SYBASE®

Connection Reference

PocketBuilder™

2.5

DOCUMENT ID: DC00165-01-0250-01

LAST REVISED: December 2007

Sybase, Inc., One Sybase Drive, Dublin, CA 94568.

Contents

About This Book

Audience

This guide is for programmers building applications with PocketBuilder™.

How to use this book

This book describes the database parameters and preferences you use to connect to a database in PocketBuilder.

Related documents

PocketBuilder reference set This guide is part of the PocketBuilder reference set, which is based on PowerBuilder® documentation. The reference set also includes the following manuals:

- *DataWindow Reference* - Lists the DataWindow® functions and properties and includes the syntax for accessing properties and data in DataWindow objects.
- *Objects and Controls* - Describes the system-defined objects and their default properties, functions, and events.
- *PowerScript Reference* - Describes syntax and usage for the PowerScript® language including variables, expressions, statements, events, and functions.

PocketBuilder documentation set The PocketBuilder documentation set includes the following manuals:

- *Introduction to PocketBuilder* - Provides an overview of PocketBuilder features and the PocketBuilder development environment and a tutorial that leads the new user through the basic process of creating and deploying PocketBuilder applications.
- *Resource Guide* - Presents advanced programming techniques and information about connecting to and synchronizing with a database.
- *Users Guide* - Gives an overview of the PocketBuilder development environment and explains how to use the interface. Describes basic techniques for building the objects in a PocketBuilder application, including windows, menus, DataWindow objects, and user-defined objects. An appendix summarizes the differences between PocketBuilder and PowerBuilder.

Online Help Reference information for PowerScript properties, events, and functions is available in the online Help with annotations indicating which objects and methods are applicable to PocketBuilder.

SQL Anywhere® documentation PocketBuilder is tightly integrated with the SQL Anywhere database server and management system (formerly Adaptive Server Anywhere), including its UltraLite®, MobiLink™, and Sybase Central™ components. You can install these products from the PocketBuilder setup program. For an introduction to these products, see Chapter 1 in *Introduction to PocketBuilder*. Documentation for SQL Anywhere is available on the PocketBuilder SyBooks CD and on the iAnywhere Web site at http://www.ianywhere.com/developer/product_manuals/sqlanywhere/.

Other sources of information

Use the Sybase Getting Started CD, the SyBooks CD, and the Sybase Product Manuals Web site to learn more about your product:

- The Getting Started CD contains release bulletins and installation guides in PDF format, and may also contain other documents or updated information not included on the SyBooks CD. It is included with your software. To read or print documents on the Getting Started CD, you need Adobe Acrobat Reader, which you can download at no charge from the Adobe Web site using a link provided on the CD.
- The SyBooks CD contains product manuals and is included with your software. The Eclipse-based SyBooks browser allows you to access the manuals in an easy-to-use, HTML-based format.

  Some documentation may be provided in PDF format, which you can access through the PDF directory on the SyBooks CD. To read or print the PDF files, you need Adobe Acrobat Reader.

  Refer to the *SyBooks Installation Guide* on the Getting Started CD, or the *README.txt* file on the SyBooks CD for instructions on installing and starting SyBooks.
- The Sybase Product Manuals Web site is an online version of the SyBooks CD that you can access using a standard Web browser. In addition to product manuals, you will find links to EBFs/Maintenance, Technical Documents, Case Management, Solved Cases, newsgroups, and the Sybase Developer Network.

  To access the Sybase Product Manuals Web site, go to Product Manuals at http://www.sybase.com/support/manuals/.

Sybase EBFs and software maintenance

❖ **Finding the latest information on EBFs and software maintenance**

1 Point your Web browser to the Sybase Support Page at http://www.sybase.com/support.

2 Select EBFs/Maintenance. If prompted, enter your MySybase user name and password.

3 Select a product.

4 Specify a time frame and click Go. A list of EBF/Maintenance releases is displayed.

Padlock icons indicate that you do not have download authorization for certain EBF/Maintenance releases because you are not registered as a Technical Support Contact. If you have not registered, but have valid information provided by your Sybase representative or through your support contract, click Edit Roles to add the "Technical Support Contact" role to your MySybase profile.

5 Click the Info icon to display the EBF/Maintenance report, or click the product description to download the software.

Conventions

The formatting conventions used in this manual are:

Formatting example	To indicate
Retrieve and Update	When used in descriptive text, this font indicates: • Command, function, and method names • Keywords such as true, false, and null • Datatypes such as integer and char • Database column names such as emp_id and f_name • User-defined objects such as dw_emp or w_main
variable or *file name*	When used in descriptive text and syntax descriptions, oblique font indicates: • Variables, such as *myCounter* • Parts of input text that must be substituted, such as *pklname*.pkd • File and path names

Formatting example	To indicate
File>Save	Menu names and menu items are displayed in plain text. The greater than symbol (>) shows you how to navigate menu selections. For example, File>Save indicates "select Save from the File menu."
`dw_1.Update()`	Monospace font indicates: • Information that you enter in a dialog box or on a command line • Sample script fragments • Sample output fragments

If you need help

Each Sybase installation that has purchased a support contract has one or more designated people who are authorized to contact Sybase Technical Support. If you cannot resolve a problem using the manuals or online help, please have the designated person contact Sybase Technical Support or the Sybase subsidiary in your area.

CHAPTER 1

DBParm Parameters

About this chapter

This chapter describes the syntax and use of each DBParm parameter that you can set in PocketBuilder. Syntax and usage depend on the database interface you are using. Some ODBC DBParms are not valid for use with SQL Anywhere.

Setting DBParms in PowerScript

Use the Preview page of the Database Connection Profile dialog box to ensure that you are using the correct syntax in PowerScript code. Most boolean DBParms can be turned on using any of the values true, Yes, or 1, and turned off using false, No, or 0. Numeric values for DBParms must *not* be enclosed in quotes.

Contents

DBParm parameters and supported database interfaces

The following table lists each supported database interface and the DBParm parameters you can use with that interface in PocketBuilder.

The DBParm parameters are described in alphabetical order following the table.

Database interface	DBParm parameters	
ODBC **Using DBParms with ODBC** These DBParm parameters are supported by the PocketBuilder ODBC interface only if *both* the ODBC driver you are using and the back-end DBMS support the feature.	Async Block CallEscape CommitOnDisconnect ConnectOption ConnectString CursorLib CursorLock CursorScroll Date DateTime DBGetTime DecimalSeparator DelimitIdentifier DisableBind	FormatArgsAsExp IdentifierQuoteChar LoginTimeOut MsgTerse NumericFormat OJSyntax PacketSize PBCatalogOwner PBUseProcOwner RPCRebind SQLCache StaticBind StripParmNames TableCriteria Time
UltraLite	CommitOnDisconnect ConnectString DBTextLimit DelimitIdentifier	FormatArgsAsExp StaticBind TrimSpaces

Alphabetical listing of database parameters

Async

Description

Allows you to perform asynchronous operations on your database in PocketBuilder. In other words, if you have coded a RetrieveRow event for a DataWindow object, before the current database retrieval operation completes, you can cancel it or start another (non-database) operation that does not use the same database connection. You can also switch to another Windows process while the retrieval takes place.

By default, PocketBuilder operates synchronously.

Applies to

ODBC

Syntax

Async = *value*

Parameter	Description
value	A value specifying synchronous or asynchronous operation. Values are: • **0** (Default) Synchronous operation. • **1** Asynchronous operation.

Default value

Async = 0

Usage

Enabling asynchronous operation in PocketBuilder is useful when you are executing a complex SQL statement that takes several minutes to return results. If the Async parameter is set to 1, you can do either of the following while the SQL statement is executing:

- Work in another window
- Cancel the statement before it retrieves the first row of data

When to set Async If you are communicating with the database in a PocketBuilder script, you can reset the Async value at any time before or after the Transaction object has connected to the database.

How data is retrieved When you retrieve data in a DataWindow object, the following steps occur in order:

1 The database server compiles and executes the SQL statement.

2 PocketBuilder retrieves (fetches) the first row of data.

3 PocketBuilder retrieves each subsequent row of data.

What happens before the first row is retrieved While the server is compiling and executing the SQL statement and before PocketBuilder retrieves the first row of data, you need to have done *both* of the following to enable asynchronous operation (allowing you to cancel the current operation before it retrieves the first row of data):

- Coded a RetrieveRow event for the DataWindow object (the code can contain only a comment)
- Set the Async DBParm parameter to 1

What happens after the first row is retrieved After the first row of data is retrieved and between subsequent row fetches, you need to have done only the following to enable asynchronous operation:

- Coded a RetrieveRow event for the DataWindow object

After the first row is retrieved, PocketBuilder operates asynchronously *without your having to set the Async DBParm to 1*, so you can cancel the current operation anytime after it retrieves the first row of data. Therefore, the Async DBParm parameter has no effect in PocketBuilder after the first row of data is retrieved.

Examples

To enable asynchronous operation:

- **Database profile** Select the Asynchronous check box on the Transaction tab in the Database Profile Setup dialog box.
- **PocketBuilder application script** Type the following in a PocketBuilder script:

```
SQLCA.dbParm = "Async = 1"
```

You can set the Async and DBGetTime parameters in a single DBParm statement. DBGetTime specifies the number of seconds you want PocketBuilder to wait for a response from the DBMS when you retrieve rows in a DataWindow object.

To enable asynchronous operation and set the DBGetTime parameter to 20 seconds:

- **Database profile** Select the Asynchronous check box and type 20 in the Number Of Seconds To Wait box on the Transaction tab in the Database Profile Setup dialog box.
- **PocketBuilder application script** Type the following in a PocketBuilder application script:

```
SQLCA.dbParm = "Async = 1, DBGetTime = 20"
```

See also DBGetTime

Block

Description

Block specifies the cursor blocking factor for a database connection. The blocking factor determines the number of rows that a DataWindow object can fetch from the database at one time.

You can improve performance by using the Block DBParm parameter when you access a database in PocketBuilder.

Applies to

ODBC

Syntax

Block = *blocking_factor*

Parameter	Description
blocking_factor	The number of rows you want the DataWindow object to fetch from the database at one time. The blocking factor can be a number from 1 to 1000, inclusive. To turn off block fetching, set Block to 1.

Default value

For most DataWindow objects, the Block default value is `Block=1000`.

If you specified that the DataWindow object should retrieve only as many rows as needed from the database (Retrieve.AsNeeded property), the Block default value is `Block = 100`.

In both cases, there is a maximum limit on the volume of data that can be retrieved of 32K per column.

Using the default blocking factor
You should not have to set a non-default value for Block. In most cases, the default blocking factor used by PocketBuilder should meet your needs.

Usage

Requirements To use the Block DBParm parameter with an ODBC data source, your ODBC driver must:

- Be ODBC Version 2.0-compliant or higher, *and*
- Support the SQLExtendedFetch API call

The SQL Anywhere ODBC driver that comes with PocketBuilder meets both of these requirements.

For information about whether your ODBC driver meets these requirements, see the documentation that comes with your driver.

Determining the Block value PocketBuilder searches the following in this order to determine the Block value for ODBC data sources:

1 The section for your database profile in the PocketBuilder initialization file or the value of the Transaction object DBParm property in a PocketBuilder application.

2 The section for your ODBC driver in the *pkodb25* initialization file.

If PocketBuilder does not find a Block value in these locations, it uses the default Block value for the DBMS you are accessing.

Turning off block fetching To turn off block fetching for an ODBC data source, set the Block parameter to 1.

Examples

To set the blocking factor for DataWindow objects to 50 rows:

- **Database profile** Type `50` in the Retrieve Blocking Factor box on the Transaction tab in the Database Profile Setup dialog box.
- **PocketBuilder application script** Type the following in a PocketBuilder application script:

```
SQLCA.dbParm = "Block = 50"
```

CallEscape

Description

Controls whether the ODBC interface uses call escape syntax for stored procedure calls (the default) or converts the calls to driver-specific native SQL syntax before sending the command to the ODBC driver.

Applies to

ODBC

Syntax

CallEscape = '*value*'

Parameter	Description
value	Controls whether the ODBC interface uses call escape syntax for stored procedure calls or converts the calls to driver-specific native SQL syntax. Values are: • **Yes** (Default) The ODBC interface uses call escape syntax for stored procedure calls. • **No** The ODBC interface converts stored procedure calls to driver-specific native SQL syntax before sending the command to the ODBC driver.

Default value

CallEscape = 'Yes'

Usage

When to use Set CallEscape to No if the ODBC driver you are using expects to receive stored procedure calls in native (driver-specific) SQL syntax instead of in call escape syntax.

For information about the stored procedure call syntax your ODBC driver expects, see your vendor's driver documentation.

Level 2 or higher ODBC driver required To use the CallEscape DBParm parameter, your ODBC driver *must* meet Level 2 or higher API conformance requirements. CallEscape has no effect when you are using an ODBC driver that meets Core or Level 1 API conformance requirements.

Examples

To convert stored procedure calls to native SQL syntax before sending the command to your ODBC driver:

- **Database profile** Clear the Use Call Escape Syntax check box on the Syntax tab in the Database Profile Setup dialog box.
- **PocketBuilder application script** Type the following in a PocketBuilder script:

```
SQLCA.DBParm = "CallEscape = 'No'"
```

The following example shows a call to a stored procedure named sp_test that uses call escape syntax:

```
{call sp_test(1,1)}
```

CommitOnDisconnect

Description

Specifies whether PocketBuilder should commit (the default) or roll back all previously uncommitted database updates before disconnecting from a data source.

When to specify CommitOnDisconnect
You must specify a value for CommitOnDisconnect *before* connecting to the database in PocketBuilder.

Applies to

ODBC
UL9 UltraLite 9
UL10 UltraLite 10

Syntax

CommitOnDisconnect = '*value*'

Parameter	Description
value	Specifies whether PocketBuilder should commit or roll back all previously uncommitted database updates before disconnecting from a data source. Values are: • **Yes** (Default) PocketBuilder commits all uncommitted database updates when an application closes or when an explicit DISCONNECT statement is issued in a PocketBuilder script. • **No** PocketBuilder rolls back all uncommitted database updates when an application closes or when an explicit DISCONNECT statement is issued in a PocketBuilder script. With this setting, PocketBuilder does not automatically commit updates when you disconnect from the database.

Default value

CommitOnDisconnect = 'Yes'

Usage

Set CommitOnDisconnect to No if you want PocketBuilder to roll back uncommitted database updates (instead of automatically committing them when you disconnect from the database).

Examples

To tell PocketBuilder to roll back uncommitted database updates instead of committing them when disconnecting from the database:

- **Database profile** Clear the Commit On Disconnect check box on the Connection tab in the Database Profile Setup dialog box.
- **PocketBuilder application script** Type the following in a PocketBuilder script:

```
SQLCA.DBParm = "CommitOnDisconnect = 'No'"
```

ConnectOption

Description

Sets driver-specific connection options when you are accessing an ODBC data source in PocketBuilder. These options specify the following:

- How the ODBC driver prompts for additional connection information
- The type of security for a Microsoft SQL Server connection
- Whether the ODBC Driver Manager Trace is on or off and what trace file it uses
- Whether cursors are closed or left open on a SQLTransact call
- How temporary stored procedures are treated for a SQLPrepare call

Certain ConnectOption parameters apply to all ODBC drivers, whereas others apply only to particular ODBC drivers.

For information on each ConnectOption parameter and whether you can use it with your ODBC driver, see the table in the Syntax section.

When to specify ConnectOption
You must specify the ConnectOption parameter *before* connecting to an ODBC data source in PocketBuilder. The ConnectOption settings take effect when you connect to the database.

Applies to

ODBC

Syntax

```
ConnectOption = 'SQL_DRIVER_CONNECT,value;
        SQL_INTEGRATED_SECURITY,value;
        SQL_OPT_TRACE,value;
        SQL_OPT_TRACEFILE,value;
        SQL_PRESERVE_CURSORS,value;
        SQL_USE_PROCEDURE_FOR_PREPARE,value'
```

The following table lists the applicable ODBC drivers, purpose, and values for each ConnectOption parameter.

Table 1-1: ConnectOption parameters

Parameter	Description
SQL_DRIVER_CONNECT	**Driver** Any ODBC driver that supports the SQLDriverConnect API call. **Purpose** Specifies how the ODBC driver prompts for additional connection information (such as the user ID and password) when connecting to an ODBC data source. **Values** The values you can specify are: • **SQL_DRIVER_COMPLETE** (Default) If the connection string contains correct and sufficient information to connect, the driver connects to the specified data source. If any information is incorrect or missing, the driver displays one or more dialog boxes to prompt for the required connection parameters. The driver then connects to the specified data source. • **SQL_DRIVER_COMPLETE_REQUIRED** The driver takes the same actions as it does when SQL_DRIVER_COMPLETE is set. In addition, the driver disables the controls for any information not required to connect to the data source. • **SQL_DRIVER_PROMPT** The driver displays one or more dialog boxes to prompt for the required connection parameters. The driver then connects to the specified data source and builds a connection string from the information specified in the dialog boxes. • **SQL_DRIVER_NOPROMPT** If the connection string contains correct and sufficient information to connect, the driver connects to the specified data source. If any information is incorrect or missing, the driver returns an error.
SQL_INTEGRATED_ SECURITY	**Driver** Microsoft SQL Server ODBC driver (not supplied with PocketBuilder). **Purpose** Specifies the type of connection to the Microsoft SQL Server database server. **Values** The values you can specify are: • **SQL_IS_OFF** (Default) Request a normal (nontrusted) connection to SQL Server using standard security. If you specify SQL_IS_OFF, you cannot request a trusted connection to SQL Server using integrated security. • **SQL_IS_ON** Request a trusted connection to SQL Server using integrated security regardless of the login security currently in use on the database server.

Parameter	Description
SQL_OPT_TRACE	**Driver** Any ODBC driver. **Purpose** Turns on or turns off the ODBC Driver Manager Trace used in PocketBuilder to troubleshoot a connection to an ODBC data source. The ODBC Driver Manager Trace provides detailed information about the ODBC API function calls that PocketBuilder makes when connected to an ODBC data source. **Values** The values you can specify are: • **SQL_OPT_TRACE_OFF** (Default) Turns off the ODBC Driver Manager Trace. • **SQL_OPT_TRACE_ON** Turns on the ODBC Driver Manager Trace. For instructions on using the ODBC Driver Manager Trace, see "Using the ODBC Driver Manager Trace tool" in the *Resource Guide*.
SQL_OPT_TRACEFILE	**Driver** Any ODBC driver. **Purpose** Specifies the name of the trace file where you want PocketBuilder to send the output of the ODBC Driver Manager Trace. PocketBuilder appends the output to the trace file you specify until you stop the trace. To display the trace file, you can use the File Editor (in PocketBuilder) or any text editor (outside PocketBuilder). **Values** You can specify any file name for the trace file, following the naming conventions of your operating system. By default, if tracing is on and you have not specified a trace file, PocketBuilder sends ODBC Driver Manager Trace output to the file *SQL.LOG*.
SQL_PRESERVE_ CURSORS	**Driver** Microsoft SQL Server ODBC driver (not supplied with PocketBuilder). **Purpose** Specifies whether cursors are closed or left open on a SQLTransact call. **Values** The values you can specify are: • **SQL_PC_OFF** (Default) Close all cursors on a SQLTransact call. • **SQL_PC_ON** Keep server cursors open on a SQLTransact call.

Parameter	Description
SQL_USE_PROCEDURE_ FOR_PREPARE	**Driver** Microsoft SQL Server ODBC driver (not supplied with PocketBuilder). **Purpose** Specifies how temporary stored procedures are treated for a SQLPrepare call. **Values** The values you can specify are: • **SQL_UP_ON** (Default) Generate temporary stored procedures for a SQLPrepare call. • **SQL_UP_OFF** Do not generate temporary stored procedures for a SQLPrepare call. The SQL statement is stored, compiled, and run at execution time. Syntax error checking does not occur until execution time. • **SQL_UP_ON_DROP** Explicitly drop temporary stored procedures for a subsequent SQLPrepare call or when a statement handle (*hstmt*) is freed for reuse.

Default value

ConnectOption = 'SQL_DRIVER_CONNECT,
SQL_DRIVER_COMPLETE;
SQL_INTEGRATED_SECURITY,SQL_IS_OFF;
SQL_OPT_TRACE,SQL_OPT_TRACE_OFF;
SQL_PRESERVE_CURSORS,SQL_PC_OFF;
SQL_USE_PROCEDURE_FOR_PREPARE,SQL_UP_ON'

Usage

You must obtain the Microsoft SQL Server ODBC driver from Microsoft Corporation. This driver is *not* supplied with PocketBuilder.

Examples

To specify nondefault options for the ConnectOption DBParm parameter:

- **Database profile** Complete the Options tab in the Database Profile Setup dialog box. Each ConnectOption parameter corresponds to an option in the dialog box, as follows:

ConnectOption parameter	Corresponding option
SQL_DRIVER_CONNECT	Connect Type
SQL_INTEGRATED_SECURITY	Integrated Security
SQL_OPT_TRACE	Trace ODBC API Calls
SQL_OPT_TRACEFILE	Trace File
SQL_PRESERVE_CURSORS	Preserve Cursors
SQL_USE_PROCEDURE_FOR_PREPARE	Use Procedure for Prepare

- **PocketBuilder application script** Type the following in a PocketBuilder application script:

```
SQLCA.dbParm = "ConnectOption =
       'SQL_DRIVER_CONNECT,SQL_DRIVER_NOPROMPT;
       SQL_INTEGRATED_SECURITY,SQL_IS_ON;
       SQL_OPT_TRACE,SQL_OPT_TRACE_ON;
       SQL_OPT_TRACEFILE,C:\myapp\odbctrce.log;
       SQL_PRESERVE_CURSORS,SQL_PC_ON;
       SQL_USE_PROCEDURE_FOR_PREPARE,SQL_UP_OFF'"
```

ConnectString

Description

Specifies the parameters required to connect to a database.

Applies to

ODBC
UL9 UltraLite 9
UL10 UltraLite 10

Syntax

The ConnectString syntax displays on a single line. You must enclose the entire ConnectString in single quotes and separate parameters within the ConnectString with semicolons.

ODBC Syntax

ConnectString = 'DSN = *data_source_name*; {UID = *user_ID*;
PWD = *password*; *driver_specific_parameters*}'

Parameter	Description
data_source_name	A name that identifies the data source
user_ID	(Optional) The user ID required to connect to the data source
password	(Optional) The password required by *user_ID* to connect to the data source
driver_specific_parameters	(Optional) Any other driver-specific parameters required to connect

UltraLite Syntax

ConnectString = 'DBF = *database_file_name*;UID = *user_ID*;
PWD = *password*{; *driver_specific_parameters*}'

Parameter	Description
database_file_name	A name that identifies the database file.
user_ID	The user ID required to connect to the database.

Parameter	Description
password	The password required by *user_ID* to connect to the database.
driver_specific_parameters	(Optional) Any other driver-specific parameters required to connect. Do not supply quotes in this string.

Default value

None

Usage

For ODBC, PocketBuilder generates the ConnectString automatically when you define an ODBC data source and copies it to the DBParm box in the Database Profile Setup dialog box. This happens before you connect to the data source in PocketBuilder.

Therefore, *you do not have to enter the ConnectString yourself* when defining an ODBC data source. However, you might need to edit the ConnectString value in the Database Profile Setup dialog box.

You can change the ConnectString parameter if necessary by editing it in the Database Profile Setup dialog box. For example, if you change the name of an existing ODBC data source, edit its database profile to update the connect string with the new DSN (data source name) value.

For UltraLite, you can use the Browse button to locate the UDB file on your file system.

Examples

This example shows a connect string for an ODBC data source that contains the data source name (DSN=Sales), user ID (UID=dba), and password (PWD=sql). Parameters within the connect string are separated by semicolons.

- **Database profile** On the Connection tab in the Database Profile Setup dialog box, select Sales from the Data Source drop-down list, select the User ID check box and type `dba`, and select the Password check box and type `sql`.
- **PocketBuilder application script** Type the following in a PocketBuilder application script:

```
SQLCA.dbParm = "ConnectString ='DSN=Sales;UID=dba;PWD=sql'"
```

This example shows a connect string for an UltraLite database that contains the database file name, user ID, and password, and two additional ConnectString parameters. Parameters within the connect string are separated by semicolons.

- **Database profile** On the Connection tab in the Database Profile Setup dialog box, click the Browse button to navigate to the location of the database file, select the User ID check box and type `dba`, select the Password check box and type `sql`, and type `key=r2viklj;cp=cs1250` in the Additional ConnectString Parameters box.
- **PocketBuilder application script** Type the following in a PocketBuilder application script:

```
SQLCA.dbParm=
"ConnectString='DBF=C:\uleq.udb;UID=dba;PWD=sql;key=r2viklj;cp=cs1250'"
```

CursorLib

Description: Specifies the cursor library to use when connecting to an ODBC data source.

Applies to: ODBC

Syntax: CursorLib = '*value*'

Parameter	Description
value	The cursor library to use when connecting to an ODBC data source. Values are: • **ODBC_Cur_Lib** Use the ODBC Version 2.0 or higher cursor library. • **If_Needed** Use the ODBC Version 2.0 or higher cursor library if your ODBC driver does not support cursors. • **Driver_Cursors** (Default) Use your data source's native cursor support.

Default value: CursorLib = 'Driver_Cursors'

Examples: To specify use of the ODBC Version 2.0 or higher cursor library when connecting to an ODBC data source:

- **Database profile** Select Cursor Library from the Cursor Library drop-down list on the Transaction tab in the Database Profile Setup dialog box.
- **PocketBuilder application script** Type the following in a PocketBuilder application script:

```
SQLCA.dbParm = "CursorLib = 'ODBC_Cur_Lib'"
```

CursorLock

Description

When used with the CursorScroll parameter, specifies locking options for cursors in an ODBC data source.

The values you can set for CursorLock control two aspects of cursor locking:

- **Concurrent access** Ensures that multiple users can simultaneously access data that is accurate and current.
- **Collision detection** Detects collisions that occur when multiple users update the same data at the same time.

Applies to

ODBC

Syntax

CursorLock = '*lock_value*'

Parameter	Description
lock_value	Specifies the type of locking you want to use for ODBC cursors. Values are: • **Lock** Use the lowest level of locking sufficient to allow updates on table rows. • **Opt** Use **optimistic concurrency control**. This means that table rows are not locked against updates by other users. To detect collisions, compare row versions or timestamps. • **OptVal** Use optimistic concurrency control. This means that table rows are not locked against updates by other users. To detect collisions, compare selected values with their previous values. • **ReadOnly** Prohibit updates on table rows by any user. For more about how the ODBC standard defines lock values, see your ODBC documentation.

Default value

If you do not specify a value for CursorLock, PocketBuilder defaults to the cursor lock setting specified by your ODBC driver.

Examples

To set scrolling and locking options for cursors in an ODBC data source:

- **Database profile** Select Dynamic Scrolling from the Scrolling Options drop-down list, and Optimistic Using Values from the Locking drop-down list on the Transaction tab in the Database Profile Setup dialog box.
- **PocketBuilder application script** Type the following in a PocketBuilder application script:

```
SQLCA.dbParm =
      "CursorScroll='Dynamic',CursorLock='OptVal'"
```

See also

CursorScroll

CursorScroll

Description

When used with the CursorLock parameter, specifies scrolling options for cursors in an ODBC data source.

The location of a cursor indicates the current position in the result set produced by a SQL statement. **Scrolling** allows a cursor to move through the data in a result set one row at a time.

Applies to

ODBC

Syntax

CursorScroll = '*scroll_value*'

Parameter	Description
scroll_value	Specifies the type of scrolling you want to use for ODBC cursors. Values are: • **Forward** The cursor only scrolls forward through the result set. • **Static** The data in the result set does not change. • **KeySet** Specifies that the cursor is **keyset-driven**. When a keyset-driven cursor is opened, the driver saves keys for the *entire result set*. As the cursor scrolls through the result set, the driver uses the keys in this **keyset** to retrieve the current values for each row. • **Dynamic** The driver saves and uses only the keys for the rows specified in the rowset.

Default value

If you do not specify a value for CursorScroll, PocketBuilder defaults to the cursor scroll settings specified for your ODBC data source driver.

Usage

For large result sets, it may be impractical to use a keyset-driven cursor that requires the driver to save keys for the entire result set. Instead, you can use a **mixed cursor** by specifying a 32-bit integer value that is the number of rows in your keyset (see Example 2). This number is typically smaller than the result set. The default keyset size is 0.

A mixed cursor uses KeySet scrolling within the specified keyset and Dynamic scrolling outside the keyset.

Examples

To set scrolling and locking options for cursors in an ODBC data source:

- **Database profile** Select Dynamic Scrolling from the Scrolling Options drop-down list and Optimistic Using Values from the Locking drop-down list on the Transaction tab in the Database Profile Setup dialog box.

- **PocketBuilder application script** Type the following in a PocketBuilder application script:

```
SQLCA.dbParm = "CursorScroll = 'Dynamic',
       CursorLock = 'OptValue'"
```

This example sets the number of rows in the keyset to 100. Assume that the entire result set has 1000 rows. When the cursor is opened, the driver saves keys for the first 100 rows of the result set. It then retrieves the next block of 100 keys until the entire result set is retrieved.

- **Database profile** Type `100` in the Scrolling Options box on the Transaction tab in the Database Profile Setup dialog box.
- **PocketBuilder application script** Type the following in a PocketBuilder application script:

```
SQLCA.dbParm = "CursorScroll = 100"
```

See also CursorLock

Date

Description When you update data in the DataWindow painter, PocketBuilder builds a SQL UPDATE statement in the background. The Date DBParm parameter determines how PocketBuilder specifies a date datatype when it builds the SQL UPDATE statement.

Applies to ODBC

Syntax The Database Profile Setup dialog box inserts special characters (quotes and backslashes) where needed, so you can specify just the date format.

In a PocketBuilder application script, you must use the following syntax. PocketBuilder parses the backslash followed by two single quotes (' \") as a single quote when it builds the SQL UPDATE statement.

Date = ' \"*date_format*\" '

Parameter	Description
' \"	Type a single quote, followed by one space, followed by a backslash, followed by two single quotes. There is no space between the two single quotes and the beginning of the date format.

Parameter	Description
date_format	The date format you want PocketBuilder to use when it builds a SQL UPDATE statement to update a data source in the DataWindow painter. For more on display formats, see the *Users Guide*.
\"'	Type a backslash, followed by two single quotes, followed by one space, followed by a single quote. There is no space between the end of the date format and the backslash.

Default value

If no value is specified for the Date DBParm parameter, PocketBuilder looks for a date format in the section for your ODBC driver in the *pkodb25* initialization file. If no date format is found in the initialization file, PocketBuilder uses the ODBC date format escape sequence.

Examples

Assume you are updating a table named Employee by setting the Startdate column to 2007-04-23. This date is represented by the following date format:

```
yyyy-mm-dd
```

To specify that PocketBuilder should use this format for the date datatype when it builds the SQL UPDATE statement:

- **Database profile** Type the following in the Date Format box on the Syntax tab in the Database Profile Setup dialog box:

  ```
  yyyy-mm-dd
  ```

- **PocketBuilder application script** Type the following in a PocketBuilder application script:

  ```
  SQLCA.dbParm = "Date = ' \''yyyy-mm-dd\'' '"
  ```

What happens PocketBuilder builds the following SQL UPDATE statement to update the table:

```
UPDATE EMPLOYEE
SET STARTDATE = '2007-04-23'
```

See also

DateTime
Time

DateTime

Description

When you update data in the DataWindow painter, PocketBuilder builds a SQL UPDATE statement in the background. The DateTime DBParm parameter determines how PocketBuilder specifies a DateTime datatype when it builds the SQL UPDATE statement. (A DateTime datatype contains both a date value and a time value.)

Applies to

ODBC

Syntax

The syntax you use to specify the DateTime DBParm differs slightly depending on the database.

The Database Profile Setup dialog box inserts special characters (quotes and backslashes) where needed, so you can specify just the DateTime format.

In a PocketBuilder application script, you must use the following syntax. PocketBuilder parses the backslash followed by two single quotes (\") as a single quote when it builds the SQL UPDATE statement.

```
DateTime = ' \"DateTime_format\" '
```

Parameter	Description
'\ "	Type a single quote, followed by one space, followed by a backslash, followed by two single quotes. There is no space between the two single quotes and the beginning of the DateTime format.
DateTime_format	The DateTime format you want PocketBuilder to use when it builds a SQL UPDATE statement to update a data source in the painter. For more on display formats, see the *Users Guide*.
\" '	Type a backslash, followed by two single quotes, followed by one space, followed by a single quote. There is no space between the end of the date format and the backslash.

Default value

If no value is specified for the DateTime DBParm parameter, PocketBuilder looks for a DateTime format in the section for your ODBC driver in the *pkodb25* initialization file. If no DateTime format is found in the initialization file, PocketBuilder uses the ODBC DateTime format escape sequence.

Examples

Assume you are updating a table named Files by setting the Timestamp column to 4/2/07 3:45 pm. This DateTime is represented by the following DateTime format. To specify that PocketBuilder should use this format for the DateTime datatype when it builds the SQL UPDATE statement:

- **Database profile** Type the following in the DateTime Format box on the Syntax tab in the Database Profile Setup dialog box:

```
m/d/yy h:mm am/pm
```

- **PocketBuilder application script** Type the following in a PocketBuilder application script:

```
SQLCA.dbParm = "DateTime =
       ' \''m/d/yy h:mm am/pm\'' '"
```

What happens PocketBuilder builds the following SQL UPDATE statement to update the table:

```
UPDATE FILES
SET TIMESTAMP = '4/2/07 3:45 pm'
```

See also

Date
Time

DBGetTime

Description

Specifies the number of seconds PocketBuilder waits for a response from the DBMS when you retrieve rows in a DataWindow object or query. When you set the Async parameter to 1 to enable asynchronous operation, you can also set the DBGetTime parameter for those DBMSs that support this parameter.

If DBGetTime is set to 0 (the default), PocketBuilder waits indefinitely for a DBMS response (the request never times out). If the DBGetTime value expires before the first row is retrieved, your request is automatically canceled.

Applies to

ODBC

Syntax

DBGetTime = *value*

Parameter	Description
value	The number of seconds PocketBuilder waits for a DBMS response while waiting to retrieve the first row of a DataWindow object or query

Default value

DBGetTime = 0

Usage

Requirements for using DBGetTime To use the DBGetTime parameter, you must do *both* of the following:

- Set the Async parameter to 1 to enable asynchronous operation, as shown in the Examples
- Code a RetrieveRow event for a DataWindow object

Examples

To enable asynchronous operation and set the DBGetTime parameter to 20 seconds:

- **Database profile** Select the Asynchronous check box and type 20 in the Number Of Seconds To Wait box on the Transaction tab in the Database Profile Setup dialog box.
- **PocketBuilder application script** Type the following in a PocketBuilder application script:

```
SQLCA.dbParm = "Async = 1, DBGetTime = 20"
```

See also

Async

DBTextLimit

Description

Specifies the maximum length of a long VarChar column that can be returned when you include the column in a SQL SELECT statement.

Applies to

UL9 UltraLite
UL10 UltraLite 10

Syntax

DBTextLimit = '*value*'

Parameter	Description
value	The maximum length in bytes of a long VarChar column that UltraLite returns when you include the column in a SQL SELECT statement. The range of valid values is from 0 bytes to 65,536 bytes. If you set DBTextLimit to 0 or to a value that exceeds the maximum value, UltraLite returns the maximum length column.

Default value

DBTextLimit='32767'

Usage

You can set the DBTextLimit parameter if you want to include a long VarChar column in a DataWindow object without treating it as a binary large object (blob) and using SelectBlob and UpdateBlob. Setting DBTextLimit preallocates a buffer for retrieving the data.

Examples

To have DB-Library or CT-Library return a long VarChar column that is up to 32,000 bytes long when you include the column in a SQL SELECT statement:

- **Database profile** Type `32000` in the DBTextLimit box on the Syntax tab in the Database Profile Setup dialog box.

- **PocketBuilder application script** Type the following in a PocketBuilder application script:

```
SQLCA.dbParm = "DBTextLimit='32000'"
```

DecimalSeparator

Description

Specifies the decimal separator setting used by the back-end DBMS that you are accessing in PocketBuilder. If your DBMS uses a decimal separator other than a period (.), which is the default, set DecimalSeparator to the value for your DBMS to ensure that PocketBuilder correctly handles numeric strings returned from your database.

Applies to

ODBC

Syntax

DecimalSeparator = '*value*'

Parameter	Description
value	The decimal separator setting used by the back-end DBMS that you are accessing in PocketBuilder. Values are: • **'.'** (Default) Specifies that your back-end DBMS uses a period (.) as the decimal separator. If you do not specify DecimalSeparator or if you specify a value other than period (.) or comma (,), PocketBuilder uses period (.) as the decimal separator. • **','** Specifies that your back-end DBMS uses a comma (,) as the decimal separator.

Default value

DecimalSeparator = '.'

Usage

When to set DecimalSeparator The DecimalSeparator DBParm currently supports a period (.) and a comma (,) as valid values. Therefore, if the decimal separator setting for your DBMS is a comma, you should set the DecimalSeparator DBParm to `','` (comma) to make sure PocketBuilder correctly handles numeric strings returned from your database.

Examples

To specify that your DBMS uses a comma (,) as the decimal separator setting:

- **Database profile** Type a comma (,) in the Decimal Separator box on the Syntax tab in the Database Profile Setup dialog box.
- **PocketBuilder application script** Type the following in a PocketBuilder application script:

```
SQLCA.dbParm = "DecimalSeparator = ','"
```

See also

NumericFormat

DelimitIdentifier

Description

Specifies whether you want PocketBuilder to enclose the names of tables, columns, indexes, and constraints in double quotes when it generates SQL statements. This affects the behavior of any PocketBuilder painter that generates SQL syntax.

When to specify DelimitIdentifier
You must specify the DelimitIdentifier DBParm parameter *before* connecting to the database in PocketBuilder.

Applies to

ODBC
UL9 UltraLite 9
UL10 UltraLite 10

Syntax

DelimitIdentifier = '*value*'

Parameter	Description
value	Specifies whether you want PocketBuilder to enclose table and column names in double quotes. Values are: • **Yes** Use double quotes. • **No** Do not use double quotes.

Default value

For ODBC, the default value for the DelimitIdentifier parameter depends on the DelimitIdentifer setting in the *pkodb25* initialization file, which is 'YES' by default. For UltraLite, DelimitIdentifier='NO'.

Usage

For ODBC, the DelimitIdentifier DBParm setting overrides the DelimitIdentifier setting specified for your ODBC driver in the *pkodb25* initialization file.

Examples

To specify that PocketBuilder should not enclose table and column names in double quotes when it generates SQL statements:

- **Database profile** Clear the Enclose Table And Column Names In Quotes check box on the Syntax tab in the Database Profile Setup dialog box.
- **PocketBuilder application script** Type the following in a PocketBuilder application script:

```
SQLCA.dbParm = "Delimitidentifier='No'"
```

DisableBind

Description

For those DBMSs that support bind variables, PocketBuilder binds input parameters to a compiled SQL statement by default. The DisableBind parameter allows you to specify whether you want to disable this default binding. When you set DisableBind to 1 to disable the binding, PocketBuilder replaces the input variable with the value entered by the application user or specified in a PocketBuilder script.

Applies to

ODBC

Syntax

DisableBind = *value*

Parameter	Description
value	Specifies whether you want to disable the default binding of input parameters to a compiled SQL statement. Values are: • **0** (Default) PocketBuilder binds input parameters to a compiled SQL statement. • **1** PocketBuilder does *not* bind input parameters to a compiled SQL statement.

Default value

DisableBind = 0

Usage

Bind variables In a SQL statement, a bind variable is a placeholder for a column value. By default, PocketBuilder associates (binds) data from a variable defined in your application to the bind variable each time the SQL statement executes.

Using bind variables in SQL statements For example, the following SQL statement retrieves those rows in the Books table about books that are written by Hemingway:

```
SELECT * FROM books WHERE author = "Hemingway"
```

Suppose that you want to execute this statement to get information about books written by other authors. Instead of compiling and executing a new statement for each author, you can define a bind variable that represents the author's name. The user then supplies the author's actual name when the application executes. By using bind variables, you ensure that the statement is compiled only once and executed repeatedly with new values supplied by the user.

If your database supports bind variables and DisableBind is set to 0 (the default) to enable binding, PocketBuilder generates the statement with parameter markers (:bind_param) and passes the actual parameter value at execution time. For example:

```
SELECT * FROM books WHERE author = :bind_param
```

Bind variables and cached statements Using bind variables in conjunction with cached statements can improve the performance of most applications, depending on the application. In general, applications that perform a large amount of transaction processing benefit the most from using bind variables and cached statements.

In order to use cached statements, make sure that DisableBind is set to 0. This enables the binding of input variables to SQL statements in PocketBuilder. (For more about using cached statements, see the description of the SQLCache parameter.)

Bind variables improve performance by allowing PocketBuilder to insert and modify strings that exceed 255 characters.

Bind variables and default column values When DisableBind is set to 0 to enable the use of bind variables, the DataWindow painter does both of the following to get maximum performance improvement from using bind variables when you add rows to a DataWindow object:

- Generates a SQL INSERT statement that includes all columns (except identity and SQL Server timestamp)
- Reuses this SQL INSERT statement for each row you add to the DataWindow object

For example, if a table named Order_T contains three columns named Order_ID, Order_Date, and Customer_ID, the DataWindow painter generates the following SQL INSERT statement when DisableBind is set to 0 (default binding enabled):

```
INSERT INTO Order_T(Order_ID, Order_Date, Customer_ID)
       VALUES(:bind_param1, :bind_param2, :bind_param3)
```

If one of these columns is null, the DataWindow painter sets a null value indicator for this column parameter and executes the statement. This behavior is important to understand if you want your back-end DBMS to set a default value for any columns in your DataWindow object.

To illustrate, suppose that your application users do not enter a value for the Order_Date column because they expect the back-end DBMS to set this column to a default value of TODAY. Then, they retrieve the row and find that a null value has been set for Order_Date instead of its default value. This happens because the SQL INSERT statement generated by the DataWindow painter specified a null value indicator, so the DBMS set the column value to null instead of to its default value as expected.

Setting a default column value when binding is enabled If you are using bind variables (DisableBind set to 0) and want the back-end DBMS to set a column to its default value when your application user does not explicitly enter a value in a new row, you should set an initial value for the DataWindow object column that mirrors the DBMS default value for this column.

In the DataWindow painter, you can set or modify a column's initial value in the Column Specifications dialog box.

For more about the Column Specifications dialog box, see the *Users Guide*.

Setting a default column value when binding is disabled If you are *not* using bind variables (DisableBind set to 1) and want the back-end DBMS to set a column to its default value when your application user does not explicitly enter a value in a new row, you do *not* need to set an initial value for the DataWindow column.

This is because with bind variables disabled, the DataWindow painter generates a SQL INSERT statement for each row added to the DataWindow. If a column does not contain an explicit value, it is not included in the SQL INSERT statement.

Using the Order_T table example, if your application user enters `123` as the value for the Order_ID column and `A-123` as the value for the Customer_ID column, the DataWindow painter generates the following SQL INSERT statement when DisableBind is set to 1 (binding disabled):

```
INSERT INTO Order_T(Order_ID, Customer_ID)
       VALUES(123, 'A-123')
```

Your back-end DBMS would then set the Order_Date column to its default value as expected, since a value for Order_Date is not explicitly set in the SQL INSERT statement generated by the DataWindow painter.

Examples

To specify that PocketBuilder should disable the binding of input parameters to a compiled SQL statement:

- **Database profile** Select the Disable Bind check box on the Transaction tab in the Database Profile Setup dialog box.
- **PocketBuilder application script** Type the following in a PocketBuilder application script:

```
SQLCA.dbParm = "DisableBind = 1"
```

See also

SQLCache

FormatArgsAsExp

Description

Controls whether PocketBuilder converts a DataWindow retrieval argument of decimal datatype to scientific (exponential) notation if the argument exceeds 12 digits but has fewer than 16 digits. If FormatArgsAsExp is set to Yes, PocketBuilder performs this conversion.

When to specify FormatArgsAsExp
You must specify a value for FormatArgsAsExp *before* connecting to the database in PocketBuilder.

Applies to

ODBC
UL9 UltraLite 9
UL10 UltraLite 10

Syntax

FormatArgsAsExp= '*value*'

Parameter	Description
value	Specifies whether you want PocketBuilder to convert a DataWindow retrieval argument of decimal datatype to scientific (exponential) notation if the argument exceeds 12 digits but has fewer than 16 digits. Values are: • **Yes** PocketBuilder converts a retrieval argument of decimal datatype to scientific notation if it exceeds 12 digits but has fewer than 16 digits. • **No** (Default) PocketBuilder leaves the retrieval argument as a decimal and does not perform the default conversion to scientific notation if it exceeds 12 digits but has fewer than 16 digits.

Default value

FormatArgsAsExp = 'No'

Usage

When to use The setting of FormatArgsAsExp might affect the speed of data retrieval in your DataWindow objects, especially if you are accessing large databases.

If FormatArgsAsExp is set to Yes, some DBMS optimizers might interpret the resulting scientific notation as a different datatype and scan all rows in the table to find it. This can slow data retrieval if, for example, you are accessing a DB2 database with many large tables.

Setting FormatArgsAsExp to No causes PocketBuilder to leave the retrieval argument as a decimal. This speeds data retrieval for large databases.

Retrieval argument size limited
The FormatArgsAsExp DBParm is relevant only if a retrieval argument of type decimal has fewer than 16 digits.

Examples

To tell PocketBuilder to convert a retrieval argument with more than 12 digits but fewer than 16 digits to scientific notation:

- **Database profile** In the Database Profile Setup dialog box, check the Format Arguments in Scientific Notation check box on the Syntax tab.
- **PocketBuilder application script** Type the following in a PocketBuilder script:

```
SQLCA.DBParm = "FormatArgsAsExp='Yes'"
```

IdentifierQuoteChar

Description

Specifies the single quote character you want PocketBuilder to use to delimit the names of identifiers (tables, columns, indexes, and constraints) when it generates SQL statements. PocketBuilder uses the quote character you specify instead of the default quote character returned by your driver or data provider.

DelimitIdentifier must be set to Yes
For IdentifierQuoteChar to take effect, the DelimitIdentifier parameter must be set to Yes. Otherwise, PocketBuilder's default behavior is *not* to delimit identifiers in SQL statements and to ignore any value specified for IdentifierQuoteChar.

Applies to

ODBC

Syntax

IdentifierQuoteChar = '*quote_character*'

Parameter	Description
quote_character	The single character you want PocketBuilder to use instead of your driver's or data provider's default quote character to delimit the names of identifiers in SQL statements

Default value

None

PocketBuilder searches the following in this order to determine the IdentifierQuoteChar value:

1 The section for your database profile in the PocketBuilder initialization file (in the development environment) or the value of the Transaction object DBParm property (in a PocketBuilder application).

2 The section for your ODBC driver in the *pkodb25* initialization file.

If PocketBuilder does not find an IdentifierQuoteChar value in these locations, it makes a SQLGetInfo call to your driver to return the default SQL_IDENTIFIER_QUOTE_CHAR value.

Usage

By default, some drivers return quote characters that do not work with PocketBuilder's parsing routines, such as the backquote character (`). As a result, delimiting is turned off for these drivers in PocketBuilder.

However, if you paint SQL statements containing identifiers that require delimiters, syntax errors can occur if you are using a driver for which delimiting is turned off. To avoid such errors, set IdentifierQuoteChar to override the driver's default quote character.

Examples

To specify *c* as the quote character you want PocketBuilder to use to delimit identifiers in SQL statements:

- **Database profile** Type `c` in the Identifier Quote Character box on the Syntax tab in the Database Profile Setup dialog box.
- **PocketBuilder application script** Type the following in a PocketBuilder application script:

```
SQLCA.dbParm = "IdentifierQuoteChar = 'c'"
```

See also

DelimitIdentifier

LoginTimeOut

Description

Specifies the number of seconds the ODBC driver should wait for a login request to an ODBC data source.

Applies to

ODBC

Syntax

LoginTimeOut = *value*

Parameter	Description
value	The number of seconds you want the driver to wait for a login request

Default value

LoginTimeOut = 15

Usage

If you set LoginTimeOut to 0, PocketBuilder does not call the ODBC driver to set the LoginTimeOut value and instead waits the number of seconds specified by the ODBC driver's client software. If you set LoginTimeOut to a value greater than 0, PocketBuilder does call the ODBC driver to set the LoginTimeOut value.

Examples

To set the LoginTimeOut value to wait 60 seconds for a login request:

- **Database profile** Type `60` in the Login Timeout box on the Network tab in the Database Profile Setup dialog box.
- **PocketBuilder application script** Type the following in a PocketBuilder application script:

```
SQLCA.dbParm = "LoginTimeOut = 60"
```

MsgTerse

Description

Specifies whether PocketBuilder should display terse error messages for ODBC drivers. A terse error message is one without the SQLSTATE = *nnnn* prefix, where *nnnn* is the number of the error message.

By default, PocketBuilder displays ODBC error messages with the SQLSTATE prefix. To display error messages without the SQLSTATE prefix, set MsgTerse to Yes.

Applies to

ODBC

Syntax

MsgTerse = '*value*'

Parameter	Description
value	Specifies whether PocketBuilder should display error messages without the SQLSTATE prefix. Values are: • **Yes** Display error messages *without* the SQLSTATE prefix. • **No** (Default) Display error messages *with* the SQLSTATE prefix.

Default value

MsgTerse = 'No'

Usage

You can set the MsgTerse DBParm parameter to Yes to display shorter ODBC error messages in PocketBuilder. This may be useful if space on your screen is limited.

Examples

To specify that PocketBuilder should display terse error messages without the SQLSTATE prefix:

- **Database profile** Select the Display Terse Error Messages check box on the System tab in the Database Profile Setup dialog box.
- **PocketBuilder application script** Type the following in a PocketBuilder application script:

```
SQLCA.dbParm = "MsgTerse = 'Yes'"
```

NumericFormat

Description

If supported by the DBMS or back-end database, setting NumericFormat tells the driver to do special formatting of numeric strings in SQL syntax. This formatting affects how PocketBuilder generates numeric values in the SQL syntax it internally builds in DataWindow objects and reports and sends to your database.

Applies to

ODBC

Syntax

In the PocketBuilder development environment, the Database Profile Setup dialog box inserts special characters (quotes) where needed, so you can specify just the NumericFormat value (%s in this example).

In a PocketBuilder application script, you must use the following syntax if you are accessing an Oracle database through the ODBC interface. Note the use of *three single quotes* at the beginning and end of the string:

```
NumericFormat = '''%s,%s'''
```

In a PocketBuilder application script, you must use the following syntax if you are accessing an IBM DB2 database through the ODBC interface. Note the use of *one single quote* at the beginning and end of the string:

```
NumericFormat = '%s,%s'
```

Parameter	Description
'	Type three single open quotes. PocketBuilder parses the second and third quotes as one single open quote in the SQL syntax it builds and sends to the database.
%s	Represents one or more digits to the *left of the decimal* in the numeric string. PocketBuilder substitutes this value with the digits to the left of the decimal when it builds the SQL syntax.
,	Represents the decimal separator character (in this case a comma).

Parameter	Description
%s	Represents one or more digits to the *right of the decimal* in the numeric string. PocketBuilder substitutes this value with the digits to the right of the decimal when it builds the SQL syntax.
'	Type three single closed quotes. PocketBuilder parses the first and second quotes as one single closed quote in the SQL syntax it builds and sends to the database.

Default value

None

Usage

When to set NumericFormat In general, you should *not* need to set the NumericFormat DBParm parameter. Most back-end DBMSs do not require that the driver do special formatting of numeric strings in SQL syntax. However, some databases may require special formatting, such as an IBM DB2/MVS database server configured to use a comma as the decimal separator.

In these cases, setting NumericFormat allows you to generate numeric values with special formatting in the SQL syntax that PocketBuilder builds in DataWindow objects and reports and sends to your database. For example, if the decimal separator for your DBMS is a comma, you might want to set NumericFormat as shown in the Examples section below to use a comma as the decimal delimiter in the SQL syntax sent to your database.

Examples

This example shows how to specify that you want PocketBuilder to generate two numeric values in the format '125,50' and '4,0'. PocketBuilder uses the comma as a decimal separator in the SQL syntax it builds in DataWindow objects and reports and sends to an Oracle database.

- **Database profile** Type the following in the Numeric Format box on the Syntax tab in the Database Profile Setup dialog box:

```
%s,%s
```

- **PocketBuilder application script** Type the following in a PocketBuilder application script:

```
SQLCA.dbParm = "NumericFormat = '''%s,%s'''"
```

What happens PocketBuilder internally builds the following SQL INSERT statement in the DataWindow object and sends the syntax to your database. PocketBuilder returns single quotes in the SQL syntax.

```
INSERT INTO MYTABLE (a, b)
VALUES ('125,50', '4,0')
```

See also

DecimalSeparator

OJSyntax

Description

Specifies how PocketBuilder formats the SQL syntax for outer joins for the database back end you are accessing.

Applies to

ODBC

Syntax

OJSyntax = *value*

Parameter	Description
value	Specifies how you want SQL syntax to be formatted. Values are: • **ANSI_Escape** Apply ANSI standards and enclose the outer joins in escape notation { oj ... } that is parsed by the driver and replaced with DBMS-specific grammar. • **ANSI** Apply ANSI standards. • **PB** Maintain rules that applied to PowerBuilder 7.

Default value

OJSyntax = ANSI_ESCAPE

Usage

PocketBuilder provides support for ANSI SQL-92 outer join SQL syntax generation. It supports both left and right outer joins in graphics mode and full outer and inner joins in syntax mode.

You must set the OJSyntax DBParm to indicate the version of outer join SQL syntax you want PocketBuilder to generate. The default is ANSI_Escape, and the default can be reset to ANSI or PB (native).

OJSyntax is a dynamic DBParm in all database drivers that support it. It can therefore be changed at any time during the life of a database connection with a PowerScript statement such as:

```
SQLCA.DBParm="OJSyntax='ANSI_ESCAPE' "
```

Define outer joins in the SQL painter for portability When you define an outer join SELECT statement graphically in the SQL painter, the DataWindow object stores the SQL in pseudocode. At runtime, the outer join syntax is generated based on the current OJSyntax DBParm setting. This provides some degree of portability for DataWindow objects among multiple DMBSs.

When you define an outer join SELECT statement in syntax mode, the DataWindow object stores the SQL as syntax. This syntax is used without modification at runtime. The OJSyntax DBParm setting does *not* affect the SQL.

Using native outer join syntax The option PB generates native outer join syntax. It is available for ODBC only if PBOuterJoin and PBOuterJoinOperator syntax entries are set in the appropriate SYNTAX section for your DBMS in the *Sybase\PocketBuilder 2.5\pkodb25.ini* file.

When you migrate applications from PowerBuilder 7 and earlier versions of PowerBuilder, using ANSI outer join syntax might produce errors, depending on how the joins were defined in the painter. If a table is joined to multiple other tables with right outer joins, a valid ANSI outer join statement cannot be generated.

For more information about outer joins, see the section on using ANSI outer joins in the *Users Guide*.

Examples

To set the value of OJSyntax:

- **Database profile** Select the appropriate value from the Outer Join Syntax drop-down list on the Syntax tab in the Database Profile Setup dialog box.
- **PocketBuilder application script** Type the following in a PocketBuilder application script:

```
SQLCA.dbParm = "OJSyntax = 'ANSI'"
```

PacketSize

Description

Specifies the network packet size in bytes when you access an ODBC data source in PocketBuilder.

Many back-end DBMSs either do not support the PacketSize option or can return only the current network packet size. For information about whether the DBMS you are accessing supports PacketSize, see your DBMS documentation.

When to specify PacketSize
If your back-end DBMS supports it, you must specify the PacketSize DBParm parameter *before* connecting to the database in PocketBuilder.

Applies to

ODBC

Syntax

PacketSize = *value*

Parameter	Description
value	A 32-bit integer value that specifies the network packet size in bytes

Default value

The default value for PacketSize is the default for your back-end DBMS.

Usage

If the PacketSize value you specify is larger than the maximum network packet size or smaller than the minimum network packet size, your ODBC driver substitutes the maximum or minimum value for the value you specified.

Examples

To set the network packet size for an ODBC data source to 2048 bytes:

- **Database profile** Type the following in the Packet Size box on the Network tab in the Database Profile Setup dialog box:

```
2048
```

- **PocketBuilder application script** Type the following in a PocketBuilder application script:

```
SQLCA.dbParm = "PacketSize = 2048"
```

PBCatalogOwner

Description

Specifies a nondefault owner for the five extended attribute system tables. These tables contain default extended attribute information for your database.

When you specify a PBCatalogOwner name that is different from the default owner for your DBMS, PocketBuilder creates a new set of tables with the owner name you specify.

When to specify PBCatalogOwner
You must specify the PBCatalogOwner DBParm parameter *before* connecting to the database in PocketBuilder.

Applies to

ODBC

Syntax

PBCatalogOwner = '*owner_name*'

Parameter	Description
owner_name	Specifies the owner of the extended attribute system tables

Default value

If a value for PBCatalogOwner is not specified in the database profile or in the registry, the default value is the user ID specified in the database profile.

Usage

When to set When you specify a nondefault owner for the extended attribute system tables, you are in effect creating alternative tables. This is useful if you want to test new validation rules or display formats without overwriting the extended attributes currently in the default tables.

When you connect to an ODBC data source and a value for PBCatalogOwner is set in both the database profile and the *pkodb25* initialization file, the setting in the profile overrides the setting in the *pkodb25* initialization file.

This parameter cannot be set dynamically. The value set when the connection is made remains in effect until it is disconnected.

Examples

This example shows how to create a new set of extended attribute system tables with the owner TEST. The names of the new tables have the prefix TEST, such as TEST.pbcatcol, TEST.pbcatedt, and so on.

- **Database profile** Type the following in the PowerBuilder Catalog Table Owner box on the System tab in the Database Profile Setup dialog box:

```
TEST
```

- **PocketBuilder application script** Type the following in a PocketBuilder application script:

```
SQLCA.dbParm = "PBCatalogOwner = 'TEST'"
```

PBUseProcOwner

Description

When you access a database through the ODBC interface and define a DataWindow object that uses a stored procedure as its data source, PBUseProcOwner specifies whether PocketBuilder should qualify the stored procedure with the owner name in the SQL EXECUTE statement passed to the driver.

PocketBuilder qualifies the stored procedure with an owner only if the owner associated with the stored procedure is different from the ID of the current user (the developer building the DataWindow object or the user running the application containing the DataWindow object).

Applies to

ODBC

Syntax

PBUseProcOwner = '*value*'

Parameter	Description
value	Specifies whether PocketBuilder should qualify the stored procedure with its owner name in the SQL EXECUTE statement built by the DataWindow object and passed to the driver. Values are: • **Yes** If the owner associated with the stored procedure is different from the current user ID, PocketBuilder qualifies the stored procedure with its owner name in the SQL EXECUTE statement and passes this information to the driver. This allows users to execute stored procedures they do not own. For example: `EXECUTE FRAN.MYPROCEDURE` • **No** (Default) PocketBuilder does not qualify the stored procedure with its owner name in the SQL EXECUTE statement passed to the driver. For example: `EXECUTE MYPROCEDURE`

Default value

PBUseProcOwner = 'No'

Usage

PocketBuilder searches the following in this order to determine the PBUseProcOwner value:

1 The section for your database profile in the PocketBuilder initialization file in the development environment, or the value of the Transaction object DBParm property in a PocketBuilder application.

2 The section for your ODBC driver in the *pkodb25* initialization file.

If PocketBuilder does not find a PBUseProcOwner value in these locations, it defaults to a value of No.

DBA (database administrator) is a reserved word in SQL Anywhere syntax. If you define a DataWindow object with a SQL Anywhere stored procedure as its data source and DBA owns the stored procedure, the painter passes the following SQL EXECUTE statement to the ODBC driver if PBUseProcOwner is set to Yes:

```
EXECUTE DBA.MYPROCEDURE
```

This statement generates a syntax error because it includes the DBA reserved word.

If DBA owns the SQL Anywhere stored procedure you are using, you can avoid this syntax error by setting PBUseProcOwner to No so that PocketBuilder does not qualify the stored procedure with DBA.

In some situations, however, you *must* qualify the stored procedure with the DBA owner. For example, the DBA might want to grant execute permission to another user ID. In this case, you can avoid errors by editing the SQL EXECUTE syntax to enclose DBA in quotes, like this:

```
EXECUTE "DBA".MYPROCEDURE
```

Examples

To specify that PocketBuilder should qualify the stored procedure with its owner name in the SQL EXECUTE statement:

- **Database profile** Select the Qualify Stored Procedures With Owner Name check box on the Transaction tab in the Database Profile Setup dialog box.
- **PocketBuilder application script** Type the following in a PocketBuilder application script:

```
SQLCA.dbParm="PBUseProcOwner='Yes'"
```

RPCRebind

Description

Specifies whether you want PocketBuilder to rebind Remote Procedure Call (RPC) parameters.

When to specify RPCRebind
If your back-end DBMS supports the RPCRebind DBParm parameter, you must specify it *before* connecting to the database in PocketBuilder.

Applies to

ODBC

Syntax

RPCRebind = *value*

Parameter	Description
value	Specifies whether you want PocketBuilder to rebind RPC parameters. Values are: • **0** (Default) Use the bound variable to determine all required binding information. • **1** Rebind the parameters and use the parameter information returned from the database to bind the parameter.

Default value

RPCRebind = 0

Usage

For those DBMSs that support RPC calls, PocketBuilder binds the parameters for the call based on the size of the variables bound to the parameters.

Some drivers require rebinding of the parameters so that the parameter size (as returned from the back-end database) is used instead of the variable size. Failure to use the parameter size might result in an error or truncation for string parameters. However, some drivers always expect the binding to reflect the variable size. The RPCRebind parameter allows you to specify whether you want to rebind the parameters when RPCs are executed.

Examples

To specify that PocketBuilder should rebind RPC parameters:

- **Database profile** Select the RPC Rebind check box on the Transaction tab in the Database Profile Setup dialog box.
- **PocketBuilder application script** Type the following in a PocketBuilder application script:

```
SQLCA.dbParm = "RPCRebind = 1"
```

SQLCache

Description

Specifies the number of SQL statements PocketBuilder should cache. The default is 0, specifying an empty SQL cache.

PocketBuilder caches:

- SQL statements generated by a DataWindow object
- Embedded SQL statements

Applies to

ODBC

Syntax

The syntax for setting SQLCache depends on the database interface you are using.

Use the following syntax to set the SQLCache parameter:

SQLCache = *value*

Parameter	Description
value	The number of cursors you want to open in a script, plus the number of DataWindow-generated SELECT statements with retrieval arguments (default = 0).

Default value

SQLCache = 0

Usage

Maintaining statements in the cache Statements in the SQL cache are maintained on a least-recently-used (LRU) basis. In other words, if a statement must be removed from the cache to make room for another statement, PocketBuilder removes the statement that was least recently executed.

SQLCache and bind variables Caching SQL statements that you frequently execute improves their performance. Statements with bind variables are generally the most frequently used. In fact, if your DBMS does not support bind variables, caching statements is of limited value.

Setting DisableBind to use cached statements In order to use cached statements, make sure the DisableBind DBParm parameter is set to 0 (the default). This enables the binding of input variables to SQL statements.

For more about using bind variables, see DisableBind.

What happens The first time you execute a SQL statement containing bind variables, PocketBuilder does the following in this sequence:

1 Parses the statement.

2 For SQL SELECT statements, calls the appropriate database function to get a description of the result set.

3 Allocates memory buffers for the bind variables.

4 Binds the allocated memory buffers to the parsed statement.

When you cache this SQL statement, PocketBuilder stores the parsed statement, result set description, and memory buffer allocation and binding in the SQL cache. The next time you execute this statement, PocketBuilder finds it in the cache and avoids the overhead of repeating these steps.

If PocketBuilder finds an exact match for this statement in the SQL cache, it simply copies the new values supplied for the bind variables to the preallocated memory buffers and executes the statement. This is much faster than having to process the statement from scratch.

Determining the size of your SQL cache To determine an appropriate size for your SQL cache, you can check the value of the SQLReturnData property of the Transaction object.

When you disconnect from the database, the number of hits, misses, and entries in the SQL cache is stored in SQLReturnData as follows:

- Hits – the number of times PocketBuilder found a matching statement in the SQL cache
- Misses – the number of times PocketBuilder did not find a matching statement in the cache
- Entries – the total number of statements in the SQL cache, which is determined by your SQLCache setting

Examples

To set the SQL cache size to 25 statements:

- **Database profile** Type `25` in the Number Of SQL Statements Cached box on the Transaction tab in the Database Profile Setup dialog box.
- **PocketBuilder application script** Type the following in a PocketBuilder application script:

```
SQLCA.dbParm = "SQLCache = 25"
```

See also

DisableBind

StaticBind

Description

Specifies whether PocketBuilder gets a result set description to validate the SELECT statement against the database server before retrieving the data.

Applies to

ODBC
UL9 UltraLite 9
UL10 UltraLite 10

Syntax

StaticBind = *value*

Parameter	Description
value	Specifies whether you want PocketBuilder to get a result set description before retrieving data from a database into a DataWindow object. Values are: • **0** Get a result set description before retrieving data. You can also specify `'No'` to set this value. • **1** Skip getting a result set description before retrieving data. You can also specify `'Yes'` to set this value.

Default value

StaticBind = 1 (ODBC), StaticBind=0 (UltraLite)

Usage

ODBC Before it retrieves data from a database into a DataWindow object, PocketBuilder does not get a result set description to validate the SELECT statement against the database server. This default behavior generally speeds up the retrieval, especially when you are accessing the database over a network. (This feature is called **describeless retrieval**.)

If you want to override the default behavior and have PocketBuilder get a description of the result set before retrieving data, set the StaticBind DBParm parameter to 0 or No.

UltraLite There is currently no advantage to setting StaticBind to 1 for UltraLite connections, therefore it is set to 0 and disabled in this release.

Validation When StaticBind is set to 1 (the default), PocketBuilder assumes that the result set matches the column format of the DataWindow object into which it is being retrieved. If a mismatch occurs, PocketBuilder displays an error.

Troubleshooting tips Problems can occur in your application if the result set description obtained by the DataWindow object is different from the current database description of the result set. This can be due to the following reasons:

- The database definition changes after you build the DataWindow object
- You build the DataWindow object while connected to one DBMS and then run it against a different DBMS

To fix problems caused by conflicting result set descriptions, you can correct your DataWindow object definition by doing either of the following:

- Export and edit your column definitions
- Force a recompile of the SQL statement in the Database painter's Interactive SQL (ISQL) view (see the *Users Guide* for instructions)

If your DataWindow object and DBMS result set descriptions do not match and you want to avoid errors, set StaticBind to 0 or No to specify that PocketBuilder should *always* get a result set description before retrieving data into a DataWindow object.

Examples

To specify that you want PocketBuilder to get a result set description before retrieving data into a DataWindow object:

- **Database profile** Clear the Static Bind check box on the Transaction tab in the Database Profile Setup dialog box.
- **PocketBuilder application script** To specify this statement in a PocketBuilder application script, type the following:

```
SQLCA.dbParm = "StaticBind=0"
```

StripParmNames

Description

Specifies that explicitly named parameters should not be passed to the ODBC driver.

Applies to

ODBC

Syntax

StripParmNames = *'value'*

Parameter	Description
value	Specifies that explicitly named parameters should not be passed to the ODBC driver. Values are: • **Yes** Remove all parameter names from the generated call escape syntax. • **No** (Default) Keep parameter names that are explicitly specified and include them in the generated call escape syntax.

Default value

StripParmNames = 'No'

Usage

By default, PocketBuilder retains parameter names if they are explicitly specified in the execution of a stored procedure. As a result, syntax such as the following might be generated and sent to the ODBC driver:

```
{call proc(a=?,b=?)}
```

Some database vendors do not allow parameter names to be specified in the generated call escape syntax. To prevent the passing of explicitly named parameters to the ODBC driver, set StripParmNames to Yes. This means that the parameters are passed in the order specified.

Examples

To strip explicitly stated parameter names from a stored procedure:

- **Database profile** Select the Strip Parameter Names check box on the Syntax tab in the Database Profile Setup dialog box.
- **PocketBuilder application script** Type the following in a PocketBuilder application script:

```
SQLCA.dbParm = "StripParmNames = 'Yes'"
```

TableCriteria

Description

Lets you specify search conditions to limit the list of tables and views that displays in the Installed Database Interfaces Tables list in PocketBuilder. Setting this parameter can be useful if you are working with a very large database in the PocketBuilder development environment.

When to specify TableCriteria
You must specify the TableCriteria DBParm parameter *before* connecting to the database in PocketBuilder. The TableCriteria DBParm parameter has no effect in a PocketBuilder application script.

Applies to

ODBC

Syntax

You specify the TableCriteria search conditions on the System tab in the Database Profile Setup dialog box.

Default value

None. If you do not specify any values, the TableCriteria parameter is not used.

Usage

To specify the TableCriteria search conditions, enter information in the following boxes:

Field	Description
Table Name	Specifies the names of tables to display in the current database. You can use wildcard characters.
Table Owner	Displays only those tables belonging to the specified table owner. You can use wildcard characters. If you omit this value, PocketBuilder displays all tables matching the table name that you have permission to access.
Include Tables	Specifies that tables should be displayed.
Include Views	Specifies that views should be displayed.
Include System Tables	Specifies that system tables should be displayed.
Include Aliases	Specifies that alias tables should be displayed.
Include Synonyms	Specifies that synonym tables should be displayed.

Examples

Type `QADB%` in the Table Name box and `DWMC31` in the Table Owner box on the System tab in the Database Profile Setup dialog box to set the Table Criteria property to:

```
TableCriteria='QADB%,DWMC31'
```

Time

Description

When you update data in the DataWindow painter, PocketBuilder builds a SQL UPDATE statement in the background. The Time DBParm parameter determines how PocketBuilder specifies a time datatype when it builds the SQL UPDATE statement.

Applies to

ODBC

Syntax

The Database Profile Setup dialog box inserts special characters (quotes and backslashes) where needed, so you can specify just the time format.

In a PocketBuilder application script, you must use the following syntax. PocketBuilder parses the backslash followed by two single quotes (\") as a single quote when it builds the SQL UPDATE statement:

```
Time = ' \"time_format\" '
```

Parameter	Description
'\"	Type a single quote, followed by one space, followed by a backslash, followed by two single quotes. There is no space between the two single quotes and the beginning of the time format.
time_format	The time format you want PocketBuilder to use when it builds a SQL UPDATE statement to update a data source in the DataWindow painter. For more on display formats, see the *Users Guide*.
\"'	Type a backslash, followed by two single quotes, followed by one space, followed by a single quote. There is no space between the end of the time format and the backslash.

Default value

If no value is specified for the Time DBParm parameter, PocketBuilder looks for a time format in the section for your ODBC driver in the *pkodb25* initialization file. If it finds no time format in the *pkodb25* initialization file, PocketBuilder uses the ODBC time format escape sequence.

Examples

Assume you are updating a table named Workhours by setting the Start column to 08:30. This time is represented by the following PocketBuilder time format:

```
hh:mm
```

To specify that PocketBuilder should use this format for the time datatype when it builds the SQL UPDATE statement:

- **Database profile** Type the following in the Time Format box on the Syntax tab in the Database Profile Setup dialog box:

  ```
  hh:mm
  ```

- **PocketBuilder application script** Type the following in a PocketBuilder application script:

  ```
  SQLCA.dbParm = "Time = ' \''hh:mm\'' '"
  ```

What happens PocketBuilder builds the following SQL UPDATE statement to update the table:

```
UPDATE WORKHOURS
SET START = '08:30'
```

See also

Date
DateTime

TrimSpaces

Description

Specifies whether PocketBuilder should trim trailing spaces from data values retrieved from the following datatypes: Char, Char for Bit Data, VarChar, and VarChar for Bit Data.

Applies to

UL9 UltraLite
UL10 UltraLite 10

Syntax

TrimSpaces = *value*

Parameter	Description
value	Specifies whether PocketBuilder should trim trailing spaces from data of type Char, Char for Bit Data, and VarChar for Bit Data. Values are: • **0** Do not trim trailing spaces. • **1** (Default) Trim trailing spaces.

Default value

TrimSpaces = 1

Usage

By default, PocketBuilder trims spaces from the following datatypes: Char, Char for Bit Data, VarChar, and VarChar for Bit Data.

If your DBMS makes a distinction between Char data with trailing spaces and Char data without trailing spaces when evaluating a WHERE clause expression, you may receive the message `Row changed between retrieve and update` when your DataWindow update properties are set to "Key and updateable columns". To prevent this, change your DataWindow update properties. In embedded SQL, you can check Sqlca.Sqlnrows after each update to determine if the update took place. Avoid using Char data columns in the WHERE clause of an UPDATE or DELETE statement when TrimSpaces=1.

Examples

To specify that PocketBuilder should not trim trailing spaces:

- **Database profile** Clear the Trim Spaces check box on the Syntax tab in the Database Profile Setup dialog box.
- **Application** Type the following in code:

```
SQLCA.DBParm = "TrimSpaces=0"
```

CHAPTER 2

Database Preferences

About this chapter

This chapter describes the syntax and use of each connection-related database preference that you can set in PocketBuilder.

Contents

The database preferences are listed in alphabetical order.

AutoCommit

Description

AutoCommit controls whether PocketBuilder issues SQL statements outside or inside the scope of a transaction.

When AutoCommit is set to false (the default), PocketBuilder issues SQL statements *inside* the scope of a transaction. When AutoCommit is set to true, PocketBuilder issues SQL statements *outside* the scope of a transaction.

When to specify AutoCommit

In the development environment, you must set AutoCommit before connecting to the database using ODBC. AutoCommit takes effect only when the database connection occurs. Changes made to AutoCommit after the connection occurs have no effect on the current connection.

When you connect to an UltraLite database in the development environment, all processing in painters takes place as if AutoCommit is set to true. The setting in the database profile applies when you run or debug an application.

In a PocketBuilder application script, you can reset the value of AutoCommit at any time. This lets you override the initial setting if necessary.

In a PocketBuilder application

You can set AutoCommit in a script as a property of the Transaction object. The following syntax assumes you are using the default Transaction object SQLCA, but you can also define your own Transaction object:

```
SQLCA.AutoCommit = value
```

Parameter	Description
value	Specifies whether PocketBuilder issues SQL statements outside or inside the scope of a transaction. Values are: • True – PocketBuilder issues SQL statements *outside the scope of a transaction*. The statements are not part of a logical unit of work (LUW). If a SQL statement is successful, the DBMS updates the database immediately as if a COMMIT statement had been issued. • False (Default) – PocketBuilder issues SQL statements *inside the scope of a transaction*. PocketBuilder issues a BEGIN TRANSACTION statement at the start of the connection and issues another BEGIN TRANSACTION statement after each COMMIT or ROLLBACK statement is issued.

In the development environment

Select or clear the AutoCommit Mode check box on the Connection tab in the Database Profile Setup dialog box, as follows:

- **Select the check box** Sets AutoCommit to true for this connection.
- **Clear the check box** (Default) Sets AutoCommit to false for this connection.

Default value

AutoCommit = false

Usage

Transactions A transaction is one or more SQL statements that form a logical unit of work (LUW). Within a transaction, all SQL statements must succeed or fail as one logical entity. Changes are made to the database only if all statements in the transaction succeed and a COMMIT is issued. If one or more statements fail, you must issue a ROLLBACK to undo the changes. This ensures the integrity and security of data in your database.

When you change the value of AutoCommit from false to true, PocketBuilder issues a COMMIT statement by default.

Caution
When you set AutoCommit to true, you cannot roll back database changes. Therefore, use care when changing the setting of AutoCommit.

Using EXECUTE IMMEDIATE When AutoCommit is set to true, you can use the EXECUTE IMMEDIATE dynamic SQL statement to issue BEGIN TRANSACTION, COMMIT TRANSACTION, ROLLBACK TRANSACTION, and other SQL statements to control your own transaction processing.

If you use the EXECUTE IMMEDIATE dynamic SQL statement to issue BEGIN TRANSACTION, you must use the EXECUTE IMMEDIATE dynamic SQL statement to issue a corresponding COMMIT TRANSACTION or ROLLBACK TRANSACTION.

For information about using the EXECUTE IMMEDIATE statement, see the *PowerScript Reference*.

Examples

To set AutoCommit to true and issue SQL statements outside the scope of a transaction:

- **Development environment** Select the AutoCommit Mode check box on the Connection tab in the Database Profile Setup dialog box.
- **PocketBuilder application script** Type the following in a script:

```
SQLCA.AutoCommit = true
```

If you specify AutoCommit Mode in your database profile, the PowerScript syntax displays on the Preview tab in the Database Profile Setup dialog box. You can copy the syntax from the Preview tab into your script.

Connect to Default Profile

Description

Connect to Default Profile controls whether the Database painter establishes a connection to a database using a default profile when the painter is invoked. If Connect to Default Profile is not selected, the Database painter opens without establishing a connection to a database.

In a PocketBuilder application

You *cannot* set the Connect to Default Profile database preference in a PocketBuilder application script.

In the development environment

In the Database painter, select or clear the Connect to Default Profile check box in the Database Preferences dialog box as follows:

- **Select the check box** (Default) The next time you invoke the Database painter, it automatically connects to the default database profile.
- **Clear the check box** The next time you invoke the Database painter, it does *not* automatically connect to the default database profile.

Default value

The Connect to Default Profile check box in the Database Preferences dialog box is selected by default.

Usage

Connect to Default Profile allows you to open the Database painter *without* establishing a connection to a database. Consequently, you can perform all database-related tasks, including defining a database profile and connecting to a database, in the Database painter. However, you might want to continue to define profiles and/or connect to a database using the Database Profile since opening the Database painter uses more system resources.

Keep Connection Open

Description

By default, PocketBuilder opens a database connection the first time you open a painter requiring a connection, and it maintains the connection until you exit.

When you connect to a database in the PocketBuilder development environment without using a database profile, you can set the Keep Connection Open database preference to specify when PocketBuilder should close the database connection. The setting of Keep Connection Open has no effect when you use a database profile to connect in PocketBuilder.

In a PocketBuilder application

You *cannot* set the Keep Connection Open database preference in a PocketBuilder application script.

In the development environment

In the Database painter, select or clear the Keep Connection Open check box in the Database Preferences dialog box as follows:

- **Select the check box** (Default) Stays connected to the database throughout your PocketBuilder session and closes the connection when you exit.
- **Clear the check box** Opens the database connection when a painter requires it and closes the connection when you close a painter or finish compiling a script.

Default value

The Keep Connection Open check box in the Database Preferences dialog box is selected by default.

Usage

Requirements for using Keep Connection Open To use the Keep Connection Open database preference, *both* of the following must be true:

- **Working in PocketBuilder development environment** You must be working in the PocketBuilder development environment.
- **Using default connection information** PocketBuilder must use the most-recently used connection information in the Windows registry to connect to the database. Keep Connection Open has no effect when you select a database profile to connect to the database.

What happens If you meet both of these requirements, clearing the Keep Connection Open check box opens a database connection only when you are working in a painter that requires a connection; it closes the connection at other times. This can save you money if you are accessing a database that charges for connect time.

Lock

Description

The Lock preference sets the isolation level to use when connecting to the database.

ODBC only
For UltraLite, only one application can connect to the database at a time. However, a single application can have up to four separate connections to the same database, and each connection can have a separate transaction running. UltraLite does not provide alternative locking behaviors and the lock value is set to Read Uncommitted in PocketBuilder.

In multiuser databases, transactions initiated by different users can overlap. If these transactions access common data in the database, they can overwrite each other or collide.

To prevent concurrent transactions from interfering with each other and compromising the integrity of your database, certain DBMSs allow you to set the isolation level when you connect to the database. Isolation levels are defined by your DBMS and specify the degree to which operations in one transaction are visible to operations in a concurrent transaction. Isolation levels determine how your DBMS isolates or locks data from other processes while it is being accessed.

PocketBuilder uses the Lock preference to allow you to set various database lock options. Each lock value corresponds to an isolation level defined by your DBMS.

When to specify the Lock value
You must set the Lock value *before* you connect to the database in the PocketBuilder development environment or in a PocketBuilder application. The Lock value takes effect only when the database connection occurs. Changes to the Lock value after the connection occurs have no effect on the current connection.

In a PocketBuilder application

You can set the Lock value in a script as a property of the Transaction object. The following syntax assumes you are using the default Transaction object, SQLCA, but you can also use a user-defined Transaction object:

SQLCA.Lock = "*value*"

where *value* is the lock value you want to set.

Lock values

The following table lists the lock values and corresponding isolation levels for ODBC. You set the lock value in a PocketBuilder application script, and the isolation level in a database profile.

Lock values	Isolation levels
RU	Read Uncommitted
RC	Read Committed
RR	Repeatable Read
TS	Serializable Transactions
TV	Transaction Versioning

For information about how SQL Anywhere handles isolation levels, see the chapter on using transactions and isolation levels in the *SQL Anywhere Server SQL Usage* book in the SQL Anywhere documentation.

In the development environment

Select the isolation level you want from the Isolation Level drop-down list on the Connection tab in the Database Profile Setup dialog box.

Default value

The default lock value depends on how your database is configured. For information, see your DBMS documentation.

Usage

The TV (Transaction Versioning) setting does *not* apply to SQL Anywhere databases.

Examples

Example 1 To set the Lock value to RC (Read Committed) for an SQL Anywhere database:

- **Development environment** Select Read Committed from the Isolation Level drop-down list in the Database Profile Setup dialog box.
- **PocketBuilder application script** Type the following in a script:

```
SQLCA.Lock = "RC"
```

If you specify Isolation Level in your database profile, the PowerScript syntax displays on the Preview tab in the Database Profile Setup dialog box. You can copy the syntax from the Preview tab into your script.

Read Only

Description

Read Only specifies whether PocketBuilder should update the extended attribute system tables and any other tables in your database. The extended attribute system tables (also known as the extended catalog) consist of five tables that contain default extended attribute information for your database.

The Read Only setting determines whether you can modify (update) the tables in your database. By default, the Read Only check box is cleared in the Database Preferences dialog box. This means that PocketBuilder updates the extended attribute system tables and other tables in your database when you make changes.

No extended attributes in UltraLite
UltraLite does not contain system tables or extended attributes.

If you select the Read Only check box, PocketBuilder *does not update* the extended attribute system tables or any other tables in your database. You *cannot* modify (update) information in the extended attribute system tables or any other database tables from the DataWindow painter when the Read Only check box is selected.

In a PocketBuilder application

You *cannot* set the Read Only database preference in a PocketBuilder application script.

In the development environment

In the PocketBuilder Database painter, select or clear the Read Only check box in the Database Preferences dialog box as follows:

- **Select the check box** PocketBuilder does not update the extended attribute system tables or any other tables in your database. You *cannot* modify (update) information in the extended attribute system tables or any other database tables from the DataWindow painter when the Read Only check box is selected.
- **Clear the check box** (Default) PocketBuilder updates the extended attribute system tables and any other tables in your database when you modify them.

Default value

The Read Only check box in the Database Preferences dialog box is cleared by default.

Usage

If you select the Read Only check box in the Database Preferences dialog box, you cannot modify information in *any* tables from the DataWindow painter.

Therefore, you can use only:

- SELECT and Retrieve statements in the DataWindow and Report painters
- SELECT statements in embedded SQL

See also

Use Extended Attributes

Shared Database Profiles

Description

Specifies the path name of the PocketBuilder initialization file containing the database profiles you want to share.

For instructions on sharing database profiles in the PocketBuilder development environment, see "Sharing database profiles" in the *Resource Guide*.

In a PocketBuilder application

You *cannot* set the Shared Database Profiles database preference in a PocketBuilder application script.

In the development environment

In the Database painter, supply the path name of the PocketBuilder initialization file containing shared profiles in the Shared Database Profiles box in the Database Preferences dialog box. You can type the path name or click the Browse button to display it.

Default value

The Shared Database Profiles box in the Database Preferences dialog box is blank (unspecified) by default.

Examples

To share database profiles contained in the file *I:\SHARE\PK.INI*, type or browse to the following in the Shared Database Profiles box in the Database Preferences dialog box:

```
I:\SHARE\PK.INI
```

SQL Terminator Character

Description

Specifies the SQL statement terminator character used by the Database painter's Interactive SQL (ISQL) view.

The default terminator character for the ISQL view is a semicolon (;). If a semicolon conflicts with the terminator character used by your DBMS syntax, you can change the painter's terminator character by specifying a different character in the SQL Terminator Character box in the Database Preferences dialog box. A good choice for a terminator character is the backquote or backprime (`) character.

Changing the terminator character is recommended when you are using the ISQL view to create or execute stored procedures, triggers, and SQL scripts.

In a PocketBuilder application

You *cannot* set the SQL Terminator Character database preference in a PocketBuilder application script.

In the development environment

In the Database Preferences dialog box in the Database painter, in the SQL Terminator Character box, type the terminator character you want to use.

Default value

The default SQL Terminator Character value in the Database Preferences dialog box is a semicolon (;).

Usage

The following are typical situations that might require you to change the default SQL Terminator Character value:

- **Creating stored procedures and triggers** If you are creating stored procedures and triggers in the ISQL view, change the painter's terminator character to one that you do not expect to use in the stored procedure or trigger syntax for your DBMS, such as the backquote (`) character.

  After you finish using the stored procedure, you can change the terminator character back to semicolon (;). If you prefer, you can continue to use the new terminator character as long as it does not conflict with any stored procedure or trigger syntax you plan to use.

- **Executing SQL scripts** If you plan to execute any SQL scripts in the ISQL view, make sure the terminator character used in the script agrees with the terminator character currently set in the view.

Examples

To change the SQL statement terminator character in the ISQL view to a backquote (`), type a backquote in the SQL Terminator Character box in the Database Preferences dialog box.

Use Extended Attributes

Description

Controls access to the extended attribute system tables by specifying whether you want PocketBuilder to create these tables. The extended attribute system tables (also known as the extended catalog) consist of five tables that contain default extended attribute information for your database.

No extended attributes in UltraLite
UltraLite does not contain system tables or extended attributes.

By default, the Use Extended Attributes check box is selected in the Database Preferences dialog box. This setting creates the extended attribute system tables the first time you connect to a database using PocketBuilder.

In a PocketBuilder application

You *cannot* set the Use Extended Attributes database preference in a PocketBuilder application script.

In the development environment

In the Database painter, select or clear the Use Extended Attributes check box in the Database Preferences dialog box as follows:

- **Select the check box** (Default) Creates the extended attribute system tables when you connect to the database for the first time.
- **Clear the check box** Does *not* create the extended attribute system tables if they do not exist. Instead, the DataWindow and Report painters use the appropriate default values for extended attributes (such as headers, labels, and text color). If the extended attribute system tables already exist, PocketBuilder does not use them when you create a new DataWindow object.

Default value

The Use Extended Attributes check box in the Database Preferences dialog box is selected by default.

Usage

If you clear the Use Extended Attributes check box in the Database Preferences dialog box, PocketBuilder *does not do* any of the following:

- Create the extended attribute system tables
- Insert, update, or delete rows in the extended attribute system tables
- Select information (such as header names) from the extended attribute system tables
- Execute statements that reference the extended attribute system tables

See also

Read Only

Index

E

F

I

K

L

M

N

O

P

R

S

T

U