: <FIELD> size (?) must be greater than zero for (?). A-5
- 115: <Else> num (?) must be greater than zero. A-6
- 115: <If> num (?) must be greater than zero. A-6
- 115: <Logical> cond1 (?) must be greater than zero. A-6
- 115: <Logical> cond2 (?) must be greater than zero. A-6
- 115: <Logical> num (?) must be greater than zero. A-6
- 116: <If> condition has an invalid op attribute (?). A-6
- 117: <Else> tag missing field attribute. A-6
- 117: <If> tag missing field attribute. A-6
- 117: <If> tag missing value attribute. A-6
- 118: characters missing. The data should be the corresponding field's name. A-6
- 119: condition has an invalid op attribute (?). It must be 2 (AND) or 9 (OR). A-6
- 120: <XFD> fileType (?) must be 8 (relative) or 12 (indexed). A-6
- 121: <XFDS> tag is after <XFD> tag. A-6
- 122: Unknown tag (?). A-7
- 123: PlanExecutor received a basic XML error. Check tags or run PlanGen to create a new XML file. A-7
- 130: More <Par> tags (?) are processed than were specified in the <Key> numParts (??) attribute. A-7
- 131: More <Key> tags (?) are processed than were specified in the <Keys> numKeys (??) attribute. A-7
- 132: More <If>, <Else>, or <Logical> tags (?) are processed than were specified in the

- <Conditions> numConditions (??) attribute. A-7
- 133: <Else> condition has an invalid op1 attribute (?), it will be ignored. The value should be 3 (OTHER). A-7
- 140: xxxxxxxx: Error initializing MetaData generator. A-8
- 141: xxxxxxxx: Error persisting COBOL MetaData data. A-9
- 142: xxxxxxxx: An error occurred while creating SQL statements to generate the Btrieve metadata for the COBOL file. A-9
- 143: xxxxxxxx: PlanExecutor was unable to set the URI for the specified database and server names. A-9

## A

- A\_PSQL\_ALT\_x\_DATABASE\_NAME configuration option 2-21
- A\_PSQL\_ALT\_x\_PATH configuration option 2-21
- A\_PSQL\_ALT\_x\_SERVER\_NAME configuration option 2-20
- A\_PSQL\_DATABASE\_NAME configuration option 2-19
- A\_PSQL\_INDEXED\_FILES\_USE\_VISION configuration option 2-20
- A\_PSQL\_RELATIVE\_FILES\_USE\_VISION configuration option 2-19
- A\_PSQL\_SERVER\_NAME configuration option 2-19
- A\_PSQL\_TEMP\_FILES\_USE\_VISION configuration option 2-22
- A\_PSQL\_WORK\_FILES configuration option 2-22
- Active connector
  - application development
    - converting COBOL metadata files to system tables 2-28
    - creating a Pervasive PSQL database 2-14
    - creating an XML schema 2-7
    - creating Pervasive PSQL metadata 2-23
    - editing the XML schema 2-9
    - installing and configuring runtime components 2-17
    - installing database engine 2-3
    - installing SDK and runtime 2-4
    - setting AcuCOBOL compiler directives 2-5

- components 1-2
- constraints 1-2
- defined 1-2
- deployment tasks 3-2
  - converting COBOL metadata to system tables 3-7
  - creating a Pervasive PSQL database 3-4
  - creating the Pervasive PSQL metadata 3-6
  - installing a Pervasive PSQL engine 3-3
  - installing and configuring runtime components 3-5
  - migrating existing data into a database 3-8
- plan executor utility 2-23
- plan generator utility 2-7
- SDK
  - components 2-4
  - for Windows 2-4
  - installing 2-4
- serverdsn utility 2-14
- system tables 2-28
- tasks 1-3
- tasks for application development 2-2
- Application development
  - converting COBOL metadata files to system tables 2-28
  - creating a Pervasive PSQL database 2-14
  - creating an XML schema 2-7
  - creating Pervasive PSQL metadata 2-23
  - editing the XML schema 2-9
  - installing and configuring runtime components 2-17
  - installing database engine 2-3
  - installing SDK and runtime 2-4
  - setting AcuCOBOL compiler directives 2-5
  - with active connector 2-2

## C

- Cblconfig.psql file 2-18
- ccond.ddf system table 2-28
- cfield.ddf system table 2-28
- cfile.ddf system table 2-28
- ckey.ddf system table 2-28
- Configuration file
  - cblconfig.psql 2-18
  - editing 2-18
  - migration 3-8

## Configuration options

A\_PSQL\_ALT\_x\_DATABASE\_NAME

altdbname 2-21

A\_PSQL\_ALT\_x\_PATH altpath 2-21

A\_PSQL\_ALT\_x\_SERVER\_NAME altlname 2-20

A\_PSQL\_DATABASE\_NAME dbname 2-19

A\_PSQL\_INDEXED\_FILES\_USE\_VISION 0 2-20

A\_PSQL\_RELATIVE\_FILES\_USE\_VISION 0 2-19

A\_PSQL\_SERVER\_NAME sname 2-19

A\_PSQL\_TEMP\_FILES\_USE\_VISION 0 2-22

A\_PSQL\_WORK\_FILES tempopath 2-22

## Configuring

runtime components 2-18

Connecting to a database C-2

## D

Data Source Names C-2

Data type mapping B-2

Data types

mapping B-2

Database

connecting to C-2

Deployment tasks

converting COBOL metadata to system tables 3-7

creating a Pervasive PSQL database 3-4

creating the Pervasive PSQL metadata 3-6

installing a Pervasive PSQL engine 3-3

installing and configuring runtime components 3-5

migrating existing data into a database 3-8

Development

application 2-2

DSN C-2

## E

Error: <XFDS> A-2

Error: cannot access directory, xxxx A-2

Error: cannot open input file, xxxx, # A-2

Error: cannot open log file, xxxx, # A-2

Error: cannot open output file, xxxx, # A-2

Error: cannot read directory, xxxx A-2

Error: failed processing, filename (reason) A-2

Error: failed processing, xxxx (yyy) A-3

Error: in <FIELDS> A-2

Error: in [Condition Section] A-2

Error: in [Identification Section] A-2

Error: in [Key Section] A-2

Error: The system cannot find the file specified, xxxx A-3

Error: Unknown content A-2

Error: XFD: Unknown error A-3

Error: xxxx is not a directory A-2

Errors

table or view already exists 2-25

too many columns in create statement 2-27

## L

Log files

plan executor A-4

plan generator A-2

plangen.log 2-8

Log messages

plan executor A-4

plan generator A-2

## M

Mapping data types B-2

Metadata

converting COBOL metadata files to system tables 2-28

creating for deployment 3-6

Pervasive PSQL 2-23

plan executor utility 2-23

Migration configuration file 3-8

## O

Occurs

errors resulting from 2-25

ODBC C-2

## P

Pervasive PSQL metadata 2-23

Plan executor utility 2-23

-dbn dbname option 2-24

-h or -? option 2-24

-l log-file option 2-25

-n option 2-25

planname option 2-24

-recreate option 2-24

-svr servername option 2-24

-x ext option 2-24

Plan generator utility 2-7

-b option 2-7

-e option 2-7

-h or -? option 2-7

-i xfd-dir option 2-8

-l log-file option 2-8

map-file option 2-8

-n option 2-7

-v option 2-7

-x xdf-ext option 2-8

xdf-file option 2-8

## R

Redefines

errors resulting from 2-25

Runtime components 2-17

configuring 2-18

installing for Windows 2-17

## S

Schema

XML 2-7, 2-9

Serverdsn utility 2-14

-bound= option 2-15

-createddf= option 2-16

-datapath=dpath option 2-15

-dbname=dname option 2-15

-dictpath=ddfpath option 2-15

-password=pword option 2-16

-ri= option 2-15

-security= option 2-15

-server= option 2-15

-serverdsn=svrdsn option 2-15

-user=uname option 2-16

System tables for active connector 2-28

## T

Table or view already exists error 2-25

Too many columns in create statement error 2-27

Too many fields

errors resulting from 2-27

## W

Warning field xxxx is Pervasive PSQL keyword.

Please name an alias A-3

Warning file xxxx is Pervasive PSQL keyword. Please

name an alias A-3

Warning group xxxx is Pervasive PSQL keyword.

Please name an alias A-3

## X

XFD: Unknown error A-2

XML schema

creating 2-7

editing 2-9