Connection Reference

PowerBuilder® Classic

12.5.2

DOCUMENT ID: DC33820-01-1252-01

LAST REVISED: February 2013

Contents

About This Book

Audience

This book is for anyone who uses Sybase® PowerBuilder® to connect to a database. It assumes that you are familiar with the database you are using and have installed the server and client software required to access the data.

How to use this book

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

Related documents

For information about connecting to a database in the PowerBuilder development environment, see *Connecting to Your Database*.

For a complete list of PowerBuilder documentation, see the preface of *Getting Started*.

Other sources of information

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

- The Getting Started CD contains release bulletins and installation guides in PDF format. 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 Sybase Product Documentation Web site is accessible using a standard Web browser. In addition to product documentation, you will find links to EBFs/Maintenance, Technical Documents, Case Management, Solved Cases, newsgroups, and the Sybase Developer Network.

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

Conventions

The formatting conventions used in this manual are:

Formatting example	Indicates
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 *pblname*.`pbd` • File and path names
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 documentation or online help, please have the designated person contact Sybase Technical Support or the Sybase subsidiary in your area.

CHAPTER 1 Database Parameters

About this chapter

This chapter describes the syntax and use of each database parameter that you can set in PowerBuilder.

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

Database parameters and supported database interfaces

The following table lists each supported database interface and the database parameters you can use with that interface in PowerBuilder. The database parameters are described in alphabetical order after the table.

Database interface	DBParm parameters	
ADO.NET	ADORelease	Namespace
	BindSPInput	NCharBind (Oracle only)
	CommandTimeout	OJSyntax
	CommitOnDisconnect	PBCatalogOwner
	Database	PBMaxBlobSize
	DataLink	Provider
	DataSource	ProviderString
	DateFormat	RecheckRows
	DateTimeFormat	SPCache
	DBConfigSection	StaticBind
	DecimalSeparator	TimeFormat
	DefaultProcOwner	Timeout
	DelimitIdentifier	TrimSpaces
	DisableBind	TrustedConnection
	GenerateEqualsNull	
	IdentifierQuoteChar	
	Isolation	

Database interface	DBParm parameters
ASE Sybase Adaptive Server® Enterprise **Release** The Release database parameter must be set to 15 or higher to use the ASE interface.	AppName Async BinTxtBlob Block (DirectConnect and Adaptive Server Enterprise) CharSet CommitOnDisconnect CursorType CursorUpdate DateTimeAllowed DBGetTime DBTextLimit DelimitIdentifier DisableBind DS_Alias DS_Copy DS_DitBase DS_Failover DS_Password DS_Principal DS_Provider DS_TimeLimit FoDialog FormatArgsAsExp GenerateEqualsNull Host KeepAlive Language Locale Log MaxConnect OJSyntax PacketSize (ASE, DIR, SNC, SYC) PBCatalogOwner ProxyUserName PWDialog PWEncrypt PWExpDialog Release Sec_Channel_Bind Sec_Confidential Sec_Cred_Timeout Sec_Data_Integrity Sec_Data_Origin Sec_Delegation Sec_Keytab_File Sec_Mechanism Sec_Mutual_Auth Sec_Network_Auth Sec_Replay_Detection Sec_Seq_Detection Sec_Server_Principal Sec_Sess_Timeout ShowTempTables StaticBind SvrFailover SystemProcs TableCriteria TrimSpaces UnicharBind UTF8

Database interface	**DBParm parameters**	
DIR Sybase DirectConnect™	AppName Async Block (DirectConnect and Adaptive Server Enterprise) CharSet CommitOnDisconnect CursorUpdate DateTimeAllowed DBGetTime DBTextLimit DecimalSeparator DelimitIdentifier FormatArgsAsExp GenerateEqualsNull HostReqOwner Language	Locale LowerCaseIdent MaxConnect PacketSize (ASE, DIR, SNC, SYC) PBCatalogOwner Request ShowWarnings SQLQualifiers StaticBind SystemOwner TableCriteria TrimSpaces TRS UseProcSyntax UTF8
IN9 Informix	Async BinTxtBlob CommitOnDisconnect DateTimeAllowed DBGetTime DecimalSeparator DelimitIdentifier DisableBind	GenerateEqualsNull INET_DBPATH INET_PROTOCOL INET_SERVICE OJSyntax Scroll ThreadSafe TrimSpaces
I10 Informix	Async BinTxtBlob Client_Locale CommitOnDisconnect DateTimeAllowed DBGetTime Db_Locale DecimalSeparator DelimitIdentifier DisableBind EncryptionPass Hint	GenerateEqualsNull INET_DBPATH INET_PROTOCOL INET_SERVICE OJSyntax OnlineIndex PBCatalogOwner Scroll StmtCache StrByCharset ThreadSafe TrimSpaces

Database interface	DBParm parameters	
JDBC	Async	MsgTerse
	CacheName	NumericFormat
	CommitOnDisconnect	OJSyntax
	Date	PBCatalogOwner
	DateTime	Properties
	DBGetTime	ProxyUserName
	DelimitIdentifier	ReleaseConnectionOption
	DisableBind	StaticBind
	Driver	TableCriteria
	FormatArgsAsExp	Time
	GenerateEqualsNull	TraceFile
	GetConnectionOption	TrimSpaces
	IdentifierQuoteChar	URL
	LoginTimeOut	UseContextObject
	MapDateToDateTime	
ODBC **Using DBParms with ODBC** These database parameters are supported by the PowerBuilder ODBC interface only if *both* the ODBC driver you are using and the back-end DBMS support the feature	Async	InsertBlock
	Block (ODBC, OLE DB, Oracle, and SNC)	LoginTimeOut
	CacheName	MsgTerse
	CallEscape	NumericFormat
	CommitOnDisconnect	ODBCU_CONLIB
	ConnectOption	OJSyntax
	ConnectString	PacketSize (ODBC)
	CursorLib	PBCatalogOwner
	CursorLock	PBNewSPInvocation
	CursorScroll	PBTrimCharColumns
	Date	PBUseProcOwner
	DateTime	ProxyUserName
	DBGetTime	ReleaseConnectionOption
	DecimalSeparator	RPCRebind
	DefaultProcOwner	SQLCache
	DelimitIdentifier	StaticBind
	DelimitIdentifierToDB	StripParmNames
	DisableBind	TableCriteria
	FormatArgsAsExp	Time
	GenerateEqualsNull	TrimSpaces
	GetConnectionOption	UseContextObject
	IdentifierQuoteChar	

Database interface	DBParm parameters	
OLE DB	BinTxtBlob Block (ODBC, OLE DB, Oracle, and SNC) CacheAuthentication CommitOnDisconnect DataLink DataSource DateFormat DateTimeFormat DecimalSeparator DelimitIdentifier DelimitIdentifierToDB DisableBind EncryptPassword GenerateEqualsNull IdentifierQuoteChar ImpersonationLevel Init_Prompt IntegratedSecurity LCID	Location MaskPassword Mode OJSyntax PBCatalogOwner PBMaxBlobSize PBTrimCharColumns PersistEncrypted PersistSensitive ProtectionLevel Provider ProviderString RecheckRows ReturnCommandHandle ServiceComponents SPCache StaticBind TimeFormat Timeout TimeStamp

Database interface	DBParm parameters	
ORA Oracle 11*g* O10 Oracle 10*g* Oracle 9.2 or later databases	AppDriverName (ORA only) Async BindSPInput BinTxtBlob Block (ODBC, OLE DB, Oracle, and SNC) CacheName CnnPool (deprecated) CommitOnDisconnect ConnectAs CSIncr (ORA only) CSMax (ORA only) CSMin (ORA only) Date DateTime DBGetTime DecimalSeparator DelimitIdentifier DisableBind FoDelay FoDialog FoRetryCount FormatArgsAsExp GenerateEqualsNull GetConnectionOption HANotification (O10 and later) MaxFetchBuffer MixedCase NCharBind NCharLiteral (O10 and later)	NLS_Charset NumbersInternal ObjectMode OJSyntax OraMTSConFlgs PackageProcs PBCatalogOwner PBNoCatalog PoolCreator (ORA only) Pooling (ORA only) PoolPwd (ORA only) PWDialog PWExpDialog QualifyPublic ReleaseConnectionOption RTConnBalancing (ORA only) SPCache ServerName (ORA only) SessionHomogeneous (ORA only) StatementCache (O10 and later) StaticBind StrByCharset SvrFailover TableCriteria ThreadSafe Time TimeStamp TrimSpaces UseContextObject

Database interface	DBParm parameters	
SNC SQL Native Client for Microsoft SQL Server	AppName	Identity
	BindSPInput	NCharBind
	BindSPInput	OJSyntax
	BinTxtBlob	PacketSize (ASE, DIR, SNC, SYC)
	Block (ODBC, OLE DB, Oracle, and SNC)	PBCatalogOwner
		PBMaxBlobSize
	CommitOnDisconnect	PBMaxTextSize
	Database	Provider
	DataLink	ProviderString
	DateFormat	RecheckRows
	DBTextLimit	SPCache
	DecimalSeparator	StaticBind
	DelimitIdentifier	TimeFormat
	DisableBind	Timeout
	Encrypt	TimeStamp
	FailoverPartner	TrimSpaces
	GenerateEqualsNull	TrustedConnection
	HighSeverityError	TrustServerCertificate
	Host	

Database interface	DBParm parameters	
SYC Sybase Adaptive Server® Enterprise **Set Release** The Release database parameter must be set to the version of your Open Client® software (11 or higher) to use DS_* and Sec_* parameters.	AppName Async BinTxtBlob Block (DirectConnect and Adaptive Server Enterprise) CharSet CommitOnDisconnect CursorType CursorUpdate DateTimeAllowed DBGetTime DBTextLimit DelimitIdentifier DisableBind DS_Alias DS_Copy DS_DitBase DS_Failover DS_Password DS_Principal DS_Provider DS_TimeLimit FoDialog FormatArgsAsExp GenerateEqualsNull Host KeepAlive Language Locale Log	MaxConnect OJSyntax PacketSize (ASE, DIR, SNC, SYC) PBCatalogOwner ProxyUserName PWDialog PWEncrypt PWExpDialog Release Sec_Channel_Bind Sec_Confidential Sec_Cred_Timeout Sec_Data_Integrity Sec_Data_Origin Sec_Delegation Sec_Keytab_File Sec_Mechanism Sec_Mutual_Auth Sec_Network_Auth Sec_Replay_Detection Sec_Seq_Detection Sec_Server_Principal Sec_Sess_Timeout ShowTempTables StaticBind SvrFailover SystemProcs TableCriteria TrimSpaces UnicharBind UTF8
SYJ Sybase Adaptive Server Enterprise	Block (DirectConnect and Adaptive Server Enterprise) CacheName CursorUpdate DBTextLimit FormatArgsAsExp GenerateEqualsNull GetConnectionOption Log PBCatalogOwner	ProxyUserName ReleaseConnectionOption StaticBind SvrFailover SystemProcs TrimSpaces UseContextObject UnicharBind UTF8

ADORelease

Description

Specifies the version of the ADO.NET data provider that is in use on the client workstation.

When to specify ADORelease
You must specify a value for ADORelease *before* connecting to the database.

Applies to

ADO.NET

Syntax

ADORelease='*value*'

Parameter	Description
value	Specifies the version of an ADO.NET data provider your application uses. For Sybase Adaptive Server® Enterprise, the optional values are: • **1.1.411.0** For Adaptive Server 12.5.x clients • **1.15.50.0** For Adaptive Server 15.x clients For Oracle, the optional values are: • **9.2.0.401** For Oracle9*i* clients • **10.1.0.301** For Oracle 10*g* clients

Default value

1.1.411.0 for Adaptive Server, 9.2.0.401 for Oracle.

Usage

The ADORelease database parameter specifies the version of the ADO.NET database provider used for native connections to a database server using ADO.NET.

For Adaptive Server, specify Sybase.Data.AseClient as the Namespace in the Database Profile Setup dialog box for ADO.NET to display available versions of the Sybase ASE ADO.NET Data Provider in the Release drop-down list.

The database provider is supplied in the .NET assemblies *Sybase.PowerBuilder.Db.dll* (for Adaptive Server 12.5.x and Oracle9*i*) and *Sybase.PowerBuilder.DbExt.dll* (for Adaptive Server 15.x and Oracle 10*g*). You must deploy the appropriate version of this DLL with your application.

For Oracle, specify Oracle.DataAccessClient as the Namespace in the Database Profile Setup dialog box for ADO.NET to display available versions of the Oracle Data provider for .NET (ODP.NET) in the Release drop-down list.

When you select a driver version, the ADO.NET interface attempts to load that driver. If the driver is redirected to a higher version of the driver, the higher driver is loaded, but only the features in the selected driver may be supported. For example with Oracle ODP.NET, if ADORelease is set to 9.2.0.401 but the policy file on your computer redirects the driver to version 10.1.0.301, the ODP.NET 10.1.0.301 driver is loaded. New features in ODP.NET 10.1.0.301 are not supported.

Examples

To specify that your PowerBuilder application accesses an Adaptive Server 15 database using the ASE ADO.NET Data Provider:

- **Database profile** Select 1.1.50.0 from the ADORelease drop-down list on the Connection page in the Database Profile Setup dialog box.
- **Application** Type the following in code:

```
SQLCA.DBParm="ADORelease='1.15.50.0'"
```

AppDriverName

Description

Allows you to set your own client driver name for the Oracle database connection.

Applies to

ORA Oracle 11*g*

Syntax

AppDriverName =*value*

Parameter	Description
value	Specifies the name of the driver to use for an Oracle database connection.

Default value

None.

Usage

The maximum length of the name is 8 characters. You can display the client driver name with the V$SESSION_CONNECT_INFO or GV$SESSION_CONNECT_INFO dynamic performance view queries.

Examples

To specify a client driver name:

- **Database profile** Type the driver name in the Application Driver Name text box on the System page of the Database Profile Setup dialog box.
- **Application** Type the following in code where *myDriver* is a string for the name of the client driver:

```
sqlca.dbparm="AppDriverName='myDriver'"
```

AppName

Description

If the DBMS supports it, specifies the application name you want to use when connecting to the database in PowerBuilder.

When to specify AppName
You must specify the AppName parameter *before* connecting to the database.

Applies to

ASE, SYC Sybase Adaptive Server Enterprise
DIR Sybase DirectConnect
SNC SQL Native Client for Microsoft SQL Server

Syntax

AppName='*application_name*'

Default value

For Adaptive Server and DirectConnect, PowerBuilder sets the CS_APPNAME connection property to PowerBuilder, as follows:

```
AppName='PowerBuilder'
```

For SQL Server, there is no default value for AppName.

Usage

Adaptive Server databases It is useful to specify a different AppName value for each of your Adaptive Server applications. If you are an administrator, you can query the MASTER.DBO.SYSPROCESSES table to determine which applications are running on the database server. The value specified for AppName displays in the program_name column of the MASTER.DBO.SYSPROCESSES table, making it easy to identify the applications.

Examples

Example 1 To set the application name to Test:

- **Database profile** Type the following in the Application Name box on the Network or System page in the Database Profile Setup dialog box:

  ```
  Test
  ```

- **Application** Type the following in code:

  ```
  SQLCA.DBParm="AppName='Test'"
  ```

Example 2 (*Does not apply to DirectConnect*) You can set the AppName and Host parameters in a single statement to specify both the application name and the host name. To set the application name to Sales and the host name to Fran:

- **Database profile** Type Sales in the Application Name box and Fran in the Workstation Name box on the Network page in the Database Profile Setup dialog box.
- **Application** Type the following in code:

```
SQLCA.DBParm="AppName='Sales',Host='Fran'"
```

See also

Host (applies only to ASE and SYC Sybase Adaptive Server Enterprise)

Async

Description

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

By default, PowerBuilder operates synchronously.

Applies to

ASE, SYC Sybase Adaptive Server Enterprise
DIR Sybase DirectConnect
I10 Informix
IN9 Informix
JDB JDBC
ODBC (if driver and back-end DBMS support this feature)
O90 Oracle9*i*
O10 Oracle 10*g*
ORA Oracle 11*g*

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 PowerBuilder 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 code, 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 or report, the following steps occur in order:

1. The database server compiles and executes the SQL statement.
2. PowerBuilder retrieves (fetches) the first row of data.
3. PowerBuilder 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 PowerBuilder retrieves the first row of data, you must 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 or report (the code can contain only a comment)
- Set the Async 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 must have done only the following to enable asynchronous operation:

- Coded a RetrieveRow event for the DataWindow object or report

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

Examples

Example 1 To enable asynchronous operation:

- **Database profile** Select the Asynchronous check box on the Transaction page in the Database Profile Setup dialog box.
- **Application** Type the following in code:

```
SQLCA.DBParm="Async=1"
```

Example 2 You can set the Async and DBGetTime parameters in a single statement. DBGetTime specifies the number of seconds you want PowerBuilder to wait for a response from the DBMS when you retrieve rows in a DataWindow object or report. 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 page in the Database Profile Setup dialog box.
- **Application** Type the following in code:

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

See also

DBGetTime

BindSPInput

Description

Specifies that PowerBuilder bind input parameters in dynamic SQL statements when executing a stored procedure.

Applies to

ADO.NET
O90 Oracle9*i*
O10 Oracle 10*g*
SNC SQL Native Client for Microsoft SQL Server

Syntax

BindSPInput=*value*

Parameter	Description
value	Specifies whether you want to bind input parameters in dynamic SQL statements when executing stored procedures. Values are: • **0** (Default) PowerBuilder does not bind parameters in dynamic SQL statements when executing stored procedures. • **1** PowerBuilder binds parameters in dynamic SQL statements when executing stored procedures.

Default value

BindSPInput=0

Usage

For SNC, when BindSPInput is set to 0, you can use the same syntax to declare a stored procedure in a script as you can when using the PowerBuilder OLE DB interface. When BindSPInput is set to 1, the SNC interface supports SQL Server large value datatypes as procedure IN/OUT parameters or function return values.

The syntax for declaring a procedure with SNC is:

DECLARE *logical_procedure_name* PROCEDURE FOR
  [@rc=]*procedure_name*
  {*@param1=value1* [OUTPUT], *@param2=value2* [OUTPUT], ...}
  {USING *transaction_object*};

[@rc=] indicates that you want to get the procedure's return value.

Use the keyword OUTPUT or OUT to indicate an output parameter if you want to get the output parameter's value.

If BindSPInput=0, *value1*, *value2*,... can be either PowerBuilder script variables or literal values. If BindSPInput=1, *value1*, *value2*,… must be PowerBuilder script variables. If you specify literal values, the interface returns a runtime error.

When you declare a dynamic SQL statement with a procedure, enter a question mark (?) for each IN/OUT parameter in the statement. Value substitution is positional. For examples, see Dynamic SQL Format 3 and 4 in the online Help.

For Oracle, set BindSPInput to 1 to ensure that CLOB, NCLOB, and BLOB parameters work correctly as stored procedure parameters.

For ADO.NET:

- When BindSPInput is set to 1, parameter values must be PowerBuilder script variables, not literal values.
- The IBM.Data.Informix driver (used to access an ADO.NET compliant Informix database) does not support the BindSPInput dbparm.
- The ADO.NET Microsoft SQL Server interface does not support Text, NText, or Image parameters. Use VarChar(max) or VarBinary(max) instead.

Examples

Setting BindSPInput To specify that PowerBuilder should bind parameters in dynamic SQL statements when executing a stored procedure:

- **Database profile** Select the Bind Procedure Parameters check box on the Transaction page in the Database Profile Setup dialog box.
- **Application** Type the following in code:

```
SQLCA.DBParm="BindSPInput=1"
```

Using the ADO.NET SQL Server interface Consider the following two SQL statement fragments:

```
create procedure p_1 (@inparm1 TEXT) AS
```

The preceding statement does not work if BindSPInput is set to 1, because the SQL Server interface does not support Text.

```
create procedure p_2 (@inparm1 VARCHAR(MAX)) AS
```

The preceding statement can work with BindSPInput set to 1, because the SQL Server interface does support VARCHAR(MAX).

See also

DisableBind

BinTxtBlob

Description

Specifies that binary data or an ANSI string is to be submitted to or retrieved from a text column with UPDATEBLOB or SELECTBLOB.

Applies to

ASE, SYC Sybase Adaptive Server Enterprise
I10 Informix
IN9 Informix
OLE DB
O90 Oracle9*i*
O10 Oracle 10*g*
ORA Oracle 11*g*
SNC SQL Native Client for Microsoft SQL Server

Syntax

BinTxtBlob=*value*

Parameter	**Description**
value	Specifies that a PowerBuilder blob that is submitted to a text column in a database with UPDATEBLOB or retrieved with SELECTBLOB contains binary data or an ANSI string. Values are: • **0** The blob contains a Unicode string. • **1** The blob contains any kind of data that is not regarded as a Unicode string.

Default value

BinTxtBlob=0

Usage

By default, when you use the UPDATEBLOB and SELECTBLOB SQL statements with a database column with a text datatype (long or clob for Oracle or text for Adaptive Server or OLE DB access to SQL Server), the PowerBuilder blob that is updated or selected is expected to contain a Unicode string. If the blob contains any other kind of data, such as binary data or an ANSI string, set the BinTxtBlob database parameter to 1 before calling SELECTBLOB or UPDATEBLOB. This prevents PowerBuilder or the database server from attempting to perform any conversion to or from Unicode.

Oracle O90, O10, and ORA The Oracle database interfaces use a Unicode database handle. With the default setting (BinTxtBlob=0) and UPDATEBLOB, they send the data directly to the Oracle server and inform the server that the binary data contains Unicode strings. Any conversion needed is performed by the server. For SELECTBLOB, they get a Unicode string from the server.

When BinTxtBlob is set to 1, the value of the NLS_LANG environment variable determines the binding character set. The ANSI string or binary data is transferred directly to or from the server as in PowerBuilder 9 and previous releases.

To set BinTxtBlob to 1 with the O90 interface, you must use an Oracle 9.2 or later client, or you will receive an error.

OLE DB, SNC, ASE, and SYC If BinTxtBlob is set to 0, the OLE DB, SNC, ASE, and SYC interfaces perform any necessary conversion. If BinTxtBlob is set to 1, the data is passed to the server without conversion.

Examples

In code, before calling SELECTBLOB or UPDATEBLOB with a PowerBuilder blob that contains ANSI string data or binary data, set the BinTxtBlob parameter to 1:

```
SQLCA.DBParm="BinTxtBlob=1"
```

Restore the default setting of 0 if an operation needs to be performed on a blob that contains Unicode string data.

For example, suppose a Unicode string “ABC” stored in client memory as “65 00 66 00 67 00” is updated to the database using UPDATEBLOB. If BinTxtBlob is set to 0, the data is converted to ANSI and stored in the database text column as “65 66 67”. If BinTxtBlob is set to 1, no conversion occurs and the data is stored in its original form as “65 00 66 00 67 00”.

Block (ODBC, OLE DB, Oracle, and SNC)

Description

For those interfaces that support it, Block specifies the cursor blocking factor when connecting to a database. The blocking factor determines the number of rows that a DataWindow object can fetch from the database at one time.

Using the Block parameter can improve performance when accessing a database in PowerBuilder.

Applies to

ODBC (if driver and back-end DBMS support this feature)
OLE DB
O90 Oracle9*i*
O10 Oracle 10*g*
ORA Oracle 11*g*
SNC SQL Native Client for Microsoft SQL Server

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

The default value for the Block parameter depends on the DBMS you are accessing, as summarized in the following table:

DBMS	Block default value

ODBC	For most DataWindow objects, the Block default value is the following, *up to a maximum of 32K per column*:
	`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 the following, *up to a maximum of 32K per column*:
	`Block=100`

OLE DB	PowerBuilder sets the blocking factor to 1
Oracle	PowerBuilder sets the blocking factor dynamically if one row size is too large. By default the blocking factor is 300 rows.

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 PowerBuilder should meet your needs.

Usage

Requirements for ODBC data sources To use the Block database 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 PowerBuilder 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 for ODBC data sources PowerBuilder searches the following in this order to determine the Block value for ODBC data sources:

1 The section for your database profile in the registry or the value of the Transaction object property (in an application)

2 The section for your ODBC driver in the PBODB125 initialization file

If PowerBuilder 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 or Oracle database, set the Block parameter to 1.

OLE DB and Microsoft SQL Server When you use the OLE DB database interface with a Microsoft SQL Server database and retrieve data into a DataWindow or use an embedded SQL cursor in a SELECT statement, server-side cursors are used to support multiple command execution. If this has a negative impact on performance, try increasing the size of the Block database parameter to 500 or more, or adding the following line to the [Microsoft SQL Server] section in the *PBODB125.INI* file to turn off server-side cursors:

```
ServerCursor='NO'
```

Oracle and MaxFetchBuffer For Oracle, the Block parameter can be used in conjunction with the MaxFetchBuffer database parameter to improve performance when the size of a row is very large. The MaxFetchBuffer parameter has a default value of 5000000 bytes, which is sufficient for most applications. The size of the actual fetch buffer is the product of the value of the blocking factor and the size of the row.

If the fetch buffer required by the blocking factor and the row size is greater than the value of MaxFetchBuffer, the value of the blocking factor is adjusted so that the buffer is not exceeded. For example, if block=500 and the row size is 10KB, the fetch buffer is 5000KB, which equals the default maximum buffer size.

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 page in the Database Profile Setup dialog box:
- **Application** Type the following in code:

  ```
  SQLCA.DBParm="Block=50"
  ```

See also

MaxFetchBuffer

Block (DirectConnect and Adaptive Server Enterprise)

Description

Specifies the internal blocking factor used by the Sybase Client Library (CT-Lib) interface when declaring a cursor. The blocking factor determines the number of rows fetched from the database at one time when CT-Lib makes a physical request for data.

The Block DBParm parameter applies only to declared cursors and *not* to DataWindow objects.

Applies to

ASE, SYC, and SYJ Sybase Adaptive Server Enterprise
DIR Sybase DirectConnect

Syntax

Block=*blocking_factor*

Parameter	Description
blocking_factor	The number of rows fetched from the database at one time when CT-Lib makes a physical request for data (default=100 rows)

Default value

Block=100

Examples

Example 1 To set the blocking factor to 1000 rows:

- **Database profile** Type the following in the Retrieve Blocking Factor box on the Transaction page in the Database Profile Setup dialog box:

  ```
  1000
  ```

- **Application** Type the following in code:

  ```
  SQLCA.DBParm="Block=1000"
  ```

Example 2 The following embedded SQL statements show how to set the blocking factor in code and use it to declare a cursor. These statements set the blocking factor to 1000 rows and declare a cursor that uses this internal blocking factor.

```
SQLCA.DBParm="Block=1000"
DECLARE dept_cursor CURSOR FOR
      SELECT dept_id, dept_name FROM department
      USING SQLCA;
OPEN dept_cursor;
```

CacheAuthentication

Description

Specifies whether the OLE DB data provider can cache sensitive authentication information, such as a password, in an internal cache.

When to specify CacheAuthentication
You must specify the CacheAuthentication parameter *before* connecting to the database.

Applies to

OLE DB

Syntax

CacheAuthentication='*value*'

Parameter	Description
value	Specifies whether the OLE DB data provider can cache authentication information. Values are: • **True** Tells the OLE DB data provider to cache information. • **False** (Default) Tells the OLE DB data provider not to cache information.

Default value

CacheAuthentication='False'

Examples

To tell the OLE DB data provider to cache authentication information:

- **Database profile** Select the Cache Authentication check box on the Security page in the Database Profile Setup dialog box.
- **Application** Type the following in code:

```
SQLCA.DBParm="CacheAuthentication='True'"
```

See also

DataLink
IntegratedSecurity

CacheName

Description

Allows PowerBuilder to specify an EAServer connection cache by name. This database parameter applies *only* when a PowerBuilder custom class user object is deployed as an EAServer component.

Applies to

JDB JDBC
ODBC
O90 Oracle9*i*
SYJ Sybase Adaptive Server Enterprise

Using the SYJ interface
Sybase EAServer uses a slightly different version of the CT-Lib software. Therefore, *at runtime*, you need to use the SYJ database interface rather than ASE and SYC to connect to an Adaptive Server Enterprise database. The SYJ Database Profile Setup dialog box provides a convenient way to set the appropriate connection parameters and then copy the syntax from the Preview page into the script for your Transaction object.

You cannot use the SYJ interface, however, to connect to the database in the PowerBuilder development environment. Therefore, *during the development phase* (before the component has been deployed to EAServer), you must use ASE or SYC to connect to the database.

Syntax	CacheName='*value*'
Default value	None

Usage

When you create a PowerBuilder custom class user object that uses an EAServer connection cache, you must specify a user name, password, server name, and connectivity library. However, if the “enable cache-by-name access” option has been enabled on EAServer, you just need to enter the connection cache name to specify the connection cache.

For information on how to use PowerBuilder to build EAServer components, see *Application Techniques*.

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

Examples

On the EAServer page in the Database Profile Setup dialog box, select the Access Cache By Name check box and enter the EAServer cache name in the Cache Name box. The PowerScript syntax for the CacheName DBParm parameter displays on the Preview page:

```
SQLCA.DBParm="CacheName='mydbcache'"
```

Copy the syntax from the Preview page into your script.

See also

GetConnectionOption
ReleaseConnectionOption
UseContextObject

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 (if driver and back-end DBMS support this feature)

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 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.

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

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

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 page in the Database Profile Setup dialog box.
- **Application** Type the following in code:

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

CharSet

Description

Specifies the character set you want the Sybase Open Client™ software to use when connecting to a Sybase Adaptive Server Enterprise database or a database accessed through DirectConnect.

When to specify CharSet
You must specify the CharSet parameter *before* connecting to a database.

Applies to

ASE, SYC Sybase Adaptive Server Enterprise
DIR Sybase DirectConnect

Syntax

CharSet='*character_set*'

Default value

None

Usage

When you specify a value for CharSet, PowerBuilder:

- Allocates a CS_LOCALE structure for this connection
- Sets the CS_SYB_CHARSET value to the character set you specify
- Sets the SQL Server CS_LOC_PROP connection property with the new locale information

Overriding the Locale parameter If you have previously set a value for the Locale parameter that includes settings for the language and character set you want to use, you can override the character set value by specifying a new value for the CharSet parameter and reconnecting to the database.

Unicode data access PowerBuilder can access Unicode data in an Adaptive Server Enterprise (ASE) 12.5 or later Unicode database or in Unicode columns in ASE 12.5 or later. PowerBuilder converts between double-byte character set (DBCS) data and Unicode automatically, provided that the CharSet and Language parameters are set with DBCS values (or the Locale parameter is set with DBCS values). For example:

```
CharSet='big5'
Language='tchinese'
```

Examples

To set the character set to iso_1:

- **Database profile** Type the following in the Character Set box on the Connection page or Regional Settings page in the Database Profile Setup dialog box:

  ```
  iso_1
  ```

- **Application** Type the following in code:

```
SQLCA.DBParm="CharSet='iso_1'"
```

See also

Language
Locale

Client_Locale

Description

Client_Locale identifies the locale that the client application uses.

When to specify Client_Locale
You must specify the Client_Locale parameter *before* connecting to a database.

Applies to

I10 Informix

Syntax

Client_locale='*language_territory*.*codeset*'

Parameter	Description
language	Two character name that represents the language for a specific locale, for example “en” for English.
territory	Two character name that represents the cultural conventions for a specific territory, for example “AU” for Australia.
codeset	Name of the code set that the locale supports, for example “utf8.”

Default value

Based on the operating system’s locale.

Usage

The I10 native interface uses the Informix GLS (Global Language Support) API for global language support. Client_Locale specifies the value of the Informix environment variable CLIENT_LOCALE. The I10 interface uses this setting to access string data in an Informix database and to process SQL statements. If you do not set the DBParm, the default client locale value is based on the OS locale.

For example, to access a database that has a Japanese SJIS locale, Client_Locale should be set to `ja_jp.sjis` on the client system.

For more information about the Informix CLIENT_LOCALE and DB_LOCALE environment variables, see the *IBM Informix GLS User's Guide*, currently available at the Informix library Web site at http://publib.boulder.ibm.com/infocenter/idshelp/v111/index.jsp?topic=/com.ibm.glsug.doc/glsug.htm.

Examples

To set the client locale to 'en_us.utf8':

- **Database profile** Type the following in the Client Locale box on the Regional Settings page in the Database Profile Setup dialog box:

  ```
  en_us.utf8
  ```

- **Application** Type the following in code:

  ```
  SQLCA.DBParm="Client_Locale='en_us.utf8'"
  ```

See also

Db_Locale
StrByCharset

CnnPool

Description

Specifies whether Oracle should maintain connections in a pool. An Oracle connection pool is a group of reusable physical connections spanning several sessions and managed by the Oracle Call Interface (OCI). The CNNPool DBParm is maintained in the ORA driver for backward compatibility only. It is ignored if you use the Pooling DBParm.

The CnnPool parameter cannot be used for PowerBuilder components deployed to EAServer, and external users cannot participate in a connection pool. By default, connection pooling is not used.

Applies to

O90 Oracle9*i*
O10 Oracle 10*g*
ORA Oracle 11*g*

Syntax

CnnPool='*value*'

Parameter	Description
value	Values are: • Yes • No (Default)

Default value

No

Usage

If `CnnPool='Yes'`, the Oracle client creates a connection pool and can connect to Oracle9*i*, and Oracle 10*g* Server. The pool contains physical connections to Oracle Server and is managed by the OCI. The pool persists until PowerBuilder terminates or *OCI.dll* is unloaded.

PowerBuilder can connect to 10 different Oracle servers, as distinguished by service name, concurrently. There can be no more than 20 connections per pool. Each pool is created by the Oracle9*i* or later client and might also contain connections to Oracle8 and Oracle8*i* servers. Once a connection pool has been created, PowerBuilder maintains the physical connections until it terminates or OCI.DLL is unloaded.

Examples

To use connection pooling:

- **Database profile** Enter values in the following fields of the Connection page in the Database Profile Setup dialog box:
    - *Profile Name* – Example value: `08i-pool`
    - *Server* – Provide the full net service name created by Oracle Net. Example value: `adcora8i.sybase.com`
    - *Login ID* – Example value: `scott`
    - *Password*
    - *Connect as* – Choose an item from the drop-down menu. Example value: `Normal`

  Make sure Use Connection Pool is selected.

- **Application** Type the following in code where *password*, *server_name*, and *login* are the appropriate values for your connection:

```
SQLCA.DBMS="O90 Oracle9i (9.0.1)"
SQLCA.LogPass=<password>
SQLCA.ServerName="server_name"
SQLCA.LogId="login"

SQLCA.AutoCommit=False
SQLCA.DBParm="CnnPool='Yes'"
```

See also

StatementCache

CommandTimeout

Description

Specifies the number of seconds the interface should wait for a command to execute.

When to specify CommandTimeout
You must specify a value for CommandTimeout *before* connecting to the database.

Applies to

ADO.NET

Syntax

CommandTimeout=*value*

Parameter	Description
value	The number of seconds the interface waits for a command to execute.

Default value

None

Usage

The default value for the CommandTimeout parameter is driver-specific.

When you use this parameter with an AdoTransaction object in DataWindow .NET, specify a value for this parameter in the second argument of the AdoTransaction constructor.

Examples

To set the CommandTimeout value to wait 60 seconds for a command to execute:

- **Database profile**   Type `60` in the Command Timeout box on the System page in the Database Profile Setup dialog box.
- **Application**   Type the following in code:

  ```
  SQLCA.DBParm="CommandTimeout=60"
  ```

  Add the following to the second argument of the AdoTransaction constructor:

  ```
  CommandTimeout=60
  ```

CommitOnDisconnect

Description

Specifies whether PowerBuilder 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.

Applies to

All database interfaces except SYJ

Syntax

CommitOnDisconnect='*value*'

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

Default value

CommitOnDisconnect='Yes'

Usage

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

Examples

To tell PowerBuilder 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 page in the Database Profile Setup dialog box.
- **Application** Type the following in code:

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

ConnectAs

Description

Allows the user to connect to the Oracle Server with SYSOPER or SYSDBA system privileges. Supports proxy authentication with additional user names (ORA driver only) when connecting to Oracle 10.2 or higher database servers.

Applies to

O90 Oracle9*i*
O10 Oracle 10*g*
ORA Oracle 11*g*

Syntax

ConnectAs='*value*'

Parameter	Description
value	Values are: • *EndUser* (ORA driver only) Type the proxy user name you want to use for your connection • `SYSDBA` • `SYSOPER` • `Normal` (Default)

Default value

Normal

Usage

If ConnectAs='Normal', this parameter is not used. If ConnectAs ='SYSDBA' or ConnectAs ='SYSOPER', Oracle allows the user to connect with SYSDBA or SYSOPER privileges, respectively, provided that these privileges have been granted the user.

If you connect using ConnectAs ='SYSDBA', Oracle uses the SYS schema instead of the schema that might already be associated with the user ID. If you connect using ConnectAs ='SYSOPER', Oracle uses the PUBLIC schema.

The PowerBuilder ORA driver supports the proxy authentication feature that was introduced in Oracle 10.2. With proxy authentication, the end user typically authenticates to a middle tier (such as a firewall), that in turn logs into the database on the user's behalf as a proxy user. After logging into the database, the proxy user can switch to the end user's identity and perform operations using the authorization accorded to that user.

The ConnectAs DBParm parameter allows you to take advantage of this proxy connection feature. For example, if the user’s Transaction object LogID is “Scott” and you set the ConnectAs DBParm parameter to “John”, the OCI client logs in to database as the proxy user (“Scott”), then switches to the end user identity (“John”).

If you are using connection or session pooling, the proxy user name is the connection or session pooling creator (which you can provide in the PoolCreator and PoolPwd DBParm parameters), and the Transaction object’s LogID is ignored. No proxy session can be created if pooling is set to homogeneous session mode.

Do not use with CnnPool='Yes'
Connection pooling cannot be used with this parameter. Do not select Use Connection Pool in the Database Profile Setup dialog box or set CnnPool to 'Yes'.

Examples

Example 1 To use the SYS schema instead of the schema associated with the User ID:

- **Database profile** Select SYSDBA from the Connect As drop-down list on the Connection page in the Database Profile Setup dialog box, and make sure Use Connection Pool is deselected.
- **Application** Type the following in code:

```
SQLCA.DBParm="ConnectAs='SYSDBA'"
```

Example 2 To connect as a proxy user (Scott) for the end user named John:

- **Database profile** Type `John` in the Login ID text box on the Connection page in the Database Profile Setup dialog box, and type `Scott` in the Connect As drop-down list.
- **Application** Type the following in code:

```
sqlca.logid = "scott"
sqlca.dbparm="ConnectAs='john'"
```

ConnectOption

Description

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

- How the ODBC driver prompts for additional connection information
- What type of security to use 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. The ConnectOption settings take effect when you connect to the database.

Applies to

ODBC (if driver and back-end DBMS support this feature)

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.

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.

Parameter	**Description**
SQL_INTEGRATED_ SECURITY	**Driver** Microsoft SQL Server ODBC driver (not supplied with PowerBuilder). **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. For more about security mechanisms in Microsoft SQL Server, see the Microsoft documentation.
SQL_OPT_TRACE	**Driver** Any ODBC driver. **Purpose** Turns on or turns off the ODBC Driver Manager Trace in PowerBuilder to troubleshoot a connection to an ODBC data source. The ODBC Driver Manager Trace provides detailed information about the ODBC API function calls that PowerBuilder 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 “About ODBC Driver Manager” in *Connecting to Your Database*.
SQL_OPT_TRACEFILE	**Driver** Any ODBC driver. **Purpose** Specifies the name of the trace file where you want PowerBuilder to send the output of the ODBC Driver Manager Trace. PowerBuilder 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 PowerBuilder) or any text editor (outside PowerBuilder). **Values** You can specify any filename 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, PowerBuilder sends ODBC Driver Manager Trace output to the file \SQL.LOG.

Parameter	Description
SQL_PRESERVE_ CURSORS	**Driver** Microsoft SQL Server ODBC driver (not supplied with PowerBuilder). **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.
SQL_USE_PROCEDURE_ FOR_PREPARE	**Driver** Microsoft SQL Server ODBC driver (not supplied with PowerBuilder). **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

Microsoft Server ODBC driver The ConnectOption parameter applies only if you are accessing a SQL Server database with the Microsoft ODBC SQL Server driver.

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

Examples

To specify nondefault options for the ConnectOption parameter:

- **Database profile** Complete the Options page in the Database Profile Setup - ODBC 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

- **Application** Type the following in code:

```
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:\PB\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 an ODBC data source. PowerBuilder uses these parameters to connect to the database.

Applies to

ODBC

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.

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.

Default value

None

Usage

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

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.

Examples

Example 1 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 page 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`.
- **Application** Type the following in code:

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

CSIncr

Description

Specifies an integer for database connection increments per session.

Applies to

ORA Oracle 11*g* and Oracle 9.2 or later databases

Syntax

CSIncr =*value*

Parameter	Description
value	Specifies an increment for the number of database connections in a connection or session pool.

Default value

1

Usage

This value is ignored when the SessionHomogeneous DBParm is set to false.

Examples

The following example sets the increment for the number of database connections in a session pool to 10:

- **Database profile** Select Session Pooling from the Pooling Type drop-down list on the Pooling page in the Database Profile Setup dialog box, select the Homogeneous Session Pooling check box, and type 10 in the Increment text box on the same page.
- **Application** Type the following in code:

```
my_trans.dbparm="Pooling='session',csincr = 10"
```

See also

CSMax
CSMin

CSMax

Description

Specifies the maximum number of database connections in a connection or session pool.

Applies to

ORA Oracle 11*g* and Oracle 9.2 or later databases

Syntax

CSMax =*value*

Parameter	Description
value	Specifies the maximum number of database connections in a connection or session pool.

Default value

100

Usage

This value is ignored when the SessionHomogeneous DBParm is set to false.

Examples

The following example limits the number of database connections in a connection pool to 15:

- **Database profile** Select Connection Pooling from the Pooling Type drop-down list on the Pooling page in the Database Profile Setup dialog box, and type 15 in the Maximum Number of Sessions text box on the same page.
- **Application** Type the following in code:

```
my_trans.dbparm="pooling='connection',csmax = 15"
```

See also

CSIncr
CSMin

CSMin

Description

Specifies the minimum number of database connections in a connection or session pool.

Applies to

ORA Oracle 11*g* and Oracle 9.2 or later databases

Syntax

CSMin =*value*

Parameter	Description
value	Specifies the minimum number of database connections in a connection or session pool.

Default value

1

Usage

This value is ignored when the SessionHomogeneous DBParm is set to false.

Examples

The following code keeps a minimum of 5 database connections open in a connection pool:

- **Database profile** Select Connection Pooling from the Pooling Type drop-down list on the Pooling page in the Database Profile Setup dialog box, and type 5 in the Minimum Number of Sessions text box on the same page.
- **Application** Type the following in code:

```
my_trans.dbparm="pooling='connection',csmin = 5"
```

See also

CSIncr
CSMax

CursorLib

Description

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

Applies to

ODBC (if driver and back-end DBMS support this feature)

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 page in the Database Profile Setup dialog box.
- **Application** Type the following in code:

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

CursorLock

Description

When used with the CursorScroll parameter, specifies locking options for cursors in 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 (if driver and back-end DBMS support this feature)

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, PowerBuilder 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** On the Transaction page in the Database Profile Setup dialog box, select Dynamic Scrolling from the Scrolling Options drop-down list, and Optimistic Using Values from the Locking drop-down list.
- **Application** Type the following in code:

```
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 (if driver and back-end DBMS support this feature)

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, PowerBuilder defaults to the cursor scroll settings specified for your ODBC data source driver.

Usage

For large result sets, it might 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

Example 1 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 page in the Database Profile Setup dialog box.
- **Application** Type the following in code:

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

Example 2 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 page in the Database Profile Setup dialog box.
- **Application** Type the following in code:

```
SQLCA.DBParm="CursorScroll=100"
```

See also

CursorLock

CursorType

Description

Supports the scrollable cursor feature introduced in Adaptive Server Enterprise 15.0, including directional scrolling (forwards and backwards) and sensitivity towards independent changes to table.

Applies to

ASE, SYC Sybase Adaptive Server Enterprise (15.0 and later)

Syntax

CursorType='*value*'

Parameter	Description
value	A string that specifies whether database cursors are scrollable and whether they are sensitive to modifications in data. Values are: • **NonScrollable** (Default) Non-scrollable forward-only cursor. Supports the PowerScript FETCH NEXT syntax. • **ScrollInsensitive** Scrollable insensitive cursor that ignores any data modifications when scrolling in either direction. Supports the PowerScript FETCH NEXT, FETCH PRIOR, FETCH FIRST, and FETCH LAST syntaxes. • **ScrollSemiSensitive** Scrollable semi-sensitive cursor that presents data modifications when scrolling forwards but ignores them when scrolling backwards. Supports the PowerScript FETCH NEXT, FETCH PRIOR, FETCH FIRST, and FETCH LAST syntaxes.

Default value

CursorType='NonScrollable'

Usage

Adaptive Server Enterprise 15.0 allows both scrollable and nonscrollable cursors, which can be either semi-sensitive or insensitive. “Scrollable” means that you can scroll through the cursor result set by fetching any, or many, rows, rather than one row at a time; you can also scan the result set repeatedly. A scrollable cursor allows you to set the position of the cursor anywhere in the cursor result set for as long as the cursor is open.

To use a scrollable cursor, you must use a DECLARE *CursorName* CURSOR SQL statement to declare it with a suitable SELECT statement, such as the PowerBuilder Dynamic SQL Format 3 and Dynamic SQL Format 4 statements, and you must have the query engine provided in Adaptive Server 15.0 or later.

For sensitive scrolling to work correctly, the table must have a clustered index or a clustered unique constraint, such as a clustered primary key.

All scrollable cursors are read-only and can only be used when the value of the CursorUpdate database parameter is 0 (the default). If you need an updatable cursor, set the CursorUpdate parameter to 1. When CursorUpdate is set to 1, the value of CursorType is ignored. All update cursors are nonscrollable.

If a scrollable cursor is moved to a position before the first row or after the last row, `SQLCA.SQLCode` returns 100 and no data is returned. However, users can continue to fetch data by using a suitable FETCH statement after receiving this SQLCode value.

Both client and server must be Adaptive Server 15.0 or higher.

Examples

To specify support for semi-sensitive scrollable cursors (data modifications are presented when scrolling forwards):

- **Database profile** Select Cursor Scrollable SemiSensitive from the Read Only Cursor Type drop-down list on the Transaction page in the Database Profile Setup dialog box.
- **Application** Type the following in code:

```
SQLCA.DBParm="CursorType='ScrollSemiSensitive'"
```

See also

CursorUpdate

CursorUpdate

Description

For those interfaces that support it, CursorUpdate specifies whether cursors in your target database are declared read-only or updatable.

Applies to

DIR Sybase DirectConnect
ASE, SYC, and SYJ Sybase Adaptive Server Enterprise

Syntax

CursorUpdate=*value*

Parameter	Description
value	A number that specifies whether database cursors are declared read-only or updatable. Values are: • **0** (Default) Cursors are declared read-only. Sybase Client Library cursor declarations include the CS_READ_ONLY option. • **1** Cursors are declared updatable. Sybase Client Library cursor declarations include the CS_FOR_UPDATE option. This option applies to all updatable columns in the table.

Default value	CursorUpdate=0

Usage

Set the CursorUpdate parameter to 1 to declare updatable cursors if you plan to use either of the following SQL statements in your application (*table* represents the table name and *cursor* represents the cursor name):

DELETE FROM *table* WHERE CURRENT OF *cursor*
UPDATE *table* SET *set_clause* WHERE CURRENT OF *cursor*

If you are communicating with the database in a PowerBuilder script, you can reset the CursorUpdate value anytime before or after the Transaction object has connected to the database.

When you declare cursors updatable in a database accessed through DirectConnect, the cursor declaration you code must include a FOR UPDATE OF *column_list* clause.

When you use updatable cursors with the DIR interface and a Gatewayless connection to the mainframe, you must set Block=1 before executing the cursor. You can reset the Block parameter to its default of 100 after you close the cursor within your code.

Examples

To specify that database cursors are declared updatable:

- **Database profile** Select the Cursors Declared Updatable check box on the Transaction page in the Database Profile Setup dialog box.
- **Application** Type the following in code:

```
SQLCA.DBParm="CursorUpdate=1"
```

Database

Description

Specifies the name of the database you want to connect to.

When to specify Database
You must specify the Database parameter *before* connecting to the database.

Applies to	ADO.NET SNC SQL Native Client for Microsoft SQL Server
Syntax	DataBase='*database_name*'
Default value	None
Examples	To connect to the database “mydb”:

- **Database profile** Enter `mydb` in the Database box on the Connection page in the Database Profile Setup dialog box.
- **Application** Type the following in code:

```
SQLCA.DBParm="Database='mydb'"
```

DataLink

Description

Specifies that you want to create a file or use an existing file containing your connection information to connect to your data source.

When to specify DataLink
You must specify the DataLink parameter *before* connecting to the database.

Applies to

ADO.NET
OLE DB
SNC SQL Native Client for Microsoft SQL Server

Syntax

DataLink='*file_name*'

Default value

None

Usage

The Data Link option allows you to access Microsoft's Data Link API. The Data Link API allows you to define a file or use an existing file that contains your connection information. A Data Link file is identified with the suffix *.udl*.

To launch the API, double-click on Manage Data Links under OLE DB Utilities in the Installed Database Interfaces list or select the File Name check box on the Connection page in the Database Profile Setup dialog box and click the button next to the File Name box.

For more information on using the Data Link API, see Microsoft's Universal Data Access Web site.

Using a Data Link file versus setting other database parameters If you use a Data Link file to connect to your data source with the ADO.NET or OLE DB interface, all other settings you make in the Database Profile Setup dialog box are ignored.

If you use a Data Link file to connect to your data source with the SNC interface, the setting in the ProviderString database parameter still takes effect. The SNC interface gets a connection string from the data link file, and then copies the contents of the ProviderString parameter into the connection string so that it contains the connection parameters from both the data link file and the ProviderString parameter. You might want to take advantage of this feature if you do not want to save the user name and password in the UDL file. You can specify them in the ProviderString parameter instead.

Examples

To use the file *oledb.udl* to connect to an OLE DB data provider:

- **Database profile** Select the File Name check box on the Connection page in the Database Profile Setup dialog box and enter a name for a new file or select an existing file.
- **Application** Type the following in code:

```
SQLCA.DBParm="DataLink='oledb.udl'"
```

DataSource

Description

Identifies the data source to which you want to connect. The data source can be a file, a database, or an ODBC data source depending on the OLE DB data provider you are using.

When to specify DataSource
You must specify the DataSource parameter *before* connecting to the database.

Applies to

ADO.NET
OLE DB

Syntax

DataSource='*datasource_name*'

Default value

None

Usage

For the SNC interface for SQL Server 2005, specifying a value for Server on the Connection page or specifying a value for SQLCA.ServerName in code is equivalent to setting this parameter.

The value of the Data Source parameter varies depending on the type of data source connection you are making. For example, if you are using Microsoft's OLE DB Provider for ODBC, you would enter the actual ODBC data source name for the Data Source value. If you are using Microsoft's OLE DB Provider for SQL Server, you would enter the actual Microsoft SQL Server server name for the Data Source value.

For more information, see the documentation provided by your OLE DB data provider.

Examples

Example 1 To use the Microsoft OLE DB Provider for ODBC to connect to the EAS Demo DB:

- **Database profile** Enter EAS Demo DB in the Data Source box on the Connection page in the Database Profile Setup dialog box.
- **Application** Type the following in code:

```
SQLCA.DBParm="DataSource='EAS Demo DB'"
```

Example 2 To use the Microsoft OLE DB Provider for Oracle to connect to an Oracle 8 database:

- **Database profile** Enter the Oracle 8 server name in the Data Source box on the Connection page in the Database Profile Setup dialog box.
- **Application** Type the following in code:

```
SQLCA.DBParm="DataSource='Or8server'"
```

See also

DataLink
Provider

Date

Description

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

Applies to

JDB JDBC
ODBC
O90 Oracle9*i*
O10 Oracle 10*g*
ORA Oracle 11*g*

Syntax

The syntax you use to specify the Date parameter 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 date format.

In a PowerBuilder application script, you must use the following syntax:

JDBC and ODBC syntax PowerBuilder parses the backslash followed by two single quotes (`\''`) as a single quote when it builds the SQL UPDATE statement.

Date=' \''*date_format*\'' '

Oracle syntax PowerBuilder parses each set of four consecutive single quotes (`''''`) as a single quote when it builds the SQL UPDATE statement.

Date=' ''''*date_format*'''' '

Parameter	Description
' \'' *' ''''*	**JDBC and ODBC syntax** 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. **Oracle syntax** Type a single quote, followed by one space, followed by four single quotes. There is no space between the four single quotes and the beginning of the date format.
date_format	The date format you want PowerBuilder 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*.
\'' ' *'''' '*	**JDBC and ODBC syntax** 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. **Oracle syntax** Type four single quotes, followed by one space, followed by a single quote. There is no space between the end of the date format and the four single quotes.

Default value

The default value for Date depends on the DBMS you are accessing, as summarized in the following table:

DBMS	Date default value
JDBC	If no value is specified for the Date database parameter, PowerBuilder looks for a date format in the section for your JDBC driver in the registry. If no date format is found in the registry, PowerBuilder uses the JDBC date format escape sequence.
ODBC	If no value is specified for the Date database parameter, PowerBuilder looks for a date format in the section for your ODBC driver in the PBODB125 initialization file. If no date format is found in the initialization file, PowerBuilder uses the ODBC date format escape sequence.

DBMS	Date default value
Oracle	The default Oracle date format. For information, see your Oracle documentation.

Examples

About these examples Assume you are updating a table named Employee by setting the Startdate column to 2006-04-23. This date is represented by the following date format:

```
yyyy-mm-dd
```

Example 1 (JDBC and ODBC syntax) To specify that PowerBuilder 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 page in the Database Profile Setup dialog box:

  ```
  yyyy-mm-dd
  ```

- **Application** Type the following in code:

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

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

```
UPDATE EMPLOYEE
SET STARTDATE='2006-04-23'
```

Example 2 (Oracle syntax) To specify that PowerBuilder 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 page in the Database Profile Setup dialog box:

  ```
  yyyy-mm-dd
  ```

- **Application** Type the following in code:

  ```
  SQLCA.DBParm="Date=' ''''yyyy-mm-dd'''' '"
  ```

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

```
UPDATE EMPLOYEE
SET STARTDATE='2006-04-23'
```

See also

DateTime
Time

DateFormat

Description

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

Applies to

ADO.NET
OLE DB
SNC SQL Native Client for Microsoft SQL Server

Syntax

DateFormat='*date_format* '

Parameter	Description
date_format	The date format you want PowerBuilder 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*.

Default value

If no value is specified for the DateFormat parameter, PowerBuilder does not use a date datatype.

Usage

When you call stored procedures, the database server might not accept the date format built by PowerBuilder. If this occurs, you can try to use another format. For example, for Microsoft SQL Server, try this format:

```
DateFormat='\''yyyy-mm-dd\'''
```

Examples

Assume you are updating a table named Employee by setting the Startdate column to 2006-04-23. This date is represented by the date format `yyyy-mm-dd`.

To specify that PowerBuilder 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 page in the Database Profile Setup dialog box:

  ```
  yyyy-mm-dd
  ```

- **Application** Type the following in code:

  ```
  SQLCA.DBParm="DateFormat='yyyy-mm-dd'"
  ```

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

```
UPDATE EMPLOYEE
SET STARTDATE='2006-04-23'
```

See also

DateTimeFormat

TimeFormat

DateTime

Description

When you update data in the DataWindow painter, PowerBuilder builds a SQL UPDATE statement in the background. The DateTime parameter determines how PowerBuilder 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

JDB JDBC
ODBC
O90 Oracle9*i*
O10 Oracle 10*g*
ORA Oracle 11*g*

Syntax

The syntax you use to specify the DateTime 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 code, you must use the following syntax:

JDBC and ODBC syntax PowerBuilder parses the backslash followed by two single quotes (\'') as a single quote when it builds the SQL UPDATE statement.

DateTime=' \''*DateTime_format*\'' '

Oracle syntax PowerBuilder parses each set of four consecutive single quotes ('''') as a single quote when it builds the SQL UPDATE statement.

DateTime=' ''''*DateTime_format*'''' '

Parameter	Description
' \''	**JDBC and ODBC syntax** 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.
' ''''	**Oracle syntax** Type a single quote, followed by one space, followed by four single quotes. There is no space between the four single quotes and the beginning of the date format.

Parameter	Description
DateTime_format	The DateTime format you want PowerBuilder 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*.
\'' ' '''' '	**JDBC and ODBC syntax** 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. **Oracle syntax** Type four single quotes, followed by one space, followed by a single quote. There is no space between the end of the DateTime format and the four single quotes.

Default value

The default value for DateTime depends on the DBMS you are accessing, as summarized in the following table:

DBMS	Date default value
JDBC	If no value is specified for the DateTime database parameter, PowerBuilder looks for a DateTime format in the section for your JDBC driver in the registry. If no DateTime format is found in the registry, PowerBuilder uses the JDBC DateTime format escape sequence.
ODBC	If no value is specified for the DateTime database parameter, PowerBuilder looks for a DateTime format in the section for your ODBC driver in the *PBODB125* initialization file. If no DateTime format is found in the initialization file, PowerBuilder uses the ODBC DateTime format escape sequence.
Oracle	The default Oracle DateTime format. For information, see your Oracle documentation.

Examples

About these examples Assume you are updating a table named Files by setting the Timestamp column to 4/2/06 3:45 pm. This DateTime is represented by the following DateTime format:

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

Example 1 (JDBC, ODBC, and OLE DB syntax) To specify that PowerBuilder 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 page in the Database Profile Setup dialog box:

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

- **Application** Type the following in code:

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

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

```
UPDATE FILES
SET TIMESTAMP='4/2/06 3:45 pm'
```

Example 2 (Oracle syntax) To specify that PowerBuilder 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 page in the Database Profile Setup dialog box:

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

- **Application** Type the following in code:

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

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

```
UPDATE FILES
SET TIMESTAMP='4/2/06 3:45 pm'
```

See also

Date
Time

DateTimeAllowed

Description

For those interfaces that support it, DateTimeAllowed controls whether columns having a DateTime datatype can appear as unique key columns in the WHERE clause of a SQL UPDATE or DELETE statement. PowerBuilder generates an UPDATE statement or a DELETE statement followed by an INSERT statement to update the database from a DataWindow object.

When you are working in the DataWindow painter, you specify which columns to include in the WHERE clause by selecting them from the Unique Key Columns list in the Specify Update Properties dialog box.

By default, DateTimeAllowed is set to 0 to prohibit DateTime columns from displaying in the Unique Key Columns list and consequently from appearing in the WHERE clause of an UPDATE or DELETE statement. When you set DateTimeAllowed to 1, any DateTime columns in your database table display in the Unique Key Columns list and can be selected to appear in the WHERE clause of an UPDATE or DELETE statement.

When to specify DateTimeAllowed
You must specify a value for DateTimeAllowed *before* connecting to the database.

Applies to

ASE, SYC Sybase Adaptive Server Enterprise
DIR Sybase DirectConnect
I10 Informix
IN9 Informix

Syntax

DateTimeAllowed=*value*

Parameter	Description
value	Specifies whether you can use DateTime columns as unique key columns in a WHERE clause of a SQL UPDATE or DELETE statement generated by PowerBuilder to update the database. Values are: • **0** (Default) Prohibit the use of DateTime columns in the WHERE clause of an UPDATE or DELETE statement. When DateTimeAllowed is set to 0, DateTime columns *do not display* in the Unique Key Columns list in the Specify Update Properties dialog box. You can also specify `'No'` or `'False'` to set this value. • **1** Allow the use of DateTime columns in the WHERE clause of an UPDATE or DELETE statement. When DateTimeAllowed is set to 1, DateTime columns *do display* in the Unique Key Columns list in the Specify Update Properties dialog box so you can select one or more to appear in the WHERE clause. You can also specify `'Yes'` or `'True'` to set this value.

Default value

DateTimeAllowed=0

Usage

When to set To allow the use of DateTime columns as unique key columns in the WHERE clause of an UPDATE or DELETE statement when you are updating the database from a DataWindow object, set DateTimeAllowed to 1.

For instructions on using the Specify Update Properties dialog box to specify update characteristics for a DataWindow object, see the chapter on controlling updates in the *Users Guide*.

What happens when you save the DataWindow object When you set DateTimeAllowed to 1, select a DateTime column to appear in the WHERE clause, and then save the DataWindow object, this column continues to display in the Unique Key Columns list even if you set DateTimeAllowed to 0 on a subsequent connection.

Examples

To allow the use of DateTime columns in the WHERE clause of an UPDATE or DELETE statement:

- **Database profile** Select the DateTime Datatype Allowed check box on the Syntax page in the Database Profile Setup dialog box.
- **Application** Type the following in code:

```
    SQLCA.DBParm="DateTimeAllowed=1"
```

DateTimeFormat

Description

When you update data in the DataWindow painter, PowerBuilder builds a SQL UPDATE statement in the background. The DateTimeFormat parameter determines how PowerBuilder 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

ADO.NET
OLE DB
SNC SQL Native Client for Microsoft SQL Server

Syntax

DateTimeFormat='*datetime_format* '

Parameter	Description
datetime_format	The datetime format you want PowerBuilder 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*.

Default value

If no value is specified for the DateTimeFormat parameter, PowerBuilder does not use a datetime datatype.

Usage

When you call stored procedures, the database server might not accept the DateTime format built by PowerBuilder. If this occurs, you can try to use another format. For example, for Microsoft SQL Server, try this format:

```
DateTimeFormat='\''yyyy-mm-dd hh:mm:ss.fff\'''
```

PowerBuilder parses the backslash followed by two single quotes (\'') as a single quote.

Examples

Assume you are updating a table named Files by setting the Timestamp column to 4/2/06 3:45 pm. This DateTime is represented by the following DateTime format:

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

To specify that PowerBuilder 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 page in the Database Profile Setup dialog box:

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

- **Application**   Type the following in code:

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

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

```
UPDATE FILES
SET TIMESTAMP='4/2/06 3:45 pm'
```

See also

DateFormat
TimeFormat

DBConfigSection

Description

Specifies the section in a .NET configuration file to be used to specify custom configuration settings.

When to specify DBConfigSection
You must specify a value for DBConfigSection *before* connecting to the database.

Applies to

ADO.NET

Syntax

DBConfigSection='*value*'

Parameter	Description
value	A string that specifies the section in a .NET configuration file to be used to specify DBParm values and the syntax used to obtain the value of an identity column. The value is the name of a section you create in a .NET configuration file.

Default value

None.

Usage

You can use the standard `select @@identity` syntax to obtain the value of an identity column. You can also use an alternative syntax, such as `select scope_identity()`, by adding sections to a .NET configuration file for your application.

The configuration file resides in the same directory as the application and has the same name as the application with the extension *.config*. It can contain multiple custom configuration sections. Each has two attributes: dbParm and getIdentity. You can set either or both of these attributes. For Web applications, add the settings to the *web.config* file.

The dbParm value sets the value of the DBParm parameter of the transaction object. It has a maximum length of 1000 characters. If you set a value for a parameter in the configuration file, any value that you set in code or in the Database Profile Setup dialog box is overridden.

The getIdentity value specifies the syntax used to retrieve the value of an identity column. It has a maximum length of 100 characters. If you do not specify a value for getIdentity, the `select @@identity` syntax is used.

For more information about creating the configuration file, see the chapter on ADO.NET in *Connecting to Your Database*.

Examples

To specify that your PowerBuilder application uses the custom configuration file called myconfig1:

- **Database profile** Specify myconfig1 in the ConfigSection Name in Configuration File box on the System tabpage in the Database Profile Setup dialog box.
- **Application** Type the following in code:

```
SQLCA.DBParm="DBConfigSection='myconfig1'"
```

This sample configuration file for PowerBuilder is called *pb125.exe.config*. It contains three custom configurations. The <myconfig> element sets both the dbParm and getIdentity attributes. <myconfig1> sets getIdentity only, and <myconfig2> sets dbParm only. The <runtime> section is in the configuration file that ships with PowerBuilder but would not be included in the configuration file that you ship with your application, which would have the same name as your application with the extension *exe.config*. For .NET Web Service targets, you add the custom configurations to the *web.config* file. The <configSections> section should be added before any other application settings.

```
<configuration>
   <configSections>
     <sectionGroup name="dbConfiguration">
        <section name="myconfig"
         type="Sybase.PowerBuilderDataWindow.Db.DbConf
iguration,
         Sybase.PowerBuilderDataWindow.Db"
        />
        <section name="myconfig1"
         type="Sybase.PowerBuilderDataWindow.Db.DbConf
iguration,
         Sybase.PowerBuilderDataWindow.Db"
        />
        <section name="myconfig2"
         type="Sybase.PowerBuilderDataWindow.Db.DbConf
iguration,
         Sybase.PowerBuilderDataWindow.Db"
        />
      </sectionGroup>
   </configSections>

<runtime>
      <assemblyBinding xmlns=
       "urn:schemas-microsoft-com:asm.v1">
         <dependentAssembly>
            <assemblyIdentity name=
            "Sybase.PowerBuilder.Db"/>
            <codeBase href="file:///C:/Program Files/
             Sybase/PowerBuilder 11.0/DotNET/bin/
             Sybase.PowerBuilder.Db.dll"/>
         </dependentAssembly>
         <dependentAssembly>
           <assemblyIdentity name=
            "Sybase.PowerBuilder.WebService.WSDL"/>
           <codeBase href="file:///C:/Program Files/
            Sybase/PowerBuilder 11.0/DotNET/bin/
            Sybase.PowerBuilder.WebService.WSDL.dll"/>
         </dependentAssembly>
         <dependentAssembly>
           <assemblyIdentity name=
            "Sybase.PowerBuilder.WebService.Runtime"/>
            <codeBase href="file:///C:/Program Files/
             Sybase/PowerBuilder 11.0/DotNET/bin/
             Sybase.PowerBuilder.WebService.
             Runtime.dll"/>
         </dependentAssembly>
         <probing privatePath="DotNET/bin" />
      </assemblyBinding>
   </runtime>

   <dbConfiguration>
      <myconfig dbParm="disablebind=0"
       getIdentity="select scope_identity()"
```

```
            />
            <myconfig1 getIdentity="select scope_identity()"
            />
            <myconfig2 =
             "Namespace='Oracle.DataAccess.Client',
             DataSource='ora10gen',DisableBind=1,
             NCharBind=1,ADORelease='10.1.0.301'"
            />
         </dbConfiguration>
      </configuration>
```

DBGetTime

Description

Specifies the number of seconds PowerBuilder 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), PowerBuilder 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

ASE, SYC Sybase Adaptive Server Enterprise
DIR Sybase DirectConnect
I10 Informix
IN9 Informix
JDB JDBC
ODBC (if driver and back-end DBMS support this feature)
O90 Oracle9*i*
O10 Oracle 10*g*
ORA Oracle 11*g*

Syntax

DBGetTime=*value*

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

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 or report.

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 page in the Database Profile Setup dialog box.
- **Application** Type the following in code:

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

See also

Async

Db_Locale

Description

Db_Locale identifies the locale of the data in the database.

When to specify Db_Locale
You must specify the Client_Locale parameter *before* connecting to a database.

Applies to

I10 Informix

Syntax

Db_Locale='*language_territory.codeset*'

Parameter	Description
language	Two character name that represents the language for a specific locale, for example “en” for English.
territory	Two character name that represents the cultural conventions for a specific territory, for example “AU” for Australia.
codeset	Name of the code set that the locale supports, for example “utf8.”

Default value

Client_Locale value.

Usage

The I10 native interface uses the Informix GLS (Global Language Support) API for global language support. Db_Locale specifies the value of the Informix environment variable DB_LOCALE. If you do not set the DBParm, the default Db_Locale value is the Client_Locale value.

Db_Locale specifies the language, territory, and code set that the database server needs to correctly interpret locale-sensitive datatypes such as NChar and NVarChar in a specific database. The code set specified in DB_LOCALE determines which characters are valid in any character column, as well as in the names of database objects such as databases, tables, columns, and views.

For more information about the Informix CLIENT_LOCALE and DB_LOCALE environment variables, see the *IBM Informix GLS User's Guide*, currently available at the Informix library Web site at http://publib.boulder.ibm.com/infocenter/idshelp/v111/index.jsp?topic=/com.ibm.glsug.doc/glsug.htm.

Examples

To set the database locale to 'en_us.utf8':

- **Database profile** Type the following in the Database Locale box on the Regional Settings page in the Database Profile Setup dialog box:

  ```
  en_us.utf8
  ```

- **Application** Type the following in code:

  ```
  SQLCA.DBParm="Db_Locale='en_us.utf8'"
  ```

See also

Client_Locale
Locale

DBTextLimit

Description

Specifies the maximum length of a text field that is returned when you include the text field in a SQL SELECT statement.

You can set the DBTextLimit parameter if you want to include a long text string in a DataWindow object without treating the text as a binary large object (blob) datatype.

Applies to

ASE, SYC, and SYJ Sybase Adaptive Server Enterprise
DIR Sybase DirectConnect
SNC SQL Native Client for Microsoft SQL Server

Syntax

DBTextLimit='*value*'

Parameter	Description
value	The maximum length in bytes of a text field that is returned when you include the text field in a SQL SELECT statement. The range of valid values is from 0 bytes to 2 GB. When you set DBTextLimit to 0, the server returns the maximum length text field.

Default value

The default value for DBTextLimit is the default specified for the DBTEXTLIMIT DB-Library or CS_TEXTLIMIT CT-Library connection property.

Usage

The text field length that DB-Library or CT-Library returns is the lesser of the DBTextLimit value and the setting for the global variable TEXTSIZE.

If the setting for TEXTSIZE is less than the value you specify for DBTextLimit, DB-Library or CT-Library returns the TEXTSIZE value.

Examples

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

- **Database profile** Type `32000` in the Text Limit in SQL box (when using the ASE, or SYC, or SYJ interface), or Maximum Length of Long VarChar box (when using the DirectConnect interface) on the Syntax page in the Database Profile Setup dialog box.
- **Application** Type the following in code:

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

DecimalSeparator

Description

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

Applies to

ADO.NET
DIR Sybase DirectConnect
ODBC (if driver and back-end DBMS support this feature)
OLE DB
O90 Oracle9*i*
O10 Oracle 10*g*
ORA Oracle 11*g*
SNC SQL Native Client for Microsoft SQL Server

Syntax

DecimalSeparator='*value*'

Parameter	Description
value	The decimal separator setting used by the back-end DBMS that you are accessing in PowerBuilder. 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 (,), PowerBuilder 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 parameter currently supports period (.) and comma (,) as valid values. Therefore, if the decimal separator setting for your DBMS is a comma, you should set the DecimalSeparator parameter to `','` (comma) to make sure PowerBuilder correctly handles numeric strings returned from your database.

Example using Oracle Assume you are accessing an Oracle database in PowerBuilder and the decimal separator setting is a comma (,). Oracle returns to PowerBuilder the numeric string '123,50' containing a comma instead of a period as the decimal separator. PowerBuilder then sends this string to its decimal conversion routines.

By default, the PowerBuilder decimal conversion routines expect a period as the decimal separator. If you set the DecimalSeparator parameter to `','` (comma), PowerBuilder correctly handles this string and returns it as `'`123,50`'`.

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 page in the Database Profile Setup dialog box.
- **Application** Type the following in code:

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

See also

NumericFormat

DefaultProcOwner

Description

The DefaultProcOwner parameter lets you set a default owner for a stored procedure. The parameter takes effect only when the stored procedure is not qualified. For ODBC, the PBNewSPInvocation parameter must also be set.

Applies to

ADO.NET
ODBC

Syntax

DefaultProcOwner='*value*'

Parameter	Description
value	A string specifying the name of the default owner of the stored procedure

Usage

The parameter can be set dynamically at runtime after connecting to a database. You can also set it in your *PBODB125.INI* file if you want to create and retrieve data into a DataWindow with a stored procedure data source in the DataWindow painter. The runtime setting overrides the setting in *PBODB125.INI*.

You can also cancel the setting at runtime. If you do so, PowerBuilder uses the current user as the owner of a non-qualified stored procedure when it obtains the parameters of the stored procedure.

Examples

To set the default owner to `proms` in *PBODB125.INI*:

```
[SQL Anywhere]
DefaultProcOwner='proms'
```

In code:

```
SQLCA.DBParm="DefaultProcOwner='proms'";
```

To cancel the setting:

```
 SQLCA.DBParm="DefaultProcOwner=''";
```

Note that the single quotes in the previous example contain an empty string, not a space.

See also

PBNewSPInvocation

DelimitIdentifier

Description

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

Applies to

ADO.NET
ASE, SYC Sybase Adaptive Server Enterprise
DIR Sybase DirectConnect
I10 Informix
IN9 Informix
JDB JDBC
ODBC (if driver and back-end DBMS support this feature)
OLE DB
O90 Oracle9*i*
O10 Oracle 10*g*
ORA Oracle 11*g*
SNC SQL Native Client for Microsoft SQL Server

Syntax

DelimitIdentifier='*value*'

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

Default value

The default value for the DelimitIdentifier parameter depends on the DBMS you are accessing, as follows:

DBMS	DelimitIdentifer default value
ADO.NET	DelimitIdentifier='Yes' for SQL Anywhere, Oracle, and DB2, 'No' for all other DBMSs.
Informix	DelimitIdentifier='No'
JDBC	Depends on the DelimitIdentifer setting in the registry
ODBC	Depends on the DelimitIdentifer setting in the PBODB125 initialization file
Oracle	DelimitIdentifier='Yes'
OLE DB	Depends on the DelimitIdentifer setting in the PBODB125 initialization file
SNC	DelimitIdentifier='No'

DBMS	DelimitIdentifer default value
Sybase DirectConnect	DelimitIdentifier='No'
Sybase Adaptive Server Enterprise	DelimitIdentifier='No'

Usage

Informix Informix database servers can create a log of database transactions in either ASCII or non-ASCII format. If the database is creating a non-ASCII log, the setting of the DelimitIdentifier is optional. If the database is creating an ASCII log, you must set DelimitIdentifier='Yes' to make the SQL syntax generated by PowerBuilder behave as expected.

Sybase Adaptive Server Enterprise When you set DelimitIdentifier to 'Yes', the “set quoted_identifier on” command is automatically sent to Adaptive Server to adjust your database connection on the server. Otherwise, the “set quoted_identifier off” command is sent to the server. This feature occurs with ASE, SYC, JDBC, ODBC, and OLE DB interfaces.

Microsoft SQL Server When you set DelimitIdentifier to 'Yes', the “set quoted_identifier on” command is automatically sent to Microsoft SQL Server to adjust your database connection on the server when you use ODBC or OLE DB. Otherwise, the “set quoted_identifier off” command is sent to the server.

Sending “set quoted_identifier off” to the server can cause some SQL commands to fail if the SQL code contains quotation marks. To prevent PowerBuilder from sending this instruction to the server, set the DelimitIdentifierToDB parameter to ‘No’ in the PBODB initialization file or the connection string.

JDBC and ODBC The DelimitIdentifier parameter setting overrides the DelimitIdentifier setting specified for your JDBC driver in the registry and for your ODBC driver in the PBODB125 initialization file.

DirectConnect If you want to use mixed-case identifier names, you must set DelimitIdentifier=‘Yes’. Also, you must set LowerCaseIdent=‘No’ to preserve case sensitivity of identifiers stored in the DB2 system catalog.

Examples

To specify that PowerBuilder 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 page in the Database Profile Setup dialog box.
- **Application** Type the following in code:

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

See also

DelimitIdentifierToDB

LowerCaseIdent

DelimitIdentifierToDB

Description

Specifies whether PowerBuilder should send a “set quoted_identifier off” instruction to the server when the DelimitIdentifier parameter is not set.

Applies to

ODBC (if driver and back-end DBMS support this feature)
OLE DB

Syntax

DelimitIdentifierToDB='*value*'

Parameter	Description
value	Specifies whether PowerBuilder sends a “set quoted_identifier off” instruction to the server when the DelimitIdentifier parameter is not set. Values are: • **Yes** Send set quoted_identifier off to the server • **No** Do not send set quoted_identifier off to the server

Default value

‘No’

Usage

When you set DelimitIdentifier to ‘Yes’, the “set quoted_identifier on” command is automatically sent to the database server to adjust your database connection on the server when you use ODBC or OLE DB. Otherwise, the “set quoted_identifier off” command is sent to the server.

Sending “set quoted_identifier off” to the server can cause some SQL commands to fail if the SQL code contains quotation marks. To prevent PowerBuilder from sending this instruction to the server, set the DelimitIdentifierToDB parameter to ‘No’ in the PBODB initialization file or the connection string.

Examples

To specify that PowerBuilder should not send a set quoted_identifier instruction to the server, add the following to the appropriate section of your PBODB initialization file or your connection string:

```
"DelimitIdentifierToDB='No'"
```

See also

DelimitIdentifier

DisableBind

Description

For those DBMSs that support bind variables, PowerBuilder can bind input parameters to a compiled SQL statement by default. The DisableBind parameter allows you to specify whether you want to disable this binding.

When you set DisableBind to 1 to disable the binding, PowerBuilder replaces the input variable with the value entered by the application user or specified in code.

Applies to

ADO.NET
ASE, SYC Sybase Adaptive Server Enterprise
I10 Informix
IN9 Informix
JDB JDBC
ODBC (if driver and back-end DBMS support this feature)
OLE DB
O90 Oracle9*i*
O10 Oracle 10*g*
ORA Oracle 11*g*
SNC SQL Native Client for Microsoft SQL Server

Syntax

DisableBind=*value*

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

Default value

DisableBind=1 for ADO.NET, ASE, SYC, SNC, and OLE DB, DisableBind=0 for other interfaces

Usage

Bind variables In a SQL statement, a **bind variable** is a placeholder for a column value. By default, PowerBuilder 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 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 to enable binding (the default for all database interfaces except ADO.NET, ASE, SYC, SNC, and OLE DB), PowerBuilder 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 PowerBuilder. (For more about using cached statements, see the description of the SQLCache parameter.)

Performance improvements For SQL Anywhere and Oracle databases, bind variables improve performance by allowing PowerBuilder 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 object. 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 PowerBuilder should disable the binding of input parameters to a compiled SQL statement:

- **Database profile** Select the Disable Bind check box on the Transaction or System page in the Database Profile Setup dialog box.
- **Application** Type the following in code:

```
SQLCA.DBParm="DisableBind=1"
```

See also

SQLCache

Driver

Description

The JDBC driver your application uses to connect to the database.

When to specify Driver
You must specify the Driver database parameter *before* connecting to the database.

Applies to

JDB JDBC

Syntax

Driver='*driver_name*'

Default value

None

Usage

The driver name identifies the Java class name for the particular driver you are using to connect to the database.

Examples

Example 1 To set the driver name of a Sybase jConnect driver:

- **Database profile** Type the following in the Driver Name box on the Connection page in the Database Profile Setup dialog box:

```
com.sybase.jdbc.SybDriver
```

- **Application** Type the following in code:

```
SQLCA.DBParm="Driver='com.sybase.jdbc.SybDriver'"
```

Example 2 To set the driver name of an Oracle JDBC Driver:

- **Database profile** Type the following in the Driver Name box on the Connection page in the Database Profile Setup dialog box.

```
oracle.jdbc.driver.OracleDriver
```

- **Application** Type the following in code:

```
SQLCA.DBParm="Driver='oracle.jdbc.driver.OracleDriv
er'"
```

See also URL

DS_Alias

Description

When you access a Sybase Adaptive Server Enterprise database in PowerBuilder through Open Client, DS_Alias is one of several parameters that you can set to enable network-based directory services in your application. (For other directory services parameters, see the See Also section.)

Some directory service providers and drivers support the creation of alias entries. An **alias entry** provides a link to a primary directory entry in a hierarchy, thereby giving users multiple ways to access the primary entry while searching the directory structure for a particular network entity.

For those directory service providers and drivers that support aliases, DS_Alias specifies whether the provider is allowed to follow links for (expand) alias entries while searching the directory hierarchy. The default behavior is to allow expansion of alias entries for providers that support this feature.

You must specify a value for DS_Alias *before* connecting to the database in PowerBuilder.

Using third-party directory service providers
For information about the third-party directory service providers and operating system platforms that Sybase has tested with Open Client directory services, see the Open Client documentation.

Applies to ASE, SYC Sybase Adaptive Server Enterprise

Syntax DS_Alias=*value*

Parameter	Description
value	For those directory services providers and drivers that support aliases, specifies whether the provider is allowed to expand alias entries while searching a directory hierarchy. Values are: • **0** Prohibit provider from expanding alias entries during a directory search. You can also specify `'No'` or `'False'` to set this value. • **1** (Default) Allow provider to expand alias entries during a directory search. You can also specify `'Yes'` or `'True'` to set this value.

Default value DS_Alias=1

Usage

When to use To prevent access to your data through directory alias entries, set DS_Alias to 0. This prohibits directory service providers that support aliases from expanding alias entries during a directory search.

Set Release parameter For this parameter to take effect, you *must* also set the Release parameter to 11or higher to specify that your application should use the appropriate version of Sybase Open Client Client-Library (CT-Lib) behavior. See the description of the Release parameter for more information.

Requirements for use To use DS_Alias or any other parameter supporting Open Client directory services, you must meet certain requirements for using directory services in your PowerBuilder application. For details, see "Requirements for using Open Client directory services" in *Connecting to Your Database*.

Corresponding CT-Lib connection property Specifying a value for DS_Alias sets the corresponding Sybase CT-Lib connection property named CS_DS_EXPANDALIAS.

Examples

To prohibit directory service providers that support aliases from expanding alias entries during a directory search:

- **Database profile** Clear the Directory Alias Entries check box on the Directory Services page in the Database Profile Setup dialog box.
- **Application** Type the following in code:

```
SQLCA.DBParm="DS_Alias=0"
```

See also

DS_Copy
DS_DitBase
DS_Failover
DS_Password
DS_Principal
DS_Provider
DS_TimeLimit
Release

DS_Copy

Description

When you access a Sybase Adaptive Server Enterprise database in PowerBuilder through Open Client, DS_Copy is one of several parameters that you can set to enable network-based directory services in your application. (For other directory services parameters, see the See Also section.)

Some directory service providers and drivers support the use of caching. **Caching** allows a directory service provider to use cached information while searching a directory instead of making a request to the directory server agent for information.

For those directory service providers and drives that support caching, DS_Copy specifies whether the provider is allowed to use cached information during a directory search. The default behavior is to allow providers that support this feature to use cached information.

You must specify a value for DS_Copy *before* connecting to the database in PowerBuilder.

Using third-party directory service providers
For information about the third-party directory service providers and operating system platforms that Sybase has tested with Open Client directory services, see the Open Client documentation.

Applies to

ASE, SYC Sybase Adaptive Server Enterprise

Syntax

DS_Copy=*value*

Parameter	Description
value	For those directory services providers and drivers that support caching, specifies whether the provider is allowed to use cached information when making a directory search. Values are: • **0** Prohibit provider from using cached information during a directory search. You can also specify `'No'` or `'False'` to set this value. • **1** (Default) Allow provider to use cached information when making a directory search. You can also specify `'Yes'` or `'True'` to set this value.

Default value

DS_Copy=1

Usage

When to use Allowing providers to use cached information during directory searches makes the searches faster, but does not ensure that the provider is using the most up-to-date directory information.

To ensure that the application gets the most recent changes to directory entries when it requests directory information, set DS_Copy to 0 to prohibit providers that support caching from using cached information during a directory search.

Set Release parameter For this parameter to take effect, you *must* also set the Release parameter to 11or higher to specify that your application should use the appropriate version of Sybase Open Client Client-Library (CT-Lib) behavior. See the description of the Release parameter for more information.

Requirements for use To use DS_Copy or any other parameter supporting Open Client directory services, you must meet certain requirements for using directory services in your PowerBuilder application. For details, see “Requirements for using Open Client directory services” in *Connecting to Your Database*.

Corresponding CT-Lib connection property Specifying a value for DS_Copy sets the corresponding Sybase CT-Lib connection property named CS_DS_COPY.

Examples

To prohibit directory service providers that support caching from using cached information during a directory search:

- **Database profile** Clear the Use Caching check box on the Directory Services page in the Database Profile Setup dialog box.
- **Application** Type the following in code:

```
SQLCA.DBParm="DS_Copy=0"
```

See also

DS_Alias
DS_DitBase
DS_Failover
DS_Password
DS_Principal
DS_Provider
DS_TimeLimit
Release

DS_DitBase

Description

When you access a Sybase Adaptive Server Enterprise database in PowerBuilder through Open Client, DS_DitBase is one of several parameters that you can set to enable network-based directory services in your application. (For other directory services parameters, see the See Also section.)

When you use Open Client directory services, a default (active) directory information tree base (DIT base) is specified in the Open Client/Server™ Configuration utility. The **DIT base** is the directory node where directory searches start. This is analogous to the current working directory in MS-DOS file systems.

DS_DitBase lets you specify the name of the directory node where you want searches for directory entries to start. The DS_DitBase value you specify must be a fully qualified name that uses the syntax required by your directory service provider and driver (see the Examples section for illustrations).

The default value for DS_DitBase is the DIT base currently specified as active in the Open Client/Open Server Configuration utility.

You must specify a value for DS_DitBase *before* connecting to the database in PowerBuilder.

Using third-party directory service providers
For information about the third-party directory service providers and operating system platforms that Sybase has tested with Open Client directory services, see the Open Client documentation.

Applies to

ASE, SYC Sybase Adaptive Server Enterprise

Syntax

DS_DitBase='*dit_base*'

Parameter	Description
dit_base	The name of the directory node where you want directory searches to start. By default, this is the DIT base currently specified as active in the Open Client/Open Server Configuration utility. The value for *dit_base* must be a fully qualified name that uses the syntax required by your directory service provider and driver. The syntax for specifying the DIT base varies for different providers; see your provider’s documentation for details. For examples of how to specify *dit_base* for different directory service providers, see the Examples section.

Default value

The default value for DS_DitBase is the DIT base currently specified as active in the Open Client/Open Server Configuration utility.

Usage

When to use Set DS_DitBase to specify a starting node for directory searches *other than* the DIT base node specified as active in the Open Client/Open Server Configuration utility. For instructions on using the Open Client/Open Server Configuration utility, see your Sybase Open Client/Server configuration guide.

Set Release parameter For this parameter to take effect, you *must* also set the Release parameter to 11or higher to specify that your application should use the appropriate version of Sybase Open Client Client-Library (CT-Lib) behavior. See the description of the Release parameter for more information.

Requirements for use To use DS_DitBase or any other parameter supporting Open Client directory services, you must meet certain requirements for using directory services in your PowerBuilder application. For details, see “Requirements for using Open Client directory services” in *Connecting to Your Database*.

Corresponding CT-Lib connection property Specifying a value for DS_DitBase sets the corresponding Sybase CT-Lib connection property named CS_DS_DITBASE.

Examples

About these examples The examples that follow show how to specify a DS_DitBase value for different directory service providers.

See your directory service provider’s documentation for complete information about the format your provider requires for specifying the DIT base.

Example 1 (Windows NT Registry) This example shows the syntax for DS_DitBase if your directory service provider is the Windows NT Registry:

Node name: SALES:software\sybase\server\SYS11NT
DS_DitBase: SALES:software\sybase\server

To set DS_DitBase:

- **Database profile** Type the following in the DIT Base box on the Directory Services page in the Database Profile Setup dialog box. Do *not* end the DS_DitBase value with a backslash (\):

  ```
  SALES:software\sybase\server
  ```

- **Application** Type the following in code. Do *not* end the DS_DitBase value with a backslash (\):

  ```
  SQLCA.DBParm =
     "DS_DitBase='SALES:software\sybase\server'"
  ```

Example 2 (DCE/CDS) This example shows the syntax for DS_DitBase if your directory service provider is Distributed Computing Environment Cell Directory Services (DCE/CDS):

Node name: /.../boston.sales/dataservers/sybase/SYS11
DS_DitBase: /.../boston.sales/dataservers

To set DS_DitBase:

- **Database profile** Type the following in the DIT Base box on the Directory Services page in the Database Profile Setup dialog box. Do *not* end the DS_DitBase value with a slash (/):

  ```
  /.../boston.sales/dataservers
  ```

- **Application** Type the following in code. Do *not* end the DS_DitBase value with a slash (/):

  ```
  SQLCA.DBParm =
  "DS_DitBase='/.../boston.sales/dataservers'"
  ```

Example 3 (Novell NDS) This example shows the syntax for DS_DitBase if your directory service provider is Novell NetWare Directory Services (NDS):

Node name: CN=SYS11.OU=miami.OU=sales.O=sybase
DS_DitBase: OU=miami.OU=sales.O=sybase

To set DS_DitBase:

- **Database profile** Type the following in the DIT Base box on the Directory Services page in the Database Profile Setup dialog box:

  ```
  OU=miami.OU=sales.O=sybase
  ```

- **Application** To specify DS_DitBase in code, type the following:

  ```
  SQLCA.DBParm =
  "DS_DitBase='OU=miami.OU=sales.O=sybase'"
  ```

See also

DS_Alias
DS_Copy
DS_Failover
DS_Password
DS_Principal
DS_Provider
DS_TimeLimit
Release

DS_Failover

Description

When you access a Sybase Adaptive Server Enterprise database in PowerBuilder through Open Client, DS_Failover is one of several parameters that you can set to enable network-based directory services in your application. (For other directory services parameters, see the See Also section.)

Sybase Open Client Client-Library (CT-Lib) requires a directory to map logical server names to network addresses. The source for this directory can be either the Sybase Interfaces file or a network-based directory service provider (such as DCE Cell Directory Services or the Windows Registry).

If you want an application to use a directory source *other than* the Interfaces file, CT-Lib must be able to load the appropriate directory driver. If CT-Lib cannot load the required driver, you can set DS_Failover to specify whether CT-Lib should silently default (fail over) to using the Interfaces file as the directory source.

By default, DS_Failover specifies that CT-Lib should use the Interfaces file as the directory source if it cannot load the requested directory driver.

You must specify a value for DS_Failover *before* connecting to the database in PowerBuilder.

Using third-party directory service providers
For information about the third-party directory service providers and operating system platforms that Sybase has tested with Open Client directory services, see the Open Client documentation.

Applies to

ASE, SYC Sybase Adaptive Server Enterprise

Syntax

DS_Failover=*value*

Parameter	Description
value	Specifies whether Sybase CT-Lib should silently default (fail over) to using the Interfaces file as the directory source if it cannot load the requested directory driver. Values are: • **0** Prohibit CT-Lib from using the Interfaces file as the directory source if it cannot load the requested directory driver. You can also specify `'No'` or `'False'` to set this value. • **1** (Default) Allow CT-Lib to use the Interfaces file as the directory source if it cannot load the requested directory driver. You can also specify `'Yes'` or `'True'` to set this value.

Default value

DS_Failover=1

Usage

When to use To prevent CT-Lib from using the Interfaces file as the directory source if it cannot load the requested directory driver, set DS_Failover to 0.

If DS_Failover is set to 0 to prevent use of the Interfaces file and CT-Lib cannot load the requested directory driver, the connection's directory source is undefined. This causes certain operations requiring directory access to fail.

Set Release parameter For this parameter to take effect, you *must* also set the Release parameter to 11or higher to specify that your application should use the appropriate version of Sybase Open Client Client-Library (CT-Lib) behavior. See the description of the Release parameter for more information.

Requirements for use To use DS_Failover or any other parameter supporting Open Client directory services, you must meet certain requirements for using directory services in your PowerBuilder application. For details, see “Requirements for using Open Client directory services” in *Connecting to Your Database*.

Corresponding CT-Lib connection property Specifying a value for DS_Failover sets the corresponding Sybase CT-Lib connection property named CS_DS_FAILOVER.

Examples

To prohibit CT-Lib from using the Interfaces file as the directory source if it cannot load the requested directory driver:

- **Database profile** Clear the Enable Failover check box on the Directory Services page in the Database Profile Setup dialog box.
- **Application** Type the following in code:

```
SQLCA.DBParm="DS_Failover=0"
```

See also

DS_Alias
DS_Copy
DS_DitBase
DS_Password
DS_Principal
DS_Provider
DS_TimeLimit
Release

DS_Password

Description

When you access a Sybase Adaptive Server Enterprise database in PowerBuilder through Open Client 12.5 or higher software, DS_Password is one of several parameters that you can set to enable network-based directory services in your application. (For other directory services parameters, see the See Also section.)

Some directory service providers and drivers require an authenticated principal (user ID) name and password to control an application's access to directory entries. For those providers and drivers, DS_Principal and DS_Password specify the principal name and password your application should use to identify you to the directory service provider.

You must specify a value for DS_Password *before* connecting to the database in PowerBuilder.

Applies to

ASE, SYC Sybase Adaptive Server Enterprise

Syntax

DS_Password='*password*'

Parameter	Description
password	The password associated with the principal (user ID) name you specified in the DS_Principal parameter.

Default value

None

PowerBuilder does not set DS_Password or the corresponding Sybase Open Client Client-Library (CT-Lib) connection parameter CS_DS_PASSWORD if you do not specify a value.

Usage

When to use If your directory service provider requires an authenticated principal name for directory access, set DS_Password to the password that goes with your directory service principal name.

Set Release parameter For this parameter to take effect, you *must* also set the Release parameter to 11or higher to specify that your application should use the appropriate version of Sybase Open Client Client-Library (CT-Lib) behavior. See the description of the Release parameter for more information.

Requirements for use To use DS_Password or any other parameter supporting Open Client 12.5 directory services, you must meet certain requirements for using directory services in your PowerBuilder application. For details, see “Requirements for using Open Client directory services” in *Connecting to Your Database*.

Corresponding CT-Lib connection property Specifying a value for DS_Password sets the corresponding Sybase CT-Lib connection property named CS_DS_PASSWORD.

Examples

To specify MYPASS as your application's password:

- **Database profile** Type the following in the Password box on the Directory Services page in the Database Profile Setup dialog box:

```
MYPASS
```

- **Application** Type the following in code:

```
SQLCA.DBParm="DS_Password='MYPASS'"
```

To specify MYPASS as your application's password, type `MYPASS` in the Password box on the Directory Services page in the Database Profile Setup dialog box.

See also

DS_Alias
DS_Copy
DS_DitBase
DS_Failover
DS_Principal
DS_Provider
DS_TimeLimit
Release

DS_Principal

Description

When you access a Sybase Adaptive Server Enterprise database in PowerBuilder through Open Client, DS_Principal is one of several parameters that you can set to enable network-based directory services in your application. (For other directory services parameters, see the See Also section.)

Some directory service providers and drivers require an authenticated principal (user ID) name to control an application's access to directory entries. For those providers and drivers, DS_Principal and DS_Password specify the principal name and password your application should use to identify you to the directory service provider.

You must specify a value for DS_Principal *before* connecting to the database in PowerBuilder.

Using third-party directory service providers
For information about the third-party directory service providers and operating system platforms that Sybase has tested with Open Client directory services, see the Open Client documentation.

Applies to

ASE, SYC Sybase Adaptive Server Enterprise

Syntax

DS_Principal='*principal_name*'

Parameter	Description
principal_name	The principal (user ID) name your application should use to identify you to the directory service provider.

Default value

None

PowerBuilder does not set DS_Principal or the corresponding Sybase Open Client Client-Library (CT-Lib) connection parameter CS_DS_PRINCIPAL if you do not specify a value.

Usage

When to use If your directory service provider requires an authenticated principal name for directory access, set DS_Principal to the principal (user ID) name that goes with your directory service password.

Set Release parameter For this parameter to take effect, you *must* also set the Release parameter to 11or higher to specify that your application should use the appropriate version of Sybase Open Client Client-Library (CT-Lib) behavior. See the description of the Release parameter for more information.

Requirements for use To use DS_Principal or any other parameter supporting Open Client directory services, you must meet certain requirements for using directory services in your PowerBuilder application. For details, see “Requirements for using Open Client directory services” in *Connecting to Your Database*.

Corresponding CT-Lib connection property Specifying a value for DS_Principal sets the corresponding Sybase CT-Lib connection property named CS_DS_PRINCIPAL.

Examples

To specify JSMITH as your application's principal name:

- **Database profile** Type the following in the Principal Name box on the Directory Services page in the Database Profile Setup dialog box:

  ```
  JSMITH
  ```

- **Application** Type the following in code:

  ```
  SQLCA.DBParm="DS_Principal='JSMITH'"
  ```

See also

DS_Alias
DS_Copy
DS_DitBase
DS_Failover
DS_Password
DS_Provider
DS_TimeLimit
Release

DS_Provider

Description

When you access a Sybase Adaptive Server Enterprise database in PowerBuilder through Open Client, DS_Provider is one of several parameters that you can set to enable network-based directory services in your application. (For other directory services parameters, see the See Also section.)

When you use Open Client directory services, you must specify your directory service provider names in the Open Client/Open Server Configuration utility so that the required drivers can be loaded for each provider. The default directory service provider is the one currently specified as active in the Configuration utility.

DS_Provider lets you specify a directory service provider name listed in the Open Client/Open Server Configuration utility *other than* the default (active) provider. The default value for DS_Provider is the provider name currently specified as active in the Configuration utility.

You must specify a value for DS_Provider *before* connecting to the database in PowerBuilder.

Using third-party directory service providers
For information about the third-party directory service providers and operating system platforms that Sybase has tested with Open Client directory services, see the Open Client documentation.

Applies to

ASE, SYC Sybase Adaptive Server Enterprise

Syntax

DS_Provider='*provider_name*'

Parameter	Description
provider_name	The directory service provider name you want to use for directory services. The provider name is case sensitive. You must specify it *exactly as it appears* in the Open Client/Open Server Configuration utility.

Default value

The default value for DS_Provider is the provider name currently specified as active in the Open Client/Open Server Configuration utility.

Usage

When to use Set DS_Provider to use a directory service provider specified in the Open Client/Open Server Configuration utility *other than* the default (active) provider. For instructions on using the Open Client/Open Server Configuration utility, see your Sybase Open Client/Server configuration guide.

Set Release parameter For this parameter to take effect, you *must* also set the Release parameter to 11or higher to specify that your application should use the appropriate version of Sybase Open Client Client-Library (CT-Lib) behavior. See the description of the Release parameter for more information.

Requirements for use To use DS_Provider or any other parameter supporting Open Client directory services, you must meet certain requirements for using directory services in your PowerBuilder application. For details, see “Requirements for using Open Client directory services” in *Connecting to Your Database*.

Corresponding CT-Lib connection property Specifying a value for DS_Provider sets the corresponding Sybase CT-Lib connection property named CS_DS_PROVIDER.

Examples

To specify NTREGISTRY as the directory service provider name:

- **Database profile** Type the following in the Provider box on the Directory Services page in the Database Profile Setup dialog box:

```
NTREGISTRY
```

- **Application** Type the following in code:

```
SQLCA.DBParm="DS_Provider='NTREGISTRY'"
```

To specify NTREGISTRY as the directory service provider name, type `NTREGISTRY` in the Provider box on the Directory Services page in the Database Profile Setup dialog box.

See also

DS_Alias
DS_Copy
DS_DitBase

DS_Failover
DS_Password
DS_Principal
DS_TimeLimit
Release

DS_TimeLimit

Description

When you access a Sybase Adaptive Server Enterprise database in PowerBuilder through Open Client, DS_TimeLimit is one of several parameters that you can set to enable network-based directory services in your application. (For other directory services parameters, see the See Also section.)

Some directory service providers and drivers support the use of time limits for a directory search. For those providers and drivers, DS_TimeLimit specifies the maximum number of seconds that a directory search lasts.

By default, DS_TimeLimit specifies that there is no time limit for a directory search.

You must specify a value for DS_TimeLimit *before* connecting to the database in PowerBuilder.

Using third-party directory service providers
For information about the third-party directory service providers and operating system platforms that Sybase has tested with Open Client directory services, see the Open Client documentation.

Applies to

ASE, SYC Sybase Adaptive Server Enterprise

Syntax

DS_TimeLimit='*value*'

Parameter	Description
value	Specifies the maximum number of seconds that you want a directory search to last. You can also specify `'no_limit'` (the default) to indicate that there is no time limit for the directory search. If the specified time limit expires and the target has not been found, the directory search is unsuccessful and the PowerBuilder connection fails.

Default value

DS_TimeLimit='no_limit'

Usage

Set Release parameter For DS_TimeLimit to take effect, you *must* also set the Release parameter to 11or higher to specify that your application should use the appropriate version of Sybase Open Client Client-Library (CT-Lib) behavior. See the description of the Release parameter for more information.

Requirements for use To use DS_TimeLimit or any other parameter supporting Open Client directory services, you must meet certain requirements for using directory services in your PowerBuilder application. For details, see "Requirements for using Open Client directory services" in *Connecting to Your Database*.

Corresponding CT-Lib connection property Specifying a value for DS_TimeLimit sets the corresponding Sybase CT-Lib connection property named CS_DS_TIMELIMIT.

Examples

To specify that you want the directory search to last a maximum of 120 seconds (2 minutes):

- **Database profile** Type `120` in the Directory Search Time Limit box on the Directory Services page in the Database Profile Setup dialog box.
- **Application** Type the following in code:

```
SQLCA.DBParm="DS_TimeLimit=120"
```

See also

DS_Alias
DS_Copy
DS_DitBase
DS_Failover
DS_Password
DS_Principal
DS_Provider
Release

Encrypt

Description

Specifies that data should be encrypted before sending it over the network.

When to specify Encrypt
You must specify the Encrypt parameter *before* connecting to the database.

Applies to

SNC SQL Native Client for Microsoft SQL Server

Syntax

Encrypt=*value*

Parameter	Description
value	Specifies whether data should be encrypted. This parameter is used in conjunction with the TrustServerCertificate DBParm. Values are: • **0** (Default) No encryption occurs. • **1** If TrustServerCertificate is not set, encryption occurs only if there is a verifiable server certificate. If TrustServerCertificate is set, encryption always occurs, but may use a self-signed server certificate.

Default value

Encrypt=0

Usage

SQL Server 2005 always encrypts network packets associated with logging in to the server. If no certificate is provided on the server when it starts up, SQL Server generates a self-signed certificate that is used to encrypt login packets.

SQL Server Configuration Manager can be used to configure the SQL Native Client to request an encrypted connection using the Secure Sockets Layer (SSL), and to accept a self-signed certificate without validation. You can also request encryption by setting the Encrypt DBParm to 1, which sets the SQL Native Client connection string keyword Encrypt. To enable encryption to be used when a certificate has not been provided on the server, set both Encrypt and TrustServerCertificate. The value of TrustServerCertificate is ignored if Encrypt is not set.

Examples

To specify that PowerBuilder should encrypt data:

- **Database profile** Select the Encrypt Data check box on the System page in the Database Profile Setup dialog box.
- **Application** Type the following in code:

```
SQLCA.DBParm="Encrypt=1"
```

See also

TrustServerCertificate

EncryptionPass

Description

Specifies a password for encrypting and decrypting data.

When to specify EncryptionPass
You must specify the EncryptionPass parameter *before* connecting to the database.

Applies to

I10 Informix

Syntax

EncryptionPass='*value*'

Parameter	Description
value	A string that will be used as the password for encrypting and decrypting data.

Default value

None.

Usage

In IDS 10.0 and later, the SQL statement SET ENCRYPTION PASSWORD can improve the confidentiality of data and support data integrity by defining or resetting a password for encryption and decryption of data at the column level.

You can set the EncryptionPass and Hint static DBParms on the System tab page in the Database Profile Setup dialog box for I10 connections to specify a password and an optional hint to help you remember the password. The password does not display in the database Profile Setup dialog box and is encrypted in the database profile in the registry. The application uses built-in Informix functions to encrypt and decrypt character data.

Examples

To specify Archimedes as the password for data encryption and Eureka as the hint to help you remember the password:

- **Database profile** Enter Archimedes in the Encryption Password box and Eureka in the Hint box on the System page in the Database Profile Setup dialog box.
- **Application** Type the following in code:

```
SQLCA.DBParm="EncryptionPass='Archimedes',
Hint='Eureka'"
```

See also

Hint

EncryptPassword

Description

Specifies whether you want PowerBuilder to encrypt your password automatically when connecting to an OLE DB data provider.

When to specify EncryptPassword
You must specify the EncryptPassword parameter *before* connecting to the database.

Applies to

OLE DB

Syntax

EncryptPassword='*value*'

Parameter	Description
value	Specifies whether you want PowerBuilder to encrypt your password. Values are: • **True** Tells PowerBuilder to encrypt the password • **False** (Default) Tells PowerBuilder not to encrypt the password

Default value

EncryptPassword='False'

Examples

To tell PowerBuilder to encrypt your password when connecting to Microsoft SQL Server or an OLE DB data provider:

- **Database profile** Select the Encrypt Password check box on the Security page in the Database Profile Setup dialog box.
- **Application** Type the following in code:

```
SQLCA.DBParm="EncryptPassword='True'"
```

See also

DataLink
IntegratedSecurity
MaskPassword
PersistEncrypted

FailoverPartner

Description

Specifies the name of a mirror server, allowing you to maintain database availability if a failover event occurs.

Applies to

SNC SQL Native Client for Microsoft SQL Server

Syntax

FailoverPartner =*value*

Parameter	Description
value	Specifies the name of a mirror server.

Default value

None

Usage

When failover occurs, the existing PowerBuilder connection to SQL Server is lost. The SNC driver releases the existing connection and tries to reopen it. If reconnection succeeds, PowerBuilder triggers the DBNotification event.

The following conditions must be satisfied for PowerBuilder to trigger the failover event:

- The FailoverPartner DBParm is supplied at connect time

- The SQL Server database is configured for mirroring
- PowerBuilder is able to reconnect successfully when the existing connection is lost

When failover occurs:

- PowerBuilder returns an error code (998) and triggers the DBNotification event with notification type DBFailover!
- Existing cursors cannot be used and should be closed
- Any failed database operation can be tried again
- Any uncommitted transaction is lost. New transactions must be started

Examples

The following example sets the name of a mirror server for failover events:

- **Database profile** Type the mirror server name in the Failover Partner text box on the System page of the Database Profile Setup dialog box.
- **Application** Type the following in code:

```
my_trans.dbparm="FailoverPartner='myMirrorServer'"
```

FoDelay

Description

Specifies the amount of time (in milliseconds) you want PowerBuilder to wait between attempts to fail over to another database server if the current database server goes down.

When to specify FoDelay
You must specify the FoDelay parameter *before* connecting to the database.

Applies to

O90 Oracle9*i*
O10 Oracle 10*g*
ORA Oracle 11*g*

Syntax

FoDelay='*value*'

Parameter	Description
value	Specifies the amount of time in milliseconds you want PowerBuilder to wait between attempts to fail over to an another database server.

Default value

FoDelay='10'

Usage

You can enter a failover delay value only if you have enabled failover.

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

Examples

To tell PowerBuilder to wait 20 milliseconds between attempts to fail over:

- **Database profile** Type `20` in the Delay box on the Network page in the Database Profile Setup dialog box.
- **Application** Type the following in code:

```
SQLCA.DBParm="FoDelay='20'"
```

See also

SvrFailover

FoDialog

Description

Specifies whether PowerBuilder displays a runtime dialog box indicating when a failover occurs.

When to specify FoDialog
You must specify the FoDialog parameter *before* connecting to the database.

Applies to

ASE, SYC Sybase Adaptive Server Enterprise
O90 Oracle9*i*
O10 Oracle 10*g*
ORA Oracle 11*g*

Syntax

FoDialog='*value*'

Parameter	Description
value	Specifies whether you want PowerBuilder to display a runtime dialog box indicating when a failover occurs. Values are: • **No** (Default) PowerBuilder should not display a dialog box. • **Yes** PowerBuilder should display a dialog box.

Default value

FoDialog='No'

Usage

You can display a runtime dialog box only if you have enabled failover. The dialog box does not display in EAServer or COM+.

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

Examples

To tell PowerBuilder to display a runtime dialog box when a failover occurs:

- **Database profile** Select the Display Runtime Dialog When Failing Over check box on the Network page in the Database Profile Setup dialog box.

- **Application** Type the following in code:

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

See also

SvrFailover

FoRetryCount

Description

Specifies the number of times you want PowerBuilder to try to fail over to an another database server if the current database server goes down.

When to specify FoRetryCount
You must specify the FoRetryCount parameter *before* connecting to the database.

Applies to

O90 Oracle9*i*
O10 Oracle 10*g*
ORA Oracle 11*g*

Syntax

FoRetryCount='*value*'

Parameter	Description
value	Specifies the number of times you want PowerBuilder to try to fail over.

Default value

FoRetryCount='10'

Usage

You can enter a failover retry value only if you have enabled failover.

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

Examples

To tell PowerBuilder to try 20 times to fail over:

- **Database profile** Enter the value 20 in the Retry Count box on the Network page in the Database Profile Setup dialog box.
- **Application** Type the following in code:

  ```
  SQLCA.DBParm="FoRetryCount='20'"
  ```

See also

SvrFailover

FormatArgsAsExp

Description

Controls whether PowerBuilder converts a DataWindow object 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 (the default), PowerBuilder performs this conversion.

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

Applies to

ASE, SYC, and SYJ Sybase Adaptive Server Enterprise
DIR Sybase DirectConnect
JDB JDBC
ODBC interface
O90 Oracle9*i*
O10 Oracle 10*g*
ORA Oracle 11*g*

Syntax

FormatArgsAsExp= '*value*'

Parameter	Description
value	Specifies whether you want PowerBuilder to convert a DataWindow or report retrieval argument of decimal datatype to scientific (exponential) notation if the argument exceeds 12 digits but has fewer than 16 digits. Values are: • **Yes** PowerBuilder converts a retrieval argument of decimal datatype to scientific notation if it exceeds 12 digits but has fewer than 16 digits. • **No** (Default) PowerBuilder 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, PowerBuilder converts retrieval arguments of type decimal to scientific notation if the argument exceeds 12 digits but has fewer than 16 digits. 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 tells PowerBuilder to leave the retrieval argument as a decimal and not convert it to scientific notation. This speeds data retrieval for large databases.

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

Examples

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

- **Database profile** Check the Format Arguments in Scientific Notation check box on the Syntax page (or Transaction page in the case of the DIR interface) in the Database Profile Setup dialog box.
- **Application** Type the following in code:

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

GenerateEqualsNull

Description

Specifies how DataWindows generates =null and <> null expressions in retrieval arguments.

Applies to

All database interfaces

Syntax

GenerateEqualsNull= '*value*'

Parameter	Description
value	• **Yes** Does not change =null or <> null expressions. • **No** Converts =null expression to IS NULL, <> null to IS NOT NULL.

Default value

GenerateEqualsNull='No'

Usage

When to use When a DataWindow retrieves data from tables that contain null columns, most DBMS interfaces expect expressions of the form, IS NULL and IS NOT NULL. Expressions that reference null data as values, such as `Where column = NULL`, can cause the DBMS to reject the retrieval. For this reason, such expressions in DataWindows are normally converted to the standard ANSI syntax during retrieval.

If your DataWindow retrieves null data from a DBMS that supports expressions of the =null or <> null form, and you want to suppress the conversion of those expressions to standard syntax, you can set the GenerateEqualsNull DBParm to true.

Examples

Consider these two statements:

```
SELECT "a1"."c1"  FROM "a1"  WHERE "a1"."c2" = :p1
SELECT "a1"."c1"  FROM "a1"  WHERE "a1"."c2" <> :p1
```

If GenerateEqualsNull is set to false, the statements are generated as:

```
SELECT "a1"."c1"  FROM "a1"  WHERE "a1"."c2" is null
SELECT "a1"."c1"  FROM "a1"  WHERE "a1"."c2" is not null
```

If GenerateEqualsNull is set to true, the statements are generated as:

```
SELECT "a1"."c1"  FROM "a1"  WHERE "a1"."c2" = null
SELECT "a1"."c1"  FROM "a1"  WHERE "a1"."c2" <> null
```

GetConnectionOption

Description

Specifies how EAServer should behave if all connections in a cache are being used. This parameter applies *only* when a PowerBuilder custom class user object is deployed as an EAServer component.

Applies to

JDB JDBC
ODBC
O90 Oracle9*i*
O10 Oracle 10*g*
ORA Oracle 11*g*
SYJ Sybase Adaptive Server Enterprise

Syntax

GetConnectionOption='*value*'

Parameter	Description
value	Specifies how EAServer should behave if all connections in a cache are being used. Values are: • **JAG_CM_FORCE** (Default) Allocates and opens a new connection. The new connection is not cached and is deallocated when the connection is explicitly or implicitly closed by the component. • **JAG_CM_NOWAIT** Fails with an error if no connection can be made. • **JAG_CM_WAIT** Does not return until a connection is available.

Default value

GetConnectionOption='JAG_CM_FORCE'

Usage

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

Using the SYJ interface

Sybase EAServer uses a slightly different version of the CT-Lib software. Therefore, *at runtime*, you need to use the SYJ database interface rather than ASE or SYC to connect to an Adaptive Server Enterprise database. The SYJ Database Profile Setup dialog box provides a convenient way to set the appropriate connection parameters and then copy the syntax from the Preview page into the script for your Transaction object.

You cannot use the SYJ interface, however, to connect to the database in the PowerBuilder development environment. Therefore, *during the development phase* (before the component has been deployed to EAServer), you must use ASE or SYC to connect to the database.

For information on how to use PowerBuilder to build EAServer components, see *Application Techniques*.

Examples

On the EAServer page in the Database Profile Setup dialog box, select JAG_CM_NOWAIT from the Get Connection Option drop-down list. The PowerScript syntax for the GetConnectionOption parameter displays on the Preview page:

```
SQLCA.DBParm="GetConnectionOption='JAG_CM_NOWAIT'"
```

Copy the syntax from the Preview page into your script.

See also

CacheName
ReleaseConnectionOption
UseContextObject

HANotification

Description

Specifies whether a High Availability (HA) client connected to an RAC database will be notified if the database server shuts down.

Applies to

O10 Oracle 10*g*

Syntax

HANotification=*value*

Parameter	Description
value	A value specifying whether the client will be notified if the database server shuts down. Values are: • **No** (Default) Do not notify the client if the server shuts down. • **Yes** Notify the client if the server shuts down.

Default value

HANotification="No"

Usage

Oracle Real Application Clusters (RAC) is a cluster database that uses a shared cache architecture. In Oracle 10*g* Release 2, a High Availability (HA) client connected to an RAC database can register a callback to indicate that it wants the server to notify it in case of a database failure event that affects a connection made by the client. The DBNotification event on the Transaction object is triggered when the client is notified that such an event has occurred.

The HANotification database parameter must be set to "Yes" for the DBNotification event to be triggered when the database server shuts down. If HANotification is set to "Yes", the Oracle client and server must both be Oracle 10*g* version 10.2.0.1.0 or higher or the connection will fail.

This database parameter is not supported by EAServer and MTS/COM+ and will be ignored in PowerBuilder component connections on EAServer and MTS/COM+.

Examples

To specify that the HA client should be notified when the database server shuts down:

- **Database profile** Select the HANotification check box on the System page in the Database Profile Setup dialog box.
- **Application** Type the following in code:

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

HighSeverityError

Description

Determines whether the first error or the highest severity error is reported when more than one error is raised during the connection or attempted connection to the database

Applies to

SNC SQL Native Client for Microsoft SQL Server

Syntax

HigSeverityError=*value*

Parameter	Description
value	Specifies the error to be reported when multiple errors are raised. Values are: • **0** (Default) The first error raised is the reported error. • **1** The error with the highest severity code is the reported error.

Default value

HighSeverityError=0

Usage

When you set this DBParm to 1, PowerBuilder chooses the error with the highest severity code to pass back to the caller.

Examples

To have PowerBuilder report the error with the highest severity code, use the following DBParm setting:

```
SQLCA.DBParm="HighServerityError=1"
```

Hint

Description

Specifies a hint to assist in remembering the password specified for encrypting and decrypting data.

When to specify Hint
You must specify the Hint parameter *before* connecting to the database.

Applies to

I10 Informix

Syntax

Hint='*value*'

Parameter	Description
value	A string that will be used as a hint to help you remember the password used for encrypting and decrypting data.

Default value

None.

Usage

In IDS 10.0 and later, the SQL statement SET ENCRYPTION PASSWORD can improve the confidentiality of data and support data integrity by defining or resetting a password for encryption and decryption of data at the column level.

You can set the EncryptionPass and Hint static DBParms on the System tab page in the Database Profile Setup dialog box for I10 connections to specify a password and an optional hint to help you remember the password. The password does not display in the database Profile Setup dialog box and is encrypted in the database profile in the registry. The application uses built-in Informix functions to encrypt and decrypt character data.

Examples

To specify Archimedes as the password for data encryption and Eureka as the hint to help you remember the password:

- **Database profile** Enter Archimedes in the Encryption Password box and Eureka in the Hint box on the System page in the Database Profile Setup dialog box.
- **Application** Type the following in code:

```
SQLCA.DBParm="EncryptionPass='Archimedes',
Hint='Eureka'"
```

See also

EncryptionPass

Host

Description

If your DBMS supports it, specifies the workstation name when connecting to the database in PowerBuilder. The Host parameter lets you assign any 10-character label to identify the process you are about to create when you connect to the database. This label helps you distinguish your process from others running on the database server.

When to specify Host
You must specify the Host parameter *before* connecting to the database in PowerBuilder.

Applies to

ASE, SYC Sybase Adaptive Server Enterprise
SNC SQL Native Client for Microsoft SQL Server

Syntax

Host='*workstation_name*'

Default value

None

Usage

For Adaptive Server, when you specify a value for Host, PowerBuilder sets the CS_HOSTNAME connection property to the workstation name you specify.

The value you specify for the Host parameter displays in the hostname column of the MASTER.DBO.SYSPROCESSES table in a SQL Server database. How you use the Host parameter depends on the design of your PowerBuilder application.

For example, many sites want to secure their production tables so that updates are possible only through a specific application. To do this, you can grant explicit authority to the PowerBuilder application but *not* to users. The application prompts the user for an authorization ID and password, verifies it, and then connects to the database through a single application login ID. Only this application login ID has authorization to update production tables.

In this scenario, you can use the Host parameter to store the name of the user running the application.

Examples

Example 1 To set the host name to Alan:

- **Database profile** Type the following in the Workstation Name box on the Network page in the Database Profile Setup dialog box:

  ```
  Alan
  ```

- **Application** Type the following in code:

  ```
  SQLCA.DBParm="Host='Alan'"
  ```

Example 2 You can use the Host and AppName parameters together to specify both the host name and the application name. To set the host name to Jane and the application name to Sales:

- **Database profile** Type Jane in the Workstation Name box and Sales in the Application Name box on the Network page in the Database Profile Setup dialog box.
- **Application** Type the following in code:

  ```
  SQLCA.DBParm="Host='Jane',AppName='Sales'"
  ```

Example 3 The Host name in the preceding examples is hard coded. You can get the name dynamically using the Windows GetComputerNameW function. There is no PowerScript equivalent for this function. Here is the external function declaration:

```
FUNCTION boolean GetComputerNameW(ref string cname,ref
long nbuf) LIBRARY "Kernel32.dll"
```

The following code in the Open event of the application uses an external function call to get the host name and set its value in the Host parameter. You must allocate sufficient space for the returned host name:

```
string ls_compname
```

```
long ll_buf
ll_buf=25

ls_compname=space(ll_buf)
GetComputerNameA(ls_compname, ll_buf)

// Profile mysyb
SQLCA.DBMS="SYC Adaptive Server Enterprise"
SQLCA.Database="mydata"
SQLCA.LogPass="mylogpass"
SQLCA.ServerName="mysybsvr"
SQLCA.LogId="mylogid"
SQLCA.AutoCommit=False
SQLCA.DBParm="Host='" + ls_compname + "'"

Connect using SQLCA;
```

See also AppName

HostReqOwner

Description Specifies the name of the host request library defined in a DB2/MVS database.

When to specify HostReqOwner
You must specify the HostReqOwner parameter *before* connecting to the database.

Applies to DIR Sybase DirectConnect (applies only to Access Service for DB2/MVS and Open ServerConnect™)

Syntax HostReqOwner='*owner_id*'

Default value HostReqOwner='Sybase'

Usage The host request library is a special DB2 table that stores host-resident requests. A host-resident request is a SQL statement that a client application can execute as a procedure. If you do not use Sybase as the owner name for this host request library, you should set the HostReqOwner parameter to an appropriate name for your site.

TRS Support
The HostReqOwner parameter is not applicable to DirectConnect TRS connections.

Examples

To set the name of your host request library to Stratus:

- **Database profile** Type `Stratus` in the Host Request Lib Owner box on the System page in the Database Profile Setup dialog box.
- **Application** Type the following in code:

```
SQLCA.DBParm="HostReqOwner='Stratus'"
```

See also

TRS
UseProcSyntax

IdentifierQuoteChar

Description

Specifies the single quote character you want PowerBuilder to use to delimit the names of identifiers (tables, columns, indexes, and constraints) when it generates SQL statements. PowerBuilder 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
In order for IdentifierQuoteChar to take effect, the DelimitIdentifier parameter must be set to Yes. Otherwise, PowerBuilder's default behavior is *not* to delimit identifiers in SQL statements and to ignore any value specified for IdentifierQuoteChar.

Applies to

ADO.NET
JDB JDBC
ODBC (if driver and back-end DBMS support this feature)
OLE DB

Syntax

IdentifierQuoteChar = '*quote_character*'

Parameter	Description
quote_character	The single character you want PowerBuilder 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

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

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

2 The section for your ODBC driver in the PBODB125 initialization file or the section for your JDBC driver in the registry

If PowerBuilder 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.

When using the OLE DB interface
If no value is specified for the IdentifierQuoteChar parameter, PowerBuilder does not use a quote character.

Usage

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

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 PowerBuilder to use to delimit identifiers in SQL statements:

- **Database profile** Type `c` in the Identifier Quote Character box on the Syntax page in the Database Profile Setup dialog box.
- **Application** Type the following in code:

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

See also

DelimitIdentifier

Identity

Description

Specifies the syntax the SNC database interface uses to obtain the identity value of a new row in a database table.

When a DataWindow update request inserts a new row into a Microsoft SQL Server table that contains an IDENTITY column, the DataWindow engine calls the SNC interface to obtain the identity value of the newly inserted row. The Identity database parameter allows you to define how this request is implemented.

Applies to

SNC SQL Native Client for Microsoft SQL Server

Syntax

Identity=*value*

Parameter	Description
value	A value specifying the syntax for obtaining the identity value of a newly inserted row. Values are: • **@@IDENTITY** (Default) Use the syntax `SELECT @@identity`. • **IDENT_CURRENT()** Use the syntax `SELECT IDENT_CURRENT('tablename').` • **SCOPE_IDENTITY()** Use the syntax `SELECT scope_identity().`

Default value

Identity="@@IDENTITY"

Usage

By default, the SNC interface issues `SELECT @@identity` to obtain the IDENTITY column value of the newly inserted row. It returns the last IDENTITY value produced on a connection, regardless of the table that produced the value, and regardless of the scope of the statement that produced the value.

`SELECT IDENT_CURRENT('tablename')` returns the last IDENTITY value produced in a table, regardless of the connection that created the value, and regardless of the scope of the statement that produced the value.

`SELECT SCOPE_IDENTITY()` returns the last IDENTITY value produced on a table and by a statement in the same scope, regardless of the table that produced the value.

Because Identity is a dynamic database parameter, it can be set and reset at any time during an application.

Examples

To specify the syntax for obtaining the identity value of a newly inserted row:

- **Database profile** Select SCOPE_IDENTITY() from the DataWindow Identity Value drop-down list on the Syntax page in the Database Profile Setup dialog box.
- **Application** Type the following in code:

```
SQLCA.DBParm="Identity='SCOPE_IDENTITY()'"
```

ImpersonationLevel

Description

Specifies the level of impersonation that the data server is allowed to use when impersonating its OLE DB data provider and PowerBuilder. This parameter applies only to network connections other than Remote Procedure Call (RPC) connections.

When to specify ImpersonationLevel
You must specify the ImpersonationLevel parameter *before* connecting to the database.

Applies to

OLE DB

Syntax

ImpersonationLevel='*value*'

Parameter	Description
value	Specifies the level of impersonation. Values are: • **Not set** No level of impersonation is selected. • **Anonymous** The client is anonymous to the server and the server process cannot obtain identification information about the client and cannot impersonate the client. • **Delegate** The process can impersonate the client's security context while acting on behalf of the client. The server process can also make outgoing calls to other servers while acting on behalf of the client. • **Identify** The server can obtain the client's identity. The server can impersonate the client for ACL checking but cannot access system objects as the client. • **Impersonate** The server process can impersonate the client's security context while acting on behalf of the client. This information is obtained when the connection is established, not on every call.

Default value

ImpersonationLevel='Not set'

Examples

To set a level of impersonation to anonymous:

- **Database profile** On the Security page in the Database Profile Setup dialog box, select Anonymous from the Impersonation Level drop-down list.
- **Application** Type the following in code:

```
SQLCA.DBParm="IMPERSONATIONLEVEL='DB_IMP_LEVEL_ANON
YMOUS'"
```

See also

DataLink

INET_DBPATH

Description

Specifies the Informix DBPATH setting. The DBPATH environment variable identifies a list of directories that contain Informix databases. INET_DBPATH typically specifies the location of Informix databases if this is *other* than in a directory on the database server.

Applies to

I10 Informix
IN9 Informix

Syntax

INET_DBPATH='*server_db_path*'

Parameter	**Description**
server_db_path	The name of the directory containing Informix databases

Default value

By default, PowerBuilder uses the value specified for DBPATH in the *HKEY_LOCAL_MACHINE\SOFTWARE\Informix\Environment* registry key.

Examples

Example 1 To specify that the directory */HOME/Informix* contains Informix databases:

- **Database profile** Type the following in the Database Path box on the Network page in the Database Profile Setup dialog box:

```
/home/Informix
```

- **Application** Type the following in code:

```
SQLCA.DBParm="INET_DBPATH='/home/Informix'"
```

Example 2 You can specify values for INET_DBPATH, INET_PROTOCOL, and INET_SERVICE together. To specify that the directory */Informix* contains Informix databases, and that you want to connect using the turbo service and the olsoctcp network protocol:

- **Database profile** Type */Informix* in the Database Path box, turbo in the Service Name box, and olsoctcp in the Protocol Type box on the Network page in the Database Profile Setup dialog box.
- **Application** Type the following on a single line in code:

```
SQLCA.DBParm="INET_DBPATH='/Informix',INET_SERVICE=
'turbo',INET_PROTOCOL='olsoctcp'"
```

See also

INET_PROTOCOL
INET_SERVICE

INET_PROTOCOL

Description

Specifies the network protocol that the Informix client software uses to communicate with a remote Informix database server.

Applies to

I10 Informix
IN9 Informix

Syntax

INET_PROTOCOL='*network_protocol* '

Parameter	Description
network_protocol	A string that specifies the name of the network protocol used by the Informix client software. For information about the correct network protocol for your site, see your Informix system administrator.

Default value

By default, PowerBuilder uses the network protocol specified in the *HKEY_LOCAL_MACHINE\SOFTWARE\Informix\SqlHosts* registry key.

Examples

Example 1 To specify that Informix client software uses the Novell IPX/SPX network protocol:

- **Database profile** Type `ipx` in the Protocol Type box on the Network page in the Database Profile Setup dialog box.
- **Application** Type the following in code:

```
SQLCA.DBParm="INET_PROTOCOL='ipx'"
```

Example 2 You can specify values for INET_DBPATH, INET_PROTOCOL, and INET_SERVICE together. To specify that the directory */Informix* contains Informix databases, and that you want to connect using the turbo service and the olsoctcp network protocol:

- **Database profile** Type */Informix* in the Database Path box, turbo in the Service Name box, and olsoctcp in the Protocol Type box on the Network page in the Database Profile Setup dialog box.
- **Application** Type the following on a single line in code:

```
SQLCA.DBParm="INET_DBPATH='/Informix',INET_SERVICE=
'turbo',INET_PROTOCOL='olsoctcp'"
```

See also

INET_DBPATH
INET_SERVICE

INET_SERVICE

Description

Specifies the name of the service that a remote Informix database server uses to listen to all incoming requests from client applications.

Applies to

I10 Informix
IN9 Informix

Syntax

INET_SERVICE='*service_name*'

Parameter	Description
service_name	A string that specifies the name of the service that a remote Informix database server uses to listen to incoming requests For information about the correct service name for your site, see your Informix system administrator.

Default value

By default, PowerBuilder uses the service name specified in the *HKEY_LOCAL_MACHINE\SOFTWARE\Informix\SqlHosts* registry key.

Examples

Example 1 To specify that your Informix database server uses the sqlexec service name:

- **Database profile** Type `sqlexec` in the Service Name box on the Network page in the Database Profile Setup dialog box.
- **Application** Type the following in code:

```
SQLCA.DBParm="INET_SERVICE='sqlexec'"
```

Example 2 You can specify values for INET_DBPATH, INET_PROTOCOL, and INET_SERVICE together. To specify that the directory */Informix* contains Informix databases, and that you want to connect using the turbo service and the olsoctcp network protocol:

- **Database profile** Type */Informix* in the Database Path box, turbo in the Service Name box, and olsoctcp in the Protocol Type box on the Network page in the Database Profile Setup dialog box.
- **Application** Type the following on a single line in code:

```
SQLCA.DBParm="INET_DBPATH='/Informix',INET_SERVICE=
'turbo',INET_PROTOCOL='olsoctcp'"
```

See also

INET_DBPATH
INET_PROTOCOL

Init_Prompt

Description

Specifies whether you want to be prompted during initialization.

When to specify Init_Prompt
You must specify the Init_Prompt parameter *before* connecting to the database.

Applies to

OLE DB

Syntax

Prompt='*value*'

Parameter	Description
value	Specifies whether you want to be prompted during initialization. Values are: • **Not set** Do not prompt. • **Always** Always prompt for initialization information. • **If needed** Prompt only if more information is needed. • **If needed (required)** Prompt only if more information is needed. Do not allow the user to enter optional information. • **Never** Do not prompt.

Default value

Init_Prompt='Not set'

Examples

To specify that you want always to be prompted during initialization:

- **Database profile** Select Always from the Prompt drop-down list on the System page in the Database Profile Setup dialog box.
- **Application** Type the following in code:

```
SQLCA.DBParm="INIT_PROMPT='DBPROMPT_PROMPT'"
```

See also

DataLink

InsertBlock

Description

Specifies the number of rows that you want the Data Pipeline in PowerBuilder to insert at one time into a table in the destination database.

For instructions on using the Data Pipeline, see the *Users Guide*.

Applies to

ODBC (only in Data Pipeline if driver and back-end DBMS support this feature)

Syntax

InsertBlock=*insert_blocking_factor*

Parameter	Description
insert_blocking_factor	The number of rows that you want the Data Pipeline to insert at one time into a table in the destination database, up to a maximum of 100 rows (Default=100 rows). To turn off block inserting for an ODBC data source in the Data Pipeline, set InsertBlock to 1 or DisableBind to 1 in the database profile of the destination database.

Default value

InsertBlock=100

Usage

Requirements for using InsertBlock To use the InsertBlock parameter, *all* of the following must be true:

- You are using an ODBC driver to access the destination database in the Data Pipeline.
- The destination database supports the use of bind variables. (For more about bind variables, see DisableBind.)
- The DisableBind parameter is not set to 1 (the default is 0) in the database profile of the destination database. This enables the default binding of input parameters to a compiled SQL statement in PowerBuilder.
- Maximum Errors is set to 1 in the Data Pipeline.

The SQL Anywhere ODBC driver meets the first two requirements.

To determine whether your ODBC driver meets these requirements, see the documentation that comes with your driver.

Determining the InsertBlock value PowerBuilder searches the following in this sequence to determine the value for InsertBlock:

1 The section for your database profile in the PowerBuilder initialization file

2 The section for your ODBC driver in the PBODB125 initialization file

If PowerBuilder does not find an InsertBlock value in these locations, it defaults to an insert blocking factor of 100 rows.

What happens When PowerBuilder finds a value for InsertBlock, the Data Pipeline batches the specified number of rows and inserts them with a single call to the ODBC driver you are using to access the destination database.

If you specify an InsertBlock value or Data Pipeline commit factor of fewer than 100 rows, the Data Pipeline batches and inserts the specified number of rows into the destination database. If you specify more than 100 rows, the Data Pipeline batches and inserts at most only 100 rows at one time.

The insert blocking factor that the Data Pipeline actually uses depends on the size of the data in each column inserted in the destination database. In addition, the Data Pipeline does not exceed 64K of data in the buffer for any one column.

Turning off block inserting To turn off block inserting for an ODBC data source in the Data Pipeline, you can do any of the following in the database profile of the destination database:

- Set the InsertBlock parameter to 1
- Set the DisableBind parameter to 1 (to disable default binding of input parameters to a compiled SQL statement)
- In the Data Pipeline, set Maximum Errors to a value other than 1

Examples

To set the insert blocking factor in the Data Pipeline to 50 rows:

- **Database profile** Type `50` in the Insert Blocking Factor box on the Transaction page in the Database Profile Setup dialog box.
- **Application** Type the following in code:

```
SQLCA.DBParm="InsertBlock=50"
```

To set the insert blocking factor in the Data Pipeline to 50 rows, type `50` in the Insert Blocking Factor box on the Transaction page in the Database Profile Setup dialog box.

See also

DisableBind

IntegratedSecurity

Description

Specifies the name of the authentication service used by the data server to identify the user.

If this parameter is specified, none of the other OLE DB authentication parameters (CacheAuthentication, EncryptPassword, MaskPassword, PersistEncrypted, and PersistSecurityInfo) are needed and are ignored if specified.

When to specify IntegratedSecurity
You must specify the IntegratedSecurity parameter *before* connecting to the database.

Applies to

OLE DB

Syntax

IntegratedSecurity='*value*'

Parameter	Description
value	A string specifying the name of the authentication service. If NULL, the default authentication service is used.

Default value

None

Examples

To use an authentication service such as the Security Support Provider Interface (SSPI) for Windows NT:

- **Database profile** Type the name of the authentication service in the Integrated Security box on the Security page in the Database Profile Setup dialog box.
- **Application** Type the following in code:

```
SQLCA.DBParm="IntegratedSecurity='SSPI'"
```

Isolation

Description

Sets the isolation level to use when connecting to the database.

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, you can set the isolation level when you connect to the database. Isolation levels are used by .NET Framework data providers when performing a transaction.

PowerBuilder uses the Isolation database parameter to allow you to set various database lock options. Each value corresponds to an isolation level defined in the .NET Framework.

When to specify the Isolation value
You must set the Isolation value *before* you connect to the database. The Isolation value takes effect only when the database connection occurs. Changes made to the Isolation value after the connection occurs have no effect on the current connection.

Applies to

ADO.NET

Syntax

Isolation='*value*'

Parameter	Description
value	A string specifying the isolation level. Values correspond to members of the .NET Framework IsolationLevel enumeration: • TC – Chaos • None specified – The default isolation level for the DBMS (Default) • RC – Read Committed • RU – Read Uncommitted • RR – Repeatable Read • TS – Serializable Transactions

Default value

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

Examples

Example 1 To set the Isolation value to RC (Read Committed):

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

```
SQLCA.DBParm="Isolation='RC'"
```

Using the example in code
If you specify Isolation Level in your database profile, the syntax displays on the Preview page in the Database Profile Setup dialog box. You can copy the syntax from the Preview page into your code.

KeepAlive

Description

Determines whether packets are sent to the database to ensure that the connection is still active.

When to specify KeepAlive
You must specify the KeepAlive parameter *before* connecting to the database.

Applies to

ASE, SYC Sybase Adaptive Server Enterprise

Syntax

KeepAlive=*value*

Parameter	Description
value	Specifies whether the Open Client/Server CS_CON_KEEPALIVE property is set for your connection. Values are: • **1** (Default) CS_CON_KEEPALIVE property is set. • **0** CS_CON_KEEPALIVE property is not set.

Default value

KeepAlive=1

Usage

KeepAlive sets the value of the Sybase CT-Lib connection property CS_CON_KEEPALIVE to true or false. The default setting ensures that your connection is alive by sending packets to the database when the connection is idle. Set the value of this property to false for mobile clients that do not maintain constant connections.

Examples

To set the KeepAlive value to 0 when you do not want to maintain a connection:

- **Database profile** Clear the Keep Connection Alive check box on the Network page.
- **Application** Type the following in code:

```
SQLCA.DBParm="KeepAlive=0"
```

Language

Description

For those interfaces that support it, specifies the language you want to use when connecting to your target database.

When to specify Language
You must specify the Language parameter *before* connecting to the database. The Language setting takes effect when you connect to the database.

Applies to

ASE, SYC Sybase Adaptive Server Enterprise
DIR Sybase DirectConnect

Syntax

Language='*language_name*'

Default value

None

Usage

When you specify a value for Language, PowerBuilder:

- Allocates a CS_LOCALE structure for this connection
- Sets the CS_SYB_LANG value to the language you specify

- Sets the SQL Server CS_LOC_PROP connection property with the new locale information

If you have previously set a value for the Locale parameter, which includes settings for the language and character set you want the Open Client software to use, you can override the language value by specifying a new value for the Language parameter and reconnecting to the database.

Unicode data access PowerBuilder can access Unicode data in an Adaptive Server Enterprise (ASE) 12.5 or higher Unicode database or in Unicode columns in ASE 12.5 or higher. PowerBuilder converts between double-byte character set (DBCS) data and Unicode automatically, provided that the Language and CharSet parameters are set with DBCS values (or the Locale parameter is set with DBCS values).

For example:

```
Language='tchinese'
CharSet='big5'
```

Examples

To set the Language parameter to French:

- **Database profile** Type `French` in the Language box on the Connection page or Regional Settings page in the Database Profile Setup dialog box.
- **Application** Type the following in code:

```
SQLCA.DBParm="Language='French'"
```

See also

CharSet
Locale

LCID

Description

Specifies the locale identifier that you want the OLE DB data provider to use.

When to specify LCID
You must specify the LCID parameter *before* connecting to the database.

Applies to

OLE DB

Syntax

LCID='*lcid_name*'

Default value

None

Usage

You specify the locale identifier at initialization. This provides a way for the data server to determine PowerBuilder's preferred locale language and character set. However, setting this parameter does not guarantee that all text returned to PowerBuilder is translated according to the locale ID.

Examples

To set the locale to US English:

- **Database profile** Type `1033` in the LCID box on the System page in the Database Profile Setup dialog box.
- **Application** Type the following in code:

```
SQLCA.DBParm="LCID='1033'"
```

See also

CharSet
Language

Locale

Description

Specifies the locale name that you want the Sybase Open Client software to use when connecting to a Sybase Adaptive Server Enterprise database or a database accessed through DirectConnect in PowerBuilder.

When to specify Locale
You must specify the Locale parameter *before* connecting to the database.

Applies to

ASE, SYC Sybase Adaptive Server Enterprise
DIR Sybase DirectConnect

Syntax

Locale='*locale_name*'

Default value

The default locale defined in your *LOCALES.DAT* file

Usage

Locales Locales are stored as entries in a file named *LOCALES.DAT*. The *LOCALES.DAT* file contains information about the languages and character sets you are using with the Sybase Open Client software. The Sybase Open Client installation places the *LOCALES.DAT* file in the *$SYBASE\LOCALES* directory.

An entry in the *LOCALES.DAT* file has the following format:

locale=*locale_name*, *language_name*, *character_set_name*

For example:

```
locale=default, us_english, cp850
locale=enu, us_english, cp850
locale=fra, french, cp850
```

Why set Locale parameter Setting a value for the Locale parameter lets you use a locale *other than the default locale* when accessing an Adaptive Server Enterprise or DirectConnect database. If you do not set a value for Locale, Sybase Open Client uses the default locale defined in your *LOCALES.DAT* file.

What happens When you specify a value for the Locale parameter, PowerBuilder:

- Allocates a CS_LOCALE structure for this connection
- Sets the CS_LC_ALL value to the locale name you specify
- Sets the SQL Server CS_LOC_PROP connection property with the new locale information

Overriding Locale parameter If you have previously set a value for the Locale parameter that includes settings for the language and character set you want to use, you can override the language or character set values by specifying new values for the Language or CharSet parameter and reconnecting to the database.

Unicode data access PowerBuilder can access Unicode data in an ASE 12.5 or later Unicode database or in Unicode columns in ASE 12.5 or later. PowerBuilder converts between double-byte character set (DBCS) data and Unicode automatically, provided that the Locale parameter is set with DBCS values. For example, the Locale parameter should be set to chs or cht.

Examples

To set the locale to *fra*:

- **Database profile** Type the following in the Locale box on the Regional Settings page in the Database Profile Setup dialog box:

  ```
  fra
  ```

- **Application** Type the following in code:

  ```
  SQLCA.DBParm="Locale='fra'"
  ```

What happens Setting the Locale parameter to *fra* has the same effect as individually setting both the Language and CharSet parameters as follows:

```
Language='French'
CharSet='cp850'
```

See also

CharSet
Language

Location

Description

Specifies the location of the data source to which you want your OLE DB data provider to connect. Typically the location is the database server name.

When to specify Location
You must specify the Location parameter *before* connecting to the database.

Applies to

OLE DB

Syntax

Location='*location_name*'

Default value

None

Usage

Implementation of the Location parameter varies depending on the OLE DB data provider you are using. For specific information, see the data provider documentation provided by the OLE DB vendor.

Log

Description

Specifies whether the database server should log updates of text and image data in the transaction log. By default, the database server logs updates of text and image data in the transaction log.

Applies to

ASE, SYC and SYJ Sybase Adaptive Server Enterprise

Syntax

Log=*value*

Parameter	Description
value	A value that specifies whether the database server should log updates of text and image data in the transaction log. Values are: • **0** Do not log text and image updates in the transaction log. Specify this value only if your database server allows you to disable logging. • **1** (Default) Log text and image updates in the transaction log.

Default value

Log=1

Usage

You should set the Log parameter to 0 only if your database server allows you to disable logging.

Examples

To specify that PowerBuilder should *not* log text and image updates in the transaction log:

- **Database profile** Clear the Log Text and Image Updates check box on the System page or Transaction page in the Database Profile Setup dialog box.
- **Application** Type the following in code:

```
SQLCA.DBParm="Log=0"
```

To specify that PowerBuilder should *not* log text and image updates in the transaction log, clear the Log Text and Image Updates check box on the System page or Transaction page in the Database Profile Setup dialog box.

LoginTimeOut

Description

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

Applies to

JDB JDBC
ODBC (if driver and back-end DBMS support this feature)

Syntax

LoginTimeOut=*value*

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

Default value

ODBC: LoginTimeOut=15; JDBC: LoginTimeOut=0

Usage

If you set LoginTimeOut to 0, PowerBuilder does not call the JDBC or ODBC driver to set the LoginTimeOut value and instead waits the number of seconds specified by the JDBC or ODBC driver's client software. If you set LoginTimeOut to a value greater than 0, PowerBuilder does call the JDBC or 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 page in the Database Profile Setup dialog box.
- **Application** Type the following in code:

```
SQLCA.DBParm="LoginTimeOut=60"
```

LowerCaseIdent

Description

Specifies whether PowerBuilder displays identifier names in lowercase.

Applies to

DIR Sybase DirectConnect (applies only to DB2/MVS)

Syntax

LowerCaseIdent='*value*'

Parameter	Description
value	Specifies whether you want PowerBuilder to display identifier names in lowercase. Values are: • **Yes** Display identifier names in lowercase • **No** (Default) Do not display identifier names in lowercase

Default value

LowerCaseIdent='No'

Usage

PowerBuilder displays identifier names in uppercase (the way they are stored in the database). The LowerCaseIdent parameter can be set only if the DelimitIdentifier parameter is set to No, indicating that PowerBuilder should not enclose table and column names in double quotes. If you try to enclose a table and column names in double quotes with identifier names in lowercase, the LowerCaseIdent parameter value is reset to the default value, and you receive a warning message.

Migrating PBMDI and PBNET applications to PBDIR
If you are migrating an application that previously used the InformationConnect DB2 Gateway or Net-Gateway for DB2 interface to the DirectConnect for DB2/MVS database interface, you should set the LowerCaseIdent parameter value to Yes. This enables you to continue to use the Select painter to edit DataWindows.

Examples

To have PowerBuilder display identifier names in lowercase:

- **Database profile** Select the Display Identifiers In Lower Case check box on the Syntax page in the Database Profile Setup dialog box.
- **Application** Type the following in code:

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

MapDateToDateTime

Description

Maps the Oracle Date datatype to the DateTime datatype to enable the time part of the data to be retrieved.

Applies to

JDB JDBC

Syntax

MapDateToDateTime=*value*

Parameter	Description
value	Specifies whether the Oracle Date datatype is mapped to the DateTime datatype. Values are: • **0** (Default) The Oracle Date datatype is not mapped to the DateTime datatype. The time part of the Date column is not retrieved. • **1** The Oracle Date datatype is mapped to the DateTime datatype. The time part of the Date column is retrieved.

Default value

MapDateToDateTime=0

Usage

The Oracle Date datatype stores both date and time data. When you connect to Oracle using the Oracle JDBC Thin Driver, only the date part of the data is retrieved. To retrieve the time part of the data, set the MapDateToDateTime database parameter to map the datatype of Oracle Date columns to the DateTime datatype.

Examples

To retrieve both date and time data from an Oracle Date column:

- **Database profile** Select the Map Date to DateTime check box on the Transaction page in the Database Profile Setup dialog box.
- **Application** Type the following in code:

```
SQLCA.DBParm="MapDateToDateTime=1"
```

MaskPassword

Description

Specifies whether you want PowerBuilder to mask your password automatically when connecting to an OLE DB data provider.

When to specify MaskPassword
You must specify the MaskPassword parameter *before* connecting to the database.

Applies to

OLE DB

Syntax

MaskPassword='*value*'

Parameter	Description
value	Specifies whether you want PowerBuilder to mask your password. Values are: • **True** Tells PowerBuilder to mask the password • **False** (Default) Tells PowerBuilder not to mask the password

Default value

MaskPassword='False'

Examples

To tell PowerBuilder to mask your password when connecting to an OLE DB data provider:

- **Database profile** Select the Mask Password check box on the Security page in the Database Profile Setup dialog box.
- **Application** Type the following in code:

```
SQLCA.DBParm="MaskPassword='True'"
```

See also

DataLink
EncryptPassword
IntegratedSecurity
PersistEncrypted

MaxConnect

Description

Sets the maximum number of simultaneous connections you want to make when accessing a database.

The default is 25 simultaneous connections. You can override this default by setting MaxConnect up to the maximum number of simultaneous connections configured on the database server.

When to specify MaxConnect
You must specify a value for the MaxConnect parameter *before* connecting to the database.

Applies to

ASE, SYC Sybase Adaptive Server Enterprise
DIR Sybase DirectConnect

Syntax

MaxConnect=*value*

Parameter	Description
value	The maximum number of simultaneous connections you want to make when accessing a database

Default value

MaxConnect=25

Usage

DirectConnect, ASE, and SYC MaxConnect sets the Sybase CT-Lib connection property CS_MAX_CONNECT to the number of simultaneous database connections you specify for a single CT-Lib context.

Examples

To set the MaxConnect value to a maximum of 50 simultaneous database connections:

- **Database profile** Type `50` in the Maximum Client Library Connections box (when using the ASE or SYC interface) or the Maximum Connections For This Context box (when using the DIR interface). This check box is on the Network page.
- **Application** Type the following in code:

```
SQLCA.DBParm="MaxConnect=50"
```

MaxFetchBuffer

Description

Sets the maximum size of the buffer into which theDataWindow object can fetch rows from the database. Using the MaxFetchBuffer parameter with the Block parameter can improve performance when accessing a database in PowerBuilder.

Applies to

O90 Oracle9*i*
O10 Oracle 10*g*
ORA Oracle 11*g*

Syntax

MaxFetchBuffer=*buffersize*

Parameter	Description
buffersize	The number of bytes the fetch buffer can hold.

Default value

5000000 (bytes)

Using the default buffer size
You should not have to set a non-default value for MaxFetchBuffer. In most cases, the default buffer size should meet your needs.

Usage

You can use the MaxFetchBuffer database parameter in conjunction with the Block database parameter to improve performance when the size of a row is very large.

The size of the actual fetch buffer is the product of the value of the blocking factor and the size of the row. If the fetch buffer required by the blocking factor and the row size is greater than the value of MaxFetchBuffer, the value of the blocking factor is adjusted so that the buffer is not exceeded.

For example, if block=500 and the row size is 10KB, the fetch buffer is 5000KB, which equals the default maximum buffer size.

You can set Block and MaxFetchBuffer dynamically in code after connecting to the database. MaxFetchBuffer cannot be set in the Database Profile Setup dialog box.

Examples

The following example sets the maximum fetch buffer size to 6000KB and the blocking factor to 500:

```
SQLCA.DBParm="MaxFetchBuffer=6000000,block=500"
```

See also

Block (ODBC, OLE DB, Oracle, and SNC)

MixedCase

Description

Specifies whether you want connections to an Oracle database to be case sensitive or case insensitive.

By default, MixedCase is set to 0. This setting specifies a case-insensitive connection and assumes that all identifiers are uppercase. To make the Oracle connection case sensitive, set the MixedCase parameter to 1.

Applies to

O90 Oracle9*i*
O10 Oracle 10*g*
ORA Oracle 11*g*

Syntax

MixedCase=*value*

Parameter	Description
value	Specifies whether an Oracle database connection is case sensitive or case insensitive. Values are: • **0** (Default) The Oracle database connection is case insensitive. It assumes that all identifiers are uppercase. • **1** The Oracle database connection is case sensitive. It supports mixed case, uppercase, and lowercase identifiers.

Default value

MixedCase=0

Usage

When you set the MixedCase parameter to 1 and define a primary key for a table in an Oracle database, all of the following must contain only uppercase letters:

- The name of the primary key
- The name of the table containing the primary key
- The names of any foreign keys that reference the primary key

Examples

To make an Oracle database connection case sensitive:

- **Database profile** Select the Case Sensitive check box on the Connection page in the Database Profile Setup dialog box.
- **Application** Type the following in code:

```
SQLCA.DBParm="MixedCase=1"
```

Mode

Description

Specifies access permission to the OLE DB data provider.

When to specify Mode
You must specify the Mode parameter *before* connecting to the database.

Applies to

OLE DB

Syntax

Mode='*value*'

Parameter	Description
value	Specifies access permission to the OLE DB data provider. Values are: • **Deny read share** Prevents other users from opening in read mode. • **Deny write share** Prevents other users from opening in write mode. • **Exclusive share** Prevents other users from opening in read/write mode. • **No share deny** Neither read nor write access can be denied to other users. • **Read/Write** Allows read/write access. • **Read-only** Allows read access. • **Write-only** Allows write access.

Default value

None

Examples

To allow other users read/write access to the OLE DB data provider:

- **Database profile** On the Transaction page in the Database Profile Setup dialog box, select Read/Write from the Mode list box.
- **Application** Type the following in code:

```
SQLCA.DBParm="MODE='DB_MODE_READWRITE'"
```

See also

DataLink

MsgTerse

Description

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

By default, PowerBuilder displays JDBC and ODBC error messages with the SQLSTATE prefix. To display error messages without the SQLSTATE prefix, set MsgTerse to `'Yes'`.

Applies to

JDB JDBC
ODBC

Syntax

MsgTerse='*value*'

Parameter	**Description**
value	Specifies whether PowerBuilder 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 parameter to 'Yes' to display shorter JDBC or ODBC error messages in PowerBuilder. This might be useful if space on your screen is limited.

For example, suppose you are using the Data Pipeline in PowerBuilder to pipe data to a SQL Anywhere ODBC database, and errors occur while you are executing the pipeline. If MsgTerse is set to 'No' (the default value), pipeline errors display in an Error dialog box *with* the SQLSTATE prefix (for example, SQLSTATE=23000).

If you specify MsgTerse='Yes' in the database profile of the SQL Anywhere destination database, the Data Pipeline displays terse ODBC error messages *without* the SQLSTATE prefix.

For instructions on using the Data Pipeline, see the PowerBuilder *Users Guide*.

Examples

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

- **Database profile** Select the Display Terse Error Messages check box on the System page in the Database Profile Setup dialog box.
- **Application** Type the following in code:

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

Namespace

Description

Specifies the .NET Framework data provider to be used to access data.

When to specify Namespace
You must specify the Namespace parameter *before* connecting to the database.

Applies to

ADO.NET

Syntax

Namespace='*value*'

Parameter	Description
value	Specifies a namespace to be used as the data provider for an ADO.NET connection. Supported namespaces can be selected from the drop-down list.

Default value

None

Usage

The .NET Framework data provider for a given DBMS describes a collection of classes used to access a data source in that format in the managed space.

Examples

To specify that PowerBuilder should use the System.Data.OleDb namespace to connect to a database:

- **Database profile** Select System.Data.OleDb from the Namespace drop-down list on the General page in the Database Profile Setup dialog box.
- **Application** Type the following in code:

```
SQLCA.DBParm="Namespace='System.Data.OleDb'"
```

NCharBind

Description

Specifies whether PowerBuilder binds input string parameters to the Char or NChar datatype.

Applies to

ADO.NET (Oracle.DataAccess.Client only)
O90 Oracle9*i*
O10 Oracle 10*g*
ORA Oracle 11*g*
SNC SQL Native Client for Microsoft SQL Server

Syntax

NCharBind=*value*

Parameter	Description
value	Specifies whether PowerBuilder binds string input parameters to the Char or NChar datatype. Values are: • **0** Binds string input parameters to the Char datatype. • **1** Binds string input parameters to the NChar datatype.

Default value

NCharBind=1 for SNC, NCharBind=0 for other interfaces

Usage

For Oracle, the default NcharBind=0 setting is recommended for binding Char, Varchar, Long and Clob data. NcharBind=1 is recommended for binding Nchar, Nvarchar2 and Nclob data.

For SNC, the default NcharBind=1 is recommended for binding Nchar, Nvarchar, Ntext and Nvarchar(max) data. It encodes the string data as Unicode. NcharBind=0 is recommended for binding Char, Varchar, Text and Varchar(max) data. This setting converts the data to the ANSI string determined by the current operating system code page.

DisableBind must be set to 0
For NcharBind to take effect, the DisableBind parameter must be set to 0. DisableBind=1 overrides the NcharBind setting.

Examples

To specify that string arguments should be bound as the NChar datatype:

- **Database profile** Select the NCharBind box on the Transaction page in the Database Profile Setup dialog box.
- **Application** Type the following in code:

```
SQLCA.DBParm="NCharBind=1"
```

See also

BindSPInput
DisableBind
RPCRebind
UnicharBind

NCharLiteral

Description

Specifies whether the NChar literal replacement feature in the Oracle client is enabled. This feature replaces string literals on the client that are prefaced with the letter N with an internal format. The internal format is decoded to Unicode by the database server when the statement is executed.

When to specify NCharLiteral
You must specify the NCharLiteral parameter *before* connecting to the database.

Applies to

O10 Oracle 10*g*
ORA Oracle 11*g*

Syntax

NCharLiteral=*value*

Parameter	Description
value	Specifies whether PowerBuilder replaces string literals prefaced with the letter N in SQL command text. Values are: • **‘No’** (Default) The text of any literal is encoded in the same character set as the rest of the statement on the client. • **‘Yes’** The text of any literal prefixed by N is replaced with an internal format.

Default value

NCharLiteral='No'

Usage

The NCharLiteral database parameter requires Oracle 10.2 or higher on both the client and the database server.

By default, in a SQL statement, the text of any literal is encoded in the same character set as the rest of the statement. The character set on the client is determined by the client character set defined in NLS_LANG. When the statement is executed, the character set on the client is converted to the character set on the database server.

Data in string literals is lost in the conversion if the character set on the database server does not contain the characters used on the client. NChar string literals are most affected by this issue because they are designed to be independent of the character set on the database server.

To avoid this data loss, add the letter N before string literals that should be replaced with an internal format and set the NCharLiteral database parameter to ‘Yes’. This setting causes the Oracle client to encode all literals prefixed with N in statements on the client with an internal format. The database server decodes the literals to Unicode when the statement is executed.

For example, when NCharLiteral is set to “Yes”, the string “some unicode data” in the following SQL statement is transferred from the client to the server with no data loss:

```
insert into table1 (id, ncharcol) values(1, N'some
unicode data')
```

Setting NCharLiteral to ‘Yes’ has no effect on DataWindow functions.

Examples

To specify that string literals prefixed by the letter N should be replaced with an internal format on the client:

- **Database profile** Select the NChar Literal box on the Connection page in the Database Profile Setup dialog box.
- **Application** Type the following in code:

```
SQLCA.DBParm="NCharLiteral=1"
```

NLS_Charset

Description

Specifies the client-side character set for the current environment handle.

Applies to

O90 Oracle9*i*
O10 Oracle 10*g*
ORA Oracle 11*g*

Syntax

NLS_Charset='*value*'

Parameter	Description
value	Specifies the client-side character set for the current environment handle. Values are: • **Unicode** (Default) The OCI client uses the UTF-16 character set. When connecting to EAServer with a connection cache, the cache’s database driver type must be OCI_9U. • **Local** The OCI client uses the current character set defined by the NLS_LANG parameter on the local computer. When connecting to EAServer with a connection cache, the cache’s database driver type must be OCI_9. • ***ValidCharsetName*** Any valid character set name except AL16UTF16. To specify a value in the Database Profile Setup dialog box, type the name of the character set into the NLS Charset drop-down list box on the System page.

Default value

NLS_Charset='Unicode'

Usage

When you specify a value for NLS_Charset, PowerBuilder sets the OCI NLS_CHARACTERSET property in the current connection. NLS_CHARACTERSET is a database parameter that specifies the character set encoding used to store CHAR, VARCHAR2, LONG, and CLOB datatypes. The value you set corresponds to the character set defined by the NLS_LANG client-side parameter. NLS_LANG defines the character set encoding of text that OCI gets from or sends to applications in bind or define variables or as SQL statements to be executed by the server.

For the national character set NLS_NCHAR_CHARACTERSET, the Oracle database interfaces always use the UTF-16 character set. If you are using pooling with the ORA driver, and two Oracle connections are connected to the same Oracle server but use different character sets, the connections must reside in different connection or session pools. All pooling-related DBParm parameters must be set before the initial database connection.

For EAServer connections, you can use only Unicode or Local. For COM+ connections, you can use only Local.

Examples

To specify that the OCI client should use the current character set defined by the NLS_LANG parameter on the local computer to store string datatypes:

- **Database profile** Select Local from the NLS Charset drop-down list on the System page in the Database Profile Setup dialog box.
- **Application** Type the following in code:

```
SQLCA.DBParm="NLS_Charset='Local'"
```

See also

StrByCharset

NumbersInternal

Description

Specifies that numbers should be retrieved from the database using Oracle’s internal 21-byte binary NUMBER datatype format instead of using OCI strings. The NumbersInternal parameter is relevant *only* when you are accessing an Oracle database configured with an EBCDIC character set or other non-ASCII character set.

When to specify NumbersInternal
You must specify the NumbersInternal parameter *before* connecting to the database.

Applies to

O90 Oracle9*i*
O10 Oracle 10*g*

ORA Oracle 11*g*

Syntax

NumbersInternal=*value*

Parameter	Description
value	Specifies that numbers should be retrieved using Oracle's internal 21-byte binary NUMBER datatype format. Values are: • **0** (Default) Do not retrieve numbers in internal format. • **1** Retrieve numbers in internal format.

Default value

NumbersInternal=0

Usage

In addition to specifying that numbers be retrieved from the database using Oracle's internal 21-byte binary NUMBER datatype format, the NumbersInternal parameter also provides an internal algorithm for deciphering the result.

Examples

To specify that you want numbers to be retrieved using Oracle's internal format:

- **Database profile** Select the Retrieve Numbers in Internal Format check box on the Syntax page in the Database Profile Setup dialog box.
- **Application** Type the following in code:

```
SQLCA.DBParm="NumbersInternal=1"
```

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 PowerBuilder generates numeric values in the SQL syntax it internally builds in DataWindow objects and sends to your database.

Applies to

JDB JDBC
ODBC

Syntax

The syntax you use depends on the back-end DBMS you are accessing and how you want to format the numeric string.

The following are typical syntax examples for Oracle databases that format a numeric string with a comma as the decimal separator. (See the Examples section for information about how PowerBuilder generates numeric values in the SQL syntax it builds and sends to the database.)

In the PowerBuilder 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 code, you must use the following syntax:

IBM DB2 syntax If you are accessing an IBM DB2 database through the ODBC interface, use the following syntax for NumericFormat. Note the use of *one single quote* at the beginning and end of the string:

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

Oracle JDBC or ODBC syntax If you are accessing an Oracle database through the JDBC or ODBC interface, use the following syntax for NumericFormat. Note the use of *three single quotes* at the beginning and end of the string:

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

Parameter	Description
' '''	**IBM DB2 syntax** Type a single open quote. PowerBuilder returns no open quote in the SQL syntax it builds and sends to the database, as required by IBM DB2 databases. **Oracle, JDBC, or ODBC syntax** Type three single open quotes. PowerBuilder 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. PowerBuilder 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).
%s	Represents one or more digits to the *right of the decimal* in the numeric string. PowerBuilder substitutes this value with the digits to the right of the decimal when it builds the SQL syntax.
' '''	**IBM DB2 syntax** Type one single closed quote. PowerBuilder returns no closed quote in the SQL syntax it builds and sends to the database, as required by IBM DB2 databases. **Oracle, JDBC, or ODBC syntax** Type three single closed quotes. PowerBuilder 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 parameter. Most back-end DBMSs do not require that the driver do special formatting of numeric strings in SQL syntax. However, some databases might 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 PowerBuilder builds in DataWindow objects 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

Example 1 (IBM DB2 syntax) This example shows how to specify that you want PowerBuilder to generate two numeric values in the format *125,50* and *4,0*. PowerBuilder uses the comma as a decimal separator in the SQL syntax it builds in DataWindow objects and sends to an IBM DB2 database.

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

  ```
  %s,%s
  ```

- **Application** Type the following in code:

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

What happens PowerBuilder internally builds the following SQL INSERT statement in the DataWindow object and sends the syntax to your database. PowerBuilder returns no quotes in the SQL syntax.

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

Example 2 (Oracle, JDBC, or ODBC syntax) This example shows how to specify that you want PowerBuilder to generate two numeric values in the format *'125,50'* and *'4,0'*. PowerBuilder uses the comma as a decimal separator in the SQL syntax it builds in DataWindow objects and sends to an Oracle database.

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

  ```
  %s,%s
  ```

- **Application** Type the following in code:

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

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

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

See also

DecimalSeparator

ObjectMode

Description

Allows PowerBuilder to turn off the Oracle Call Interface (OCI) object mode. By default, PowerBuilder sets the mode parameter of OCIInitialize(), the first OCI call in any OCI application, to OCI_OBJECT. When object mode is on, your application can define and use new database object types.

Applies to

O90 Oracle9*i*
O10 Oracle 10*g*
ORA Oracle 11*g*

Syntax

ObjectMode=*value*

Parameter	Description
value	Specifies whether object mode is enabled or not. Values are: • **Yes** (Default) Use object mode. • **No** Do not use object mode.

Default value

Yes

Usage

To turn ObjectMode off, clear the ObjectMode check box on the Connection page of the Database Profile Setup dialog box, or set ObjectMode to “No” in a script.

Examples

To specify that you want ObjectMode disabled:

- **Database profile** Deselect the ObjectMode check box on the Connection page of the Database Profile Setup dialog box.
- **Application** Type the following in code:

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

To specify that you want ObjectMode disabled, clear the ObjectMode check box on the Connection page of the Database Profile Setup dialog box.

ODBCU_CONLIB

Description

Specifies whether EAServer establishes an ODBC connection cache in ANSI or Unicode mode.

This parameter applies *only* when a PowerBuilder custom class user object is deployed to EAServer.

Applies to

ODBC

Syntax

ODBCU_CONLIB =*value*

Parameter	**Description**
value	Specifies whether EAServer establishes an ODBC connection cache in ANSI or Unicode mode. Values are: • **0** (Default) Establish a connection in ANSI mode • **1** Establish a connection in Unicode mode

Default value

ODBCU_CONLIB=0

Usage

Set this parameter to 1 to specify that the EAServer JAG_CM_CONLIB connection cache property should be set to ODBCU to establish an ODBC connection in Unicode mode.

Examples

To specify that you want a Unicode connection cache:

- **Database profile** Select the Enable Unicode Connection check box on the EAServer/COM+ page of the Database Profile Setup dialog box.
- **Application** Type the following in code:

```
SQLCA.DBParm="ODBCU_CONLIB=1"
```

OJSyntax

Description

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

Applies to

ADO.NET
ASE, SYC Sybase Adaptive Server Enterprise
I10 Informix
IN9 Informix
JDB JDBC
ODBC
OLE DB
O90 Oracle9*i*
O10 Oracle 10*g*

SNC SQL Native Client for Microsoft SQL Server

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 for IN9, SNC, ASE, and SYC, OJSyntax=ANSI_ESCAPE for ADO.NET, JDBC, ODBC, and OLE DB, OJSyntax=PB for O90 and O10.

Usage

All PowerBuilder database interfaces provide support for ANSI SQL-92 outer join SQL syntax generation. PowerBuilder supports both left and right outer joins in graphics mode and full outer and inner joins in syntax mode.

You must set the OJSyntax parameter to indicate the version of outer join SQL syntax you want PowerBuilder to generate. For ADO.NET, JDBC, ODBC, and OLE DB, the default is ANSI_Escape and can be reset to ANSI or PB (native). For IN9, SNC, ASE, and SYC, the default is ANSI and can be reset to PB. For O90 and O10, the default is PB, which means use Oracle native outer join syntax, and can be reset to ANSI.

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

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

Define outer joins in the SQL Select painter for portability When you define an outer join SELECT statement graphically in the SQL Select painter, the DataWindow object stores the SQL in pseudocode. At runtime, the outer join syntax is generated based on the current OJSyntax parameter 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 parameter setting does *not* affect the SQL.

Using native outer join syntax The option PB generates native outer join syntax. It is available for ODBC and OLE DB only if PBOuterJoin and PBOuterJoinOperator syntax entries are set in the appropriate SYNTAX section for your DBMS in the *Sybase\Shared\PowerBuilder\pbodb125.ini* file.

The PB option is available for JDBC only if PBOuterJoin and PBOuterJoinOperator syntax entries are set in the Windows registry in the appropriate key for your DBMS in the *HKEY_CURRENT_USER\Software\Sybase\PowerBuilder\11.0\pbjdbc* key. This key is not installed by default. See the *egreg.txt* file in *Sybase\Shared\PowerBuilder* for an example of a registry file you could execute to add or change PowerBuilder JDBC settings for your DBMS.

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 PowerBuilder *Users Guide*.

OJSyntax does not apply to DIR For one database interface, DIR, the database connection always uses ANSI outer join SQL syntax.

Examples

To set the value of OJSyntax:

- **Database profile** Select the appropriate value from the Outer Join Syntax drop-down list on the Syntax page in the Database Profile Setup dialog box.
- **Application** Type the following in code:

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

OnlineIndex

Description

Specifies that the Database painter should use the ONLINE keyword when you create or drop an index on a table so that the index can be created or dropped without locking the table.

When to specify OnlineIndex
You must specify the OnlineIndex parameter *before* connecting to the database.

Applies to

I10 Informix

Syntax

OnlineIndex='*value*'

Parameter	Description
value	Specifies that the Database painter should use the ONLINE keyword when you create or drop an index. Values are: • **0** The Database painter does not use the ONLINE keyword. • **1** The Database painter uses the ONLINE keyword.

Default value

OnlineIndex=0

Usage

In IDS 10.0 and later, the SQL syntax of CREATE INDEX and DROP INDEX supports the ONLINE keyword to create or drop an index in an online environment where the database and its tables are continuously available. When you use the ONLINE keyword to create or drop an index, data definition language (DDL) operations execute without applying an exclusive lock on the table on which the specified index is defined.

If you use CREATE INDEX ONLINE to create an index on a table that other users are accessing, the index is not available until no users are updating the table. If you issue DROP INDEX ONLINE to drop an index, no users can reference the index, but concurrent data manipulation language (DML) operations can use the index until the operations terminate. Dropping the index is deferred until no users are using the index.

You can set the OnlineIndex static DBParm on the System tab page in the Database Profile Setup dialog box for I10 connections to specify that the Database painter should use the ONLINE keyword when you create or drop an index.

Clustered index not supported
You cannot create a clustered index using online mode because it is not supported by IDS.

Examples

To specify that the Database painter use the ONLINE keyword when you create or drop an index in the Database painter:

- **Database profile** Select the Create or Drop Non-Clustered Indexes Without Dropping Tables check box on the System page in the Database Profile Setup dialog box.
- **Application** Type the following in code:

```
SQLCA.DBParm="OnlineIndex=1"
```

OraMTSConFlgs

Description

Specifies the behavior of a transactional PowerBuilder component deployed to COM+. This parameter applies *only* when a PowerBuilder custom class user object is deployed as a COM+ component and is connecting to an Oracle 8.1.5 or higher database.

Deprecated DBParm
You can no longer deploy components from PowerBuilder to COM+ servers. The OraMTSConFlgs DBParm is maintained for backward compatibility to existing PowerBuilder components on COM+ servers. For new development, deploy components as .NET projects instead.

Applies to

O90 Oracle9*i*
O10 Oracle 10*g*
ORA Oracle 11*g*

Syntax

OraMTSConFlgs='*value*'

Parameter	Description
value	Specifies the behavior of a transactional component deployed to COM+. Values are: • **All Default** (Default) Obtains a pooled connection and enlists the connection in any COM+ transaction, if one exists (ORAMTS_CFLG_ALLDEFAULT). • **No Implicit Enlistment** Obtains a pooled connection but does not enlist the resource in any COM+ transaction even if the component is transactional (ORAMTS_CFLG_NOIMPLICIT). • **Unique Server Session** Requests a single OCI session per OCI Server. Since multiplexing is not supported in Version 8.1.5, this option is always used (ORAMTS_CFLG_UNIQUESRVR). • **SYSDBA Login** Required if connecting as SYSDBA (ORAMTS_CFLG_SYSDBALOGN). • **SYSOPER Login** Required if connecting as SYSOPER (ORAMTS_CFLG_SYSOPRLOGN). • **Preliminary INTERNAL Login** Required if connecting as INTERNAL (ORAMTS_CFLG_PRELIMAUTH).

Default value

OraMTSConFlgs='ORAMTS_CFLG_ALLDEFAULT'

Usage

If a transactional PowerBuilder component deployed to COM+ uses a PowerBuilder native interface to connect to an Oracle 8.1.5 or higher database, COM+ attempts to obtain a pooled connection and enlist the connection in a transaction. You can specify different behavior by selecting one or more of the available options.

When the Oracle database interface is running under COM+, ThreadSafe mode is enabled by default and the value of the ThreadSafe parameter is ignored.

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

These values are not mutually exclusive. They are chained using the pipe character in the parameter.

Requirements for COM+ transactional support
Oracle Services for COM+ must be installed and configured.

Examples

To obtain an enlisted connection using the INTERNAL account:

- **Database profile** Select the Preliminary INTERNAL Login check box on the EAServer/COM+ page in the Database Profile Setup dialog box. The All Default check box is selected by default.
- **Application** Type the following in code (use | to signify a logical OR of the flags):

```
SQLCA.DBParm="OraMTSConFlgs='ORAMTS_CFLG_ALLDEFAULT
|ORAMTS_CFLG_PRELIMAUTH'"
```

PackageProcs

Description

Specifies that the stored procedures and functions encapsulated in an Oracle database package should be appended to the lists of Oracle standalone stored procedures and functions displayed in the DataWindow object and Database painters.

When to specify PackageProcs
You must specify the PackageProcs parameter *before* connecting to the database.

Applies to

O90 Oracle9*i*
O10 Oracle 10*g*
ORA Oracle 11*g*

Syntax

PackageProcs=*value*

Parameter	Description
value	Specifies that package-stored procedures and functions should be appended to the lists of stored procedures and functions. Values are: • **0** (Default) Do not append package-stored procedures and functions. • **1** Append package-stored procedures and functions.

Default value

PackageProcs=0

Usage

A package is an encapsulated collection of related program objects (such as procedures, functions, variables, and cursors) stored together in an Oracle database. Listing the objects contained in a package might impose a performance penalty on your Oracle database connection. When displayed in the DataWindow painter, only those objects that contain a REF CURSOR or SELECT statement parameter are listed. When displayed in the Database painter, all objects are listed. The text source displayed is that of the entire package.

Examples

To specify that you want Oracle package objects appended to the lists of stored procedures and functions:

- **Database profile** Select the List Package Subprograms check box on the System page in the Database Profile Setup dialog box.
- **Application** Type the following in code:

```
SQLCA.DBParm="PackageProcs=1"
```

PacketSize (ODBC)

Description

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

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 parameter *before* connecting to the database.

Applies to

ODBC (if ODBC 2.0 or higher driver and back-end DBMS support this feature)

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 page in the Database Profile Setup dialog box:

```
2048
```

- **Application** Type the following in code:

```
SQLCA.DBParm="PacketSize=2048"
```

PacketSize (ASE, DIR, SNC, SYC)

Description

When connecting to a database, specifies the packet size in bytes that you want the server to set for transferring data to and from your PowerBuilder application. A **packet** is a fixed-size chunk of data for sending information over a network.

If the server has space limitations, it sets the packet size to less than the specified PacketSize value. Otherwise, it sets the size equal to the PacketSize value. The default value is 512 bytes.

When to specify PacketSize
You must specify the PacketSize parameter *before* connecting to the database.

Applies to

ASE, SYC Sybase Adaptive Server Enterprise
DIR Sybase DirectConnect
SNC SQL Native Client for Microsoft SQL Server

Syntax

PacketSize=*value*

Parameter	Description
value	A value specifying the packet size in bytes that a database server sets for transferring data to and from your application. The value must be a multiple of 512 bytes (default=512 bytes).

Default value

PacketSize=512 (4096 for SNC)

Usage

When to set If your PowerBuilder application sends or receives large amounts of text or image data from the server, setting the PacketSize value larger than the default 512 bytes might speed performance by causing fewer network read and write operations.

Adaptive Server Enterprise and DirectConnect Before setting PacketSize for use with an Adaptive Server Enterprise or DirectConnect database, you or your system administrator must set the following configuration variables on the server for PacketSize to take effect:

- **Additional netmem** Sets the maximum size of additional memory that can be used for network packets larger than the default size.
- **Maximum network packet size** Sets the maximum network packet size for all database users.

For instructions on setting these configuration variables, see your database documentation.

Examples

To specify that the database server should set the packet size equal to or less than 2048 bytes:

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

```
2048
```

- **Application** Type the following in code:

```
SQLCA.DBParm="PacketSize=2048"
```

PBCatalogOwner

Description

Specifies a nondefault owner for the extended attribute system tables. These five 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, PowerBuilder creates a new set of tables with the owner name you specify.

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

Applies to

ADO.NET
ASE, SYC and SYJ Sybase Adaptive Server Enterprise
DIR Sybase DirectConnect
I10 Informix
JDB JDBC
ODBC
OLE DB
O90 Oracle9*i*
O10 Oracle 10*g*
ORA Oracle 11*g*
SNC SQL Native Client for Microsoft SQL Server

Syntax

PBCatalogOwner='*owner_name*'

Parameter	Description
owner_name	Specifies the owner of the extended attribute system tables. **For DB2 databases** If you use the DB2SYSPB.SQL script to create the extended attribute system tables in a DB2 database and replace all instances of PBOwner in the script with the name of a nondefault table owner, *owner_name* must be the same as the owner specified in the DB2SYSPB.SQL script.

Default value

The default value for PBCatalogOwner depends on the DBMS you are accessing, as follows:

DBMS	PBCatalogOwner default value
ADO.NET	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.
Informix	PBCatalogOwner='informix'
JDBC	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.
ODBC	If a value for PBCatalogOwner is not specified in the database profile or in the PBODB*n*0 initialization file, the default value is the user ID specified in the database profile.

DBMS	PBCatalogOwner default value
OLE DB	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.
Oracle	PBCatalogOwner='SYSTEM'
SNC	PBCatalogOwner='dbo'
Sybase Adaptive Server Enterprise	PBCatalogOwner='dbo'
Sybase DirectConnect	PBCatalogOwner='sqlca.logid'

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.

Informix databases For ANSI-compliant databases, the owner name that you specify must be unique but the table name does not have to be unique. You can create multiple sets of catalog tables prefaced with different user names. However, if the database is not ANSI-compliant, the table name must be unique, so that only one set of catalog tables can be created with an assigned owner name.

JDBC databases When you connect to a JDBC database and a value for PBCatalogOwner is set in both the database profile and the registry, the setting in the profile overrides the setting in the registry.

ODBC data sources When you connect to an ODBC data source and a value for PBCatalogOwner is set in both the database profile and the PBODB125 initialization file, the setting in the profile overrides the setting in the PBODB125 initialization file.

DB2 databases When you connect to a DB2 database, you can use the DB2SYSPB.SQL script to create the extended attribute system tables. If you use the DB2SYSPB.SQL script, keep the following in mind:

- You can edit the script to change all instances of PBOwner to another name, or leave the table owner as PBOwner in the script (the default).

  Specifying SYSIBM is prohibited
  DB2 prohibits you from specifying SYSIBM as the table owner.

- You can set the PBCatalogOwner parameter to the owner you specified in this script or to PBOwner if you did not edit the script.

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 PB Catalog Table Owner box on the System page in the Database Profile Setup dialog box:

```
TEST
```

- **Application** Type the following in code:

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

PBMaxBlobSize

Description

Specifies the maximum blob size that PowerBuilder can read into memory.

When to specify PBMaxBlobSize
You must specify a value for the PBMaxBlobSize parameter *before* connecting to the database.

Applies to

ADO.NET
OLE DB
SNC SQL Native Client for Microsoft SQL Server

Syntax

PBMaxBlobSize=*value*

Default value

PBMaxBlobSize=1024000

Usage

PowerBuilder does not restrict the maximum blob size. Instead, the maximum blob size is determined by the machine on which the application is running. If the blob size exceeds the available memory on the machine on which the application is running, PowerBuilder reads the blob in chunks if the data provider supports the ISequentialStream interface. If the blob size exceeds the default value and the data provider does not support the ISequentialStream interface, PowerBuilder truncates it and reports an out-of-memory error. Use the PBMaxBlobSize parameter to specify larger maximum blob sizes.

Examples

To set the PBMaxBlobSize value to 200000:

- **Database profile** Type the following in the Maximum In-Memory Blob Size box on the Transaction page in the Database Profile Setup dialog box:

```
200000
```

- **Application** Type the following in code:

```
SQLCA.DBParm="PBMaxBlobSize=200000"
```

PBMaxTextSize

Description

Specifies the maximum length of text or large value datatypes returned when you include them in a DataWindow.

You can set the PBMaxTextSize parameter if you want to include a long text string in a DataWindow object without treating the text as a binary large object (blob) datatype.

Applies to

SNC SQL Native Client for Microsoft SQL Server

Syntax

PBMaxTextSize='*value*'

Parameter	Description
value	The maximum length in bytes of text or large value datatypes returned when you include them in a DataWindow. The DataWindow can be created by a procedure. The range of valid values is from 10,000 bytes to 1,000,000 bytes.

Default value

PBMaxTextSize=32767

Usage

The SQL Native Client OLE DB Provider for SQL Server automatically sets the SQL Server TEXTSIZE property to 2147483647 when connecting. The size of data fetched depends on the PBMaxTextSize parameter. If a result set includes a large value type, the blocking factor will be set to 1.

Examples

To have SQLServer return text or large datatypes that are up to 48,000 bytes long when you include them in a SQL SELECT statement:

- **Database profile** Type `48000` in the Maximum String Size in DataWindow box on the Transaction page in the Database Profile Setup dialog box.
- **Application** Type the following in code:

```
SQLCA.DBParm="PBMaxTextSize='48000'"
```

PBNewSPInvocation

Description

Uses an alternative method to invoke a stored procedure.

Applies to

ODBC

Syntax

PBNewSPInvocation='*value*'

Parameter	Description
value	Specifies whether the standard method or an alternative method is used to invoke a stored procedure. Values are: • **No** (Default) Use the standard method to invoke a stored procedure. • **Yes** Use the alternative method to invoke a stored procedure.

Default value

PBNewSPInvocation='No'

Usage

Output parameters might not be returned when you use an embedded SQL command to call a stored procedure. You can set PBNewSPInvocation to ‘Yes' to use an alternative method to invoke a stored procedure. The behavior of the PowerBuilder ODBC driver when this parameter is set is consistent with the default behavior of the OLE DB and JDBC drivers.

If PBNewSPInvocation is set to ‘Yes', the alternative method is used when you retrieve data into a DataWindow object that uses a stored procedure. This parameter has no effect when you use RPC to invoke a stored procedure.

When PBNewSPInvocation is set to ‘Yes', the values of the PBUseProcOwner and CallEscape parameters are ignored.

Examples

To set the parameter for all connections, add the following line to every relevant section (such as `;IBM DB2/NT 2.1 DB2CLI` for a DB2 connection on Windows) in your *pbodb125.ini* file:

```
PBNewSPInvocation='Yes'
```

For more information about editing *pbodb125.ini*, see the Appendix in *Connecting to Your Database*.

You can also set the parameter at runtime. For example:

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

The value that is set at runtime overrides the value in the *pbodb125.ini* file.

To obtain the value of the stored procedure’s output parameter, use the OUTPUT or OUT keyword. For example:

```
DECLARE sp_test PROCEDURE FOR SP1 VAR0=:ARGIN,
VAR1=:ARGOUT OUTPUT USING SQLCA;
```

If the stored procedure contains result sets, you must fetch the result sets first. If the stored procedure has a return value and you want to obtain it, use the format `RC=SP1`:

```
DECLARE sp_test PROCEDURE FOR RC=SP1 VAR0=:ARGIN,
VAR1=:ARGOUT OUTPUT USING SQLCA;
```

See also

DefaultProcOwner

PBNoCatalog

Description

Specifies that PowerBuilder will not check for the existence of catalog tables when creating a DataWindow object at runtime.

When to specify PBNoCatalog
You must specify the PBNoCatalog parameter *before* connecting to the database.

Applies to

O90 Oracle9*i*
O10 Oracle 10*g*
ORA Oracle 11*g*

Syntax

PBNoCatalog='*value*'

Parameter	Description
value	Specifies that PowerBuilder will not check for the existence of catalog tables when creating a DataWindow object at runtime.

Default value

PBNoCatalog=‘NO’

Usage

When a DataWindow object is created dynamically using SyntaxFromSQL and modified so that it can use stored procedure updates, PowerBuilder looks for the PowerBuilder catalog tables. If PBCatalogOwner is not set, PowerBuilder looks for the catalog tables under the owner SYSTEM. The DataWindow created may be different depending on whether PBCatalogOwner was set, and errors may be generated if the tables do not exist under the owner SYSTEM. To prevent PowerBuilder from using the catalog tables, set the PBNoCatalog parameter to true.

This parameter can also be set in the PBODB initialization file for connection to Oracle through the ODBC and OLE DB drivers.

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

Examples

To tell PowerBuilder not to use catalog tables, type the following in code:

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

See also

PBCatalogOwner

PBTrimCharColumns

Description

Specifies whether PowerBuilder 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

ODBC
OLE DB

Syntax

PBTrimCharColumns=*value*

Parameter	Description
value	Specifies whether PowerBuilder should trim trailing spaces from data of type Char, Char for Bit Data, and VarChar for Bit Data. Values are: • **NO** (Default) Do not trim trailing spaces. • **YES** Trim trailing spaces.

Default value

'NO'

Usage

This parameter can only be set in the pbodb125.ini file. For ODBC, you can set the TrimSpaces parameter in the Database Profile Setup dialog box or in code to perform the same function.

By default, PowerBuilder 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 might receive the message `Row changed between retrieve and update` when your DataWindow object's update properties are set to "Key and updateable columns." To prevent this, change your DataWindow object's 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 PBTrimCharColumns='YES'.

Examples

To specify that PowerBuilder should trim trailing spaces, add the following line to the section for the database you are accessing:

```
PBTrimCharColumns='YES'
```

See also

TrimSpaces

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 PowerBuilder should qualify the stored procedure with the owner name in the SQL EXECUTE statement passed to the driver.

PowerBuilder 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 PowerBuilder 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, PowerBuilder 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) PowerBuilder 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

Determining the PBUseProcOwner value PowerBuilder searches the following in this order to determine the PBUseProcOwner value:

1. The section for your database profile in the PowerBuilder initialization file (in the development environment) or the value of the transaction object's DBParm property (in an application).
2. The section for your ODBC driver in the *PBODB125* initialization file.

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

If DBA owns the SQL Anywhere® stored procedure   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 PowerBuilder 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 PowerBuilder 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 page in the Database Profile Setup dialog box.
- **Application**   Type the following in code:

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

PersistEncrypted

Description

Specifies whether the data source you are accessing through the OLE DB interface is allowed to save your encrypted password.

When to specify PersistEncrypted
You must specify the PersistEncrypted parameter *before* connecting to the database.

Applies to

OLE DB

Syntax

PersistEncrypted='*value*'

Parameter	Description
value	Specifies whether the data source can save your encrypted password. Values are: • **True** Tells the data source it can save your password. • **False** (Default) Tells the data source it cannot save your password.

Default value

PersistEncrypted='False'

Examples

To tell the data source you are accessing through OLE DB that it can save your password:

- **Database profile** Select the Persist Encrypted check box on the Security page in the Database Profile Setup dialog box.
- **Application** Type the following in code:

```
SQLCA.DBParm="PersistEncrypted='True'"
```

See also

DataLink
MaskPassword
EncryptPassword
PersistSensitive

PersistSensitive

Description

Specifies whether the data source you are accessing through the OLE DB interface is allowed to save sensitive authentication information, such as a password, along with other authentication information.

When to specify PersistSensitive
You must specify the PersistSensitive parameter *before* connecting to the database.

Applies to

OLE DB

Syntax

PersistSensitive='*value*'

Parameter	Description
value	Specifies whether the data source can save your authentication information. Values are: • **True** Tells the data source it can save your authentication information. • **False** (Default) Tells the data source it cannot save your authentication information.

Default value PersistSensitive='False'

Examples To tell the data source you are accessing through OLE DB that it can save your authentication information:

- **Database profile** Select the Persist Security Info check box on the Security page in the Database Profile Setup dialog box.
- **Application** Type the following in code:

```
SQLCA.DBParm="PersistSensitive='True'"
```

See also MaskPassword
EncryptPassword
PersistEncrypted

PoolCreator

Description Specifies the user name used to create a connection or session pool.

Applies to ORA Oracle 11*g*

Syntax PoolCreator =*value*

Parameter	Description
value	String to specifiy the name of the connection or session pooling creator.

Default value None.

Usage Use in conjunction with the PoolPWD DBParm. If no value is provided for this DBParm, the LogID value is used to create the connection or session pool.

Examples The following example sets the name for a connection pool creator to Scott:

- **Database profile** Select Connection Pooling from the Pooling Type drop-down list on the Pooling page in the Database Profile Setup dialog box, type `Scott` in the Pool Creator text box, and Scott’s password in the Password text box on the same page.

- **Application** Type the following in code:

```
my_trans.dbparm =
  "pooling='connection',poolcreator=
    'Scott',poolpwd='scottspass'"
```

See also

Pooling
PoolPwd

Pooling

Description

Specifies the type of pooling to use with an Oracle database.

Applies to

ORA Oracle 11*g*

Syntax

Pooling =*value*

Parameter	Description
value	Specifies whether to use pooling, and if so, what kind of pooling. Values are: • **Session** For session pooling. • **Connection** For connection pooling. • **None** (Default) For no pooling.

Default value

No pooling.

Usage

With the ORA driver you specify the type of pooling you want with the Pooling DBParm. The O90 and O10 database drivers that you can use in PowerBuilder to connect to the 9.x and 10.x versions of the Oracle DBMS support connection pooling with the DBParm parameter CNNPool. For backward compatibility purposes, the CNNPool parameter is also supported by the ORA driver. However, if you set the Pooling parameter with the ORA driver, the CNNPool parameter is ignored.

If you are using connection or session pooling, the proxy user name is the connection or session pooling creator (which you can provide in the PoolCreator and PoolPwd DBParm parameters), and the Transaction object's LogID is ignored.

If you select session pooling, you can enter a value for the SessionHomogeneous DBParm to authenticate all sessions in the pool with the user name and password in effect when the session pool was created. No proxy session can be created if pooling is set to homogeneous session mode.

Examples

The following example creates a connection pool:

- **Database profile** Select Connection Pooling from the Pooling Type drop-down list on the Pooling page in the Database Profile Setup dialog box.
- **Application** Type the following in code:

```
my_trans.dbparm="pooling='connection'"
```

See also

PoolCreator
PoolPwd

PoolPwd

Description

Specifies the password used to create a connection or session pool.

Applies to

ORA Oracle 11*g*

Syntax

PoolCreator =*value*

Parameter	Description
value	String to specifiy the name of the connection or session pooling creator.

Default value

None

Usage

Use in conjunction with the PoolCreator DBParm. If no value is provided for this DBParm, the LogPass value of the Transaction object is used to create the connection or session pool.

Examples

The following example creates a session pool with the user name “Scott”, and the password “mypass”:

- **Database profile** Select Session Pooling from the Pooling Type drop-down list on the Pooling page in the Database Profile Setup dialog box, type `Scott` in the Pool Creator text box, and `mypass` in the Password text box on the same page.
- **Application** Type the following in code:

```
my_trans.dbparm =   "pooling='session',poolcreator=
  'Scott',poolpwd='mypass'"
```

See also

PoolCreator
Pooling

Properties

Description

Sets properties specific to the particular JDBC driver you are using to connect to the database.

Applies to

JDB JDBC

Syntax

Properties='*property_value*'

Default value

None

Usage

The Driver-Specific Properties box allows you to set properties specific to a particular driver.

For information about the properties supported by your JDBC driver, see the vendor’s documentation.

Define User ID and Password
If properties are defined, you *must* also define the user ID and password in the properties box.

Examples

To set a property for the Sybase jConnect driver:

- **Database profile** Type the following in the Driver-Specify Properties box on the Connection page in the Database Profile Setup dialog box:

```
SQLINITSTRING=set TextSize 32000;
user=sa;password=manager
```

- **Application** Type the following in code:

```
SQLCA.DBParm="Properties='SQLINITSTRING=set
TextSize 32000;user=sa;password=manager'"
```

See also

Driver
URL

ProtectionLevel

Description

Specifies the level of protection applied to data sent between PowerBuilder and the data server through the OLE DB data provider. This parameter applies only to network connections other than Remote Procedure Call (RPC) connections. Similar levels of protection can be specified for authenticated RPC connections.

When to specify ProtectionLevel
You must specify the ProtectionLevel parameter *before* connecting to the database.

Applies to

OLE DB

Syntax

ProtectionLevel='*value*'

Parameter	Description
value	Specifies the level of protection applied to data sent between PowerBuilder and the data server. Values are: • **Not set** No level of protection is selected. • **Call** Authenticates the source of the data at the beginning of each request from the client to the server. • **Connect** Authenticates only when the client establishes the connection with the server. • **None** Performs no authentication of data. • **Packet** Authenticates that all data received is from the client. • **Packet (Integrity)** Authenticates that all data received is from the client and that it has not been changed in transit. • **Packet (Privacy)** Authenticates that all data received is from the client, that it has not been changed in transit, and that it protects the privacy of the data by encrypting it.

Default value

Not set

Examples

To set a level of protection for the data sent between PowerBuilder and the data server:

- **Database profile** On the Security page in the Database Profile Setup dialog box, select Connect from the Protection Level drop-down list.
- **Application** Type the following in code:

```
SQLCA.DBParm=
"PROTECTIONLEVEL='DB_PROT_LEVEL_CONNECT'"
```

See also

DataLink

Provider

Description

Identifies the data provider you want to use to connect to your data source.

When to specify Provider
You must specify the Provider parameter *before* connecting to the database.

Applies to

ADO.NET
OLE DB
SNC SQL Native Client for Microsoft SQL Server

Syntax

Provider='*provider_name*'

Default value

None

Usage

Select a data provider from the list of installed data providers displayed in the Provider drop-down list. For example, if you are using Microsoft's OLE DB Provider for ODBC, select MSDASQL as the Provider value. If you are using Microsoft's OLE DB Provider for SQL Server, select SQLOLEDB as the Provider value.

For more information, see the documentation provided by your OLE DB or ADO.NET data provider.

SNC SQL Native Client driver
The Provider DBParm parameter for the Microsoft SQL Native Client (SNC) interface allows you to select the SNC version that you want to use for a database connection. You can set this parameter in script to SQLNCLI (for the SNC 9.0 driver that connect to SQL Server 2005) or to SQLNCLI10 (for the SNC 10.0 driver that connects to SQL Server 2008). Otherwise, you can select one of these providers on the Connection tab of the Database Profile Setup dialog box for the SNC interface.

If you do not set or select a provider, the default selection is SQLNCLI (SNC 9.0 for SQL Server 2005). This allows existing SNC interface users to be able to migrate to the current release of PowerBuilder without any modifications. If PowerBuilder fails to connect with the SQLNCLI provider, it will attempt to connect to SQLNCLI10 provider. However, if you explicitly set the provider and the connection fails, PowerBuilder displays an error message.

Examples

Example 1 To use the Microsoft OLE DB Provider for ODBC to connect to the EAS Demo DB:

- **Database profile** Select MSDASQL from the Provider drop-down list on the Connection page in the Database Profile Setup dialog box for OLE DB.
- **Application** Type the following in code:

```
SQLCA.DBParm="Provider='MSDASQL'"
```

Example 2 To use the Microsoft OLE DB Provider for Oracle to connect to an Oracle 8 database:

- **Database profile** Select MSDAORA from the Provider drop-down list on the Connection page in the Database Profile Setup dialog box.
- **Application** Type the following in code:

```
SQLCA.DBParm="Provider='MSDAORA'"
```

Example 3 To use the Sybase Oracle8 ADO Provider to connect to an Oracle 8 database:

- **Database profile** Select Sybase.Oracle8ADOProvider from the Provider drop-down list on the Connection page in the Database Profile Setup dialog box for ADO.NET.
- **Application** Type the following in code:

```
SQLCA.DBParm="PROVIDER='Sybase.Oracle8ADOProvider'"
```

Example 4 To use the Microsoft SNC Provider to connect to a Microsoft SQL Server 2008 database:

- **Database profile** Select SQLNCLI10 from the Provider drop-down list on the Connection page in the Database Profile Setup dialog box.
- **Application** Type the following in code:

```
SQLCA.DBParm="Provider='SQLNCLI10'"
```

See also

DataLink
DataSource

ProviderString

Description

A string containing provider-specific extended connection information. Use of this database parameter requires that you know how this string will be interpreted and used by the provider. You should use this parameter only for provider-specific connection information that cannot be explicitly described by other database parameters.

When to specify ProviderString
You must specify the ProviderString parameter *before* connecting to the database.

Applies to

ADO.NET
OLE DB

SNC SQL Native Client for Microsoft SQL Server

Syntax

ProviderString='*value*'

Default value

None

Usage

OLE DB applications can initialize data source objects using two methods: IDBInitialize::Initialize and IDataInitialize::GetDataSource.

Using IDBInitialize::Initialize, a provider string can be used to initialize connection properties by setting the DBPROP_INIT_PROVIDERSTRING property in the DBPROPSET_DBINIT property set. An initialization string can also be passed to the IDataInitialize::GetDataSource method to initialize connection properties.

For the OLE DB interface, the ProviderString parameter is used with IDBInitialize::Initialize. For the SNC interface, the ProviderString parameter is used with IDataInitialize::GetDataSource.

Both methods initialize the same OLE DB connection properties, but they use different sets of keywords. For lists of keywords, see the information about OLE DB provider connection string keywords in the Microsoft documentation at http://msdn.microsoft.com/en-us/library/ms130822.aspx.

Examples

Since Microsoft SQL Server supports multiple instances of a database on a single server, you must identify the specific database to which you want to connect by entering the database name. For the ADO.NET and SNC interfaces, you should set the Database parameter on the Connection page in the Database Profile Setup dialog box. For OLE DB, you can use the ProviderString parameter.

To identify a specific Microsoft SQL Server database named ts3:

- **Database profile** Enter the following in the Extended Properties box on the Connection page in the Database Profile Setup dialog box for OLE DB:

  ```
  Database=ts3
  ```

- **Application** Type the following in code for an OLE DB connection:

  ```
  SQLCA.DBParm="ProviderString='database=ts3'"
  ```

See also

URL

ProxyUserName

Description

Specifies that you want EAServer to retrieve a connection from a connection cache by proxy.

This parameter applies *only* when a PowerBuilder custom class user object is deployed as an EAServer component.

Applies to

JDB JDBC
ODBC
SYJ Sybase Adaptive Server Enterprise

Syntax

ProxyUserName='*value*'

Parameter	Description
value	Specifies the alternate login name that can use a connection.

Default value

None

Usage

Regardless of whether you access a cache by user or name, you can retrieve a connection by proxy. Retrieving a connection by proxy means that you can assume the identity and privileges of another user by providing an alternative login name.

This feature can be used with any database that recognizes the SQL command **set session authorization**. In order for user A to use the ProxyUserName parameter to assume the identity of another user B, user A must have permission to execute this statement. For example, for SQL Anywhere, user A must have DBA authority, and for Adaptive Server Enterprise, user A must have been granted permission to execute **set session authorization** by a System Security Officer.

Using the SYJ interface

Sybase EAServer uses a slightly different version of the CT-Lib software. Therefore, *at runtime*, you need to use the SYJ database interface rather than ASE or SYC to connect to an Adaptive Server Enterprise database. The SYJ Database Profile Setup dialog box provides a convenient way to set the appropriate connection parameters and then copy the syntax from the Preview page into the script for your Transaction object.

You cannot use the SYJ interface, however, to connect to the database in the PowerBuilder development environment. Therefore, *during the development phase* (before the component has been deployed to EAServer), you must use ASE or SYC to connect to the database.

For information on how to use PowerBuilder to build EAServer components and enable set-proxy support, see *Application Techniques*.

Examples

On the EAServer page in the Database Profile Setup dialog box, enter the alternative login name in the Proxy User Name box. The PowerScript syntax for the ProxyUserName parameter displays on the Preview page:

```
SQLCA.DBParm="ProxyUserName='pikachu'"
```

Copy the syntax from the Preview page into your script.

See also

CacheName

PWDialog

Description

Controls whether a Password Expired dialog box displays in an application at runtime if a user’s password has expired.

When PWDialog is set to 1, the Password Expired dialog box prompts users to change their passwords if they attempt to log in to the database with an expired password. By default, PWDialog is set to 0 to specify that the Password Expired dialog box does not display in your application at runtime.

The setting of PWDialog affects applications only at runtime. It has no effect in the development environment because, regardless of the PWDialog setting, the Change Password dialog box displays in the development environment to prompt users to change an expired password.

When to specify PWDialog
You must specify a value for PWDialog *before* connecting to the database.

Applies to

ASE, SYC Sybase Adaptive Server Enterprise
O90 Oracle9*i*
O10 Oracle 10*g*
ORA Oracle 11*g*

Syntax

PWDialog=*value*

Parameter	Description
value	Specifies whether the Password Expired dialog box displays in an application at execution time to prompt the user to change an expired login password. Values are: • **0** (Default) Do not display the Password Expired dialog box at execution time. • **1** Display the Password Expired dialog box at execution time to prompt the user to change an expired password.

Default value

PWDialog=0

Usage

When to use Setting PWDialog to 1 to display the Password Expired dialog box in your application provides a convenient way for you to notify your users that a password has expired and allow them to change it.

What happens When the Password Expired dialog box displays in your application at runtime, it notifies users that the password for their login ID has expired and prompts them to supply a new password. For example, for Adaptive Server Enterprise, the sp_password system stored procedure runs to set the new password. Once the password has been changed, the database connection succeeds.

If the user clicks Cancel to close the Password Expired dialog box without changing the password, the database connection fails and a message displays indicating that the password has expired.

Examples

To display the Password Expired dialog box when needed in your application:

- **Database profile** Although the setting of PWDialog has no effect in the development environment, you might want to set it in your database profile to generate connection syntax on the Preview page that you can copy into your code. Select the Display Runtime Dialog When Password Expires check box (for ASE or SYC connections) or the Password Expiration Dialog check box (for Oracle connections) on the Connection page in the Database Profile Setup dialog box.
- **Application** Type the following in code:

```
SQLCA.DBParm="PWDialog=1"
```

PWEncrypt

Description

PWEncrypt specifies whether you want Open Client to automatically encrypt your password when connecting to a Sybase Adaptive Server Enterprise database in PowerBuilder.

When to specify PWEncrypt
You must specify the PWEncrypt parameter *before* connecting to the database.

Applies to

ASE, SYC Sybase Adaptive Server Enterprise

Syntax

PWEncrypt='*value*'

Parameter	Description
value	Specifies whether you want the Open Client software to encrypt your password. Values are: • **Yes** (Default) Tells Open Client to encrypt the password by setting the CS_SEC_ENCRYPTION connection property to CS_TRUE. • **No** Tells Open Client not to encrypt the password by setting the CS_SEC_ENCRYPTION connection property to CS_FALSE.

Default value

PWEncrypt='Yes'

Examples

To tell Open Client not to encrypt your password when connecting to a Sybase Adaptive Server Enterprise database in PowerBuilder:

- **Database profile** Clear the Encrypt Password check box on the Network page in the Database Profile Setup dialog box.
- **Application** Type the following in code:

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

PWExpDialog

Description

Controls whether an informational dialog box displays in an application at runtime if a user’s password is about to expire.

When PWExpDialog is set to 1, a dialog box displays advising users that their passwords will expire in a given number of days. By default, PWExpDialog is set to 0 to specify that the dialog box does not display in your application at runtime.

When to specify PWExpDialog
You must specify a value for PWExpDialog *before* connecting to the database.

Applies to

ASE, SYC Sybase Adaptive Server Enterprise
O90 Oracle9*i*

Syntax

PWExpDialog=*value*

Parameter	Description
value	Specifies whether an informational dialog box displays in an application at runtime to advise the user that a login password will expire in a given number of days. Values are: • **0** (Default) Do not display the dialog box at runtime. • **1** Display the dialog box at runtime.

Default value

PWExpDialog=0

Usage

When this parameter is set to 1, the Oracle ORA-28002 and ORA-28011 and the Adaptive Server 4023 informational messages display.

Examples

To display the dialog box when needed in your application, type the following in code:

```
SQLCA.DBParm="PWExpDialog=1"
```

QualifyPublic

Description

Specifies that the PUBLIC qualifier prepended to Oracle synonyms belonging to the public schema or user group is retained in the SQL Select table list.

When to specify QualifyPublic
You must specify the QualifyPublic parameter *before* connecting to the database.

Applies to

O90 Oracle9*i*
O10 Oracle 10*g*
ORA Oracle 11*g*

Syntax

QualifyPublic=*value*

Parameter	Description
value	Specifies that the PUBLIC qualifier should be retained in the SQL Select table list. Values are: • **0** (Default) Do not retain the PUBLIC qualifier. • **1** Do retain the PUBLIC qualifier.

Default value

QualifyPublic=0

Usage

PowerBuilder's default behavior has been to discard the PUBLIC qualifier so that the object reference is generalized in the generated SQL statement, facilitating the deployment of an application from a development database instance to a production database. However, in certain DataWindow objects, the absence of the PUBLIC qualifier breaks the association of the synonym with its extended attributes, preventing these attributes from being used. The QualifyPublic parameter allows you to specify whether the PUBLIC qualifier should be retained.

Examples

To specify that you want the PUBLIC qualifier to be retained in the SQL Select table list:

- **Database profile** Select the Qualify Public Synonyms check box on the System page in the Database Profile Setup dialog box.
- **Application** Type the following in code:

```
SQLCA.DBParm="QualifyPublic=1"
```

RecheckRows

Description

Rechecks the number of rows affected by an INSERT, UPDATE, or DELETE command and returns it in the SQLNRows property of the Transaction object.

Applies to

ADO.NET
OLE DB
SNC SQL Native Client for Microsoft SQL Server

Syntax

RecheckRows=*value*

Parameter	Description
value	Specifies whether the value of the Microsoft SQL Server @@rowcount property is checked and returned in the SQLNRows property of the Transaction object. Values are: • **0** (Default) Do not recheck the count of affected rows. • **1** Recheck the count of affected rows.

Default value

RecheckRows=0

Usage

In Microsoft SQL server, if a table has an insert, update, or delete trigger, the number of affected rows returned to the SQLNRows property of the Transaction object after an INSERT, UPDATE, or DELETE command depends on the driver. With an ADO.NET driver, the value returned is the sum of the rows affected by the command itself and the trigger.

When you are connected to Microsoft SQL Server using ADO.NET or OLE DB, you can set the RecheckRows runtime database parameter to 1 to recheck how many rows of data were affected by the INSERT, UPDATE, or DELETE command itself and return that value in the SQLNRows property.

Setting RecheckRows to 1 before issuing an INSERT, UPDATE, or DELETE command causes a SELECT @@ROWCOUNT command to be executed. To improve performance, you should set it only when required, and reset it to the default value of 0 after use.

Examples

To set RecheckRows to 1, type the following in code:

```
SQLCA.DBParm="RecheckRows=1"
```

Release

Description

Specifies what version of Sybase Open Client Client-Library (CT-Lib) software is in use on the client workstation.

When to specify Release
You must specify a value for Release *before* connecting to the database.

Applies to

ASE, SYC Sybase Adaptive Server Enterprise

Syntax

Release='*value*'

Parameter	Description
value	Specifies the version of Open Client Client-Library your application uses. The value corresponds to the value of the CS_VERSION property that is used to allocate a context structure for the client. Multiple values that specify the same client context are provided for backwards compatibility. Release number 15 or higher for ASE. Release number 15 or lower for SYC: • **10.x** CS_VERSION_100 • **11** CS_VERSION_110 • **11.5** CS_VERSION_110 • **12** CS_VERSION_110 • **12.5** CS_VERSION_125 • **12.5.1** CS_VERSION_125 • **15** CS_VERSION_15 • **15.5** CS_VERSION_155

Default value

Release='15' for ASE, Release='11' for SYC

Usage

The Release parameter must correspond to the version of Open Client software installed on the client workstation. For example, do not specify 12.5 or 12.5.1 if your Open Client version is 12.0, even if your Adaptive Server version is 12.5 or 12.5.1.

To use Adaptive Server 15, you must install Open Client version 15 on the client computer and set the Release parameter to 15 to establish an Open Client 15 client context.

The Open Client context is allocated by the first *PBSYC125.DLL* database connection. This context acts as the parent context for all subsequent *PBSYC125.DLL* connections. Therefore, you must specify the same value for the Release parameter for all your connections.

If the client library is not loaded, the version that corresponds to the value of Release is loaded.If the value of Release is 15 and that version is not available on the client, the library fails to load. If the value of Release is less than 15 and that version is not available, *PBSYC125.DLL* attempts to load version 15 of the client library and fails only if neither the requested version nor version 15 is available. If there is no active client context, *PBSYC125.DLL* connects using the value of the Release parameter. If there is an active client context, the new connection fails if the value of the Release parameter is higher than the current client context. The new connection succeeds if the value of the Release parameter is the same as or lower than the current client context.

The SYC driver links to the appropriate version of the client libraries dynamically and the Open Client context is released when all connections are closed. If you open multiple connections, the first Open Client context established is used for all of them. If you need to establish a new Open Client context in the development environment, close all open connections and establish a new connection with the Release parameter set to the context you require.

During each database login, *PBSYC125.DLL* automatically determines the version of Adaptive Server being accessed. It customizes its behavior to optimize performance and features for the combination of the Adaptive Server version and the Open Client context specified in the Release parameter. Specifying a value for Release that does not correspond to the Open Client software on the client can cause unpredictable results.

The values 12.5 and 12.5.1 both open an Open Client 12.5 context. However, you should always specify Release='12.5' if you are using Open Client 12.5 and Release='12.5.1' if you are using Open Client 12.5.1. This ensures that *PBSYC125.DLL* correctly handles the following scenarios that require Open Client 12.5.1 and Adaptive Server 12.5.1:

- Use the Date and Time datatypes introduced in Adaptive Server 12.5.1 in RPC calls that explicitly call for these datatypes in the stored procedure argument list.
- Use the Date and Time datatypes in `Update where current of` and `Delete where current of` statements.

Retrieval, insert, update, and delete processing work correctly against Date and Time datatypes using any Open Client software and Adaptive Server 12.5.1 or later. In the Database painter, the Date and Time datatypes display in the list of metadata types when you are connected to an Adaptive Server 12.5.1 server in any Open Client context.

Certain other features are supported only when you access a specified version of a SQL Server 10/11 or Adaptive Server Enterprise database using its associated Open Client software. For example, you must:

- Set the Release parameter to 11 or higher and use Open Client 11.x or higher and Adaptive Server 11.x or higher to take advantage of network-based security and directory services in your application.
- Set Release to 12.5 and use Open Client 12.5 or higher and Adaptive Server 12.5 or higher to access Char and VarChar columns with more than 255 characters.
- Set Release to 15 and use Open Client 15 or higher and Adaptive Server 15 or higher to access the UniText and 64-bit integer (BigInt) SQL datatypes added in version 15 of Adaptive Server.
- Set Release to 15.5 and use Open Client 15.5 or higher to access the BIGTIME and BIGDATETIME SQL datatypes added in version 15.5 of Adaptive Server.

Examples

To specify that your PowerBuilder application accesses an Adaptive Server Enterprise 15 database using an Open Client Client-Library 15 context:

- **Database profile** Select 15 from the Release drop-down list on the Connection page in the Database Profile Setup dialog box.
- **Application** Type the following in code:

```
SQLCA.DBParm="Release='15'"
```

ReleaseConnectionOption

Description

Specifies how EAServer should behave when it releases control of a connection. This parameter applies *only* when a PowerBuilder custom class user object is deployed as an EAServer component.

Applies to

JDB JDBC
ODBC
O90 Oracle9*i*
O10 Oracle 10*g*
ORA Oracle 11*g*
SYJ Sybase Adaptive Server Enterprise

Syntax

ReleaseConnectionOption='*value*'

Parameter	Description
value	Specifies how EAServer should behave when it releases control of a connection. Values are: • **JAG_CM_UNUSED** (Default) A connection taken from a cache is placed back in the cache and a connection created outside of a cache is closed and destroyed. • **JAG_CM_DROP** The connection is forced to close and is deallocated. If the connection came from a cache, a new connection is created in its place.

Default value

ReleaseConnectionOption='JAG_CM_UNUSED'

Usage

Use JAG_CM_DROP to destroy a connection when errors have made it unusable. This parameter cannot be set dynamically. The value set when the connection is made remains in effect until it is disconnected.

Using the SYJ interface
Sybase EAServer uses a slightly different version of the CT-Lib software. Therefore, *at runtime*, you need to use the SYJ database interface rather than ASE or SYC to connect to an Adaptive Server Enterprise database. The SYJ Database Profile Setup dialog box provides a convenient way to set the appropriate connection parameters and then copy the syntax from the Preview page into the script for your Transaction object.

You cannot use the SYJ interface, however, to connect to the database in the PowerBuilder development environment. Therefore, *during the development phase* (before the component has been deployed to EAServer), you must use ASE or SYC to connect to the database.

For information on how to use PowerBuilder to build EAServer components, see *Application Techniques*.

Examples

On the EAServer page in the Database Profile Setup dialog box, select JAG_CM_DROP from the Release Connection Option drop-down list. The PowerScript syntax for the ReleaseConnectionOption parameter displays on the Preview page:

```
SQLCA.DBParm="ReleaseConnectionOption='JAG_CM_DROP'"
```

Copy the syntax from the Preview page into your script.

See also

CacheName
GetConnectionOption
UseContextObject

Request

Description

Specifies whether to allocate new transaction resources each time the client application sends a request and then release those resources after each request.

Applies to

DIR Sybase DirectConnect (applies only to Access Service for DB2/MVS)

Syntax

Request=*value*

Parameter	Description
value	Specifies whether you want to release transaction resources after each request. Values are: • **0** (Default) Maintain resources for the duration of the database connection. • **1** Release resources after each request.

Default value

Request=0

Usage

Requirements for using the Request parameter Setting the Request parameter to 1 to release resources has an effect only when you do both of the following:

- Set the AutoCommit database preference to True to specify that PowerBuilder should issue SQL statements outside the scope of a transaction. (See the description of AutoCommit.)
- Specify the value for Request *before* connecting to a database.

What happens When you set the Request parameter to 1, transaction resources are allocated for each request and released when the request finishes. This might slow the performance of your application, but it allows more simultaneous users of the system.

Examples

To specify that you want to release resources after each request:

- **Database profile** Select the Release Transaction Resources After Each Request check box on the Transaction page in the Database Profile Setup dialog box.
- **Application** Type the following in code:

```
SQLCA.DBParm="Request=1"
```

See also

AutoCommit

RPCRebind

Description

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

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

Applies to

ODBC

Syntax

RPCRebind=*value*

Parameter	Description
value	Specifies whether you want PowerBuilder 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, PowerBuilder 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 the parameter size (as returned from the back-end database) is used instead of the variable size. Failure to do this 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 PowerBuilder should rebind RPC parameters:

- **Database profile** Select the RPC Rebind check box on the Transaction page in the Database Profile Setup dialog box.
- **Application** Type the following in code:

```
SQLCA.DBParm="RPCRebind=1"
```

ReturnCommandHandle

Description

Specifies whether you want PowerBuilder to return a handle to a session object or a data source object when you call the DBHandle function.

Applies to

OLE DB

Syntax

ReturnCommandHandle =*value*

Parameter	Description
value	Specifies whether you want PowerBuilder to return a handle to a session object or a data source object. Values are: • **0** (Default) Return a data source object pointer. • **1** Return a session object pointer.

Default value

ReturnCommandHandle=0

Usage

DBHandle takes a transaction object as a parameter and returns a long variable that is an interface pointer to a data source object or a session object. By default the OLE DB interface returns the handle of a data source object. If ReturnCommandHandle is set to 1, the OLE DB interface returns the handle of a session object. This handle can then be passed to an external program and instantiated for use in enlisting the connection in a Microsoft DTC (Distributed Transaction Coordinator) transaction.

Examples

For an example, see “DBHandle, OLE DB example” in the online Help.

RTConnBalancing

Description

Supports the runtime connection load balancing feature.

Applies to

ORA Oracle 11*g*

Syntax

RTConnBalancing =*value*

Parameter	Description
value	Specifies whether the load balancing feature is used. Values are: • **True** (Default) The load balancing feature is used if session pooling is set. • **False** The load balancing feature is not used.

Default value

True, but this value is ignored if session pooling is not set.

Usage

Runtime connection load balancing routs work requests to the sessions in a session pool that best serve the work. Runtime connection load balancing is enabled by default when an Oracle 11.1 or higher client is connected to a release 10.2 or higher Oracle server using OCI session pooling.

Examples

The following code disables the load balancing feature in a session pool:

- **Database profile** Select Session Pooling from the Pooling Type drop-down list on the Pooling page in the Database Profile Setup dialog box and clear the Runtime Connection Load Balancing for Session Pooling check box on the same page.
- **Application** Type the following in code:

```
my_trans.dbparm ="pooling='session',
   rtconnbalancing='No'"
```

See also

Pooling

Scroll

Description

Specifies whether you want to use a scroll cursor when connecting to an Informix database in PowerBuilder. When you fetch rows in an Informix table, using a scroll cursor enables you to fetch the next row, previous row, first row, or last row.

By default, PowerBuilder does not use scroll cursors in an Informix database connection.

Applies to

I10 Informix
IN9 Informix

Syntax

Scroll=*value*

Parameter	Description
value	Specifies whether you want to use a scroll cursor when connecting to an Informix database in PowerBuilder. Values are: • **0** (Default) Do not use a scroll cursor. • **1** Use a scroll cursor.

Default value

Scroll=0

Examples

To specify that you want to use a scroll cursor when connecting to an Informix database in PowerBuilder:

- **Database profile** Select the Use A Scroll Cursor check box on the Transaction page in the Database Profile Setup dialog box.
- **Application** Type the following in code:

```
SQLCA.DBParm="Scroll=1"
```

Sec_Channel_Bind

Description

When you access a Sybase Adaptive Server Enterprise database in PowerBuilder through Open Client, Sec_Channel_Bind is one of several parameters that support login authentication for network-based security services. (For other login authentication parameters, see the See Also section.)

Sec_Channel_Bind controls whether your connection's security mechanism performs channel binding. When Sec_Channel_Bind is set to 1, both Sybase Open Client Client-Library (CT-Lib) and the server provide a network channel identifier to the security mechanism before connecting. The channel identifier contains the network addresses of the client and server.

When Sec_Channel_Bind is set to 0 (the default), no channel binding is performed.

You must specify a value for Sec_Channel_Bind *before* connecting to the database.

Using third-party security mechanisms
For information about the third-party security mechanisms and operating system platforms that Sybase has tested with Open Client security services, see the Open Client documentation.

Applies to

ASE, SYC Sybase Adaptive Server Enterprise

Syntax

Sec_Channel_Bind=*value*

Parameter	Description
value	Specifies whether your connection's security mechanism performs channel binding. Values are: • **0** (Default) Do *not* perform channel binding. You can also specify `'No'` or `'False'` to set this value. • **1** Perform channel binding. Both CT-Lib and the server provide a channel identifier to the connection's security mechanism. You can also specify `'Yes'` or `'True'` to set this value.

Default value

Sec_Channel_Bind=0

Usage

Not supported with CyberSafe Kerberos Sec_Channel_Bind is *not supported* if your security mechanism is CyberSafe Kerberos.

Set Release parameter For this parameter to take effect, you *must* also set the Release parameter to 11or higher to specify that your application should use the appropriate version of Sybase Open Client Client-Library (CT-Lib) behavior. See the description of the Release parameter for more information.

Requirements for use To use Sec_Channel_Bind or any other parameter supporting Open Client, you must meet certain requirements for using security services in your PowerBuilder application. For details, see “Requirements for using Open Client security services” in *Connecting to Your Database*.

Corresponding CT-Lib connection property Specifying a value for Sec_Channel_Bind sets the corresponding Sybase CT-Lib connection property named CS_SEC_CHANBIND.

Examples

To specify that your connection's security mechanism performs channel binding:

- **Database profile** Select the Enable Channel Binding check box on the Security page in the Database Profile Setup dialog box.
- **Application** Type the following in code:

```
SQLCA.DBParm="Sec_Channel_Bind=1"
```

See also

Sec_Cred_Timeout
Sec_Delegation
Sec_Keytab_File
Sec_Mechanism
Sec_Mutual_Auth
Sec_Network_Auth
Sec_Server_Principal

Sec_Sess_Timeout

Sec_Confidential

Description

When you access a Sybase Adaptive Server Enterprise database in PowerBuilder through Open Client, Sec_Confidential is one of several parameters that support per-packet security for network-based security services. (For other per-packet security parameters, see the See Also section.)

Sec_Confidential controls whether transmitted data is encrypted. When Sec_Confidential is set to 1, all requests sent to the server and all results returned by the server are encrypted.

When Sec_Confidential is set to 0 (the default), transmitted data is not encrypted.

You must specify a value for Sec_Confidential *before* connecting to the database in PowerBuilder.

Using third-party security mechanisms
For information about the third-party security mechanisms and operating system platforms that Sybase has tested with Open Client security services, see the Open Client documentation.

Applies to

ASE, SYC Sybase Adaptive Server Enterprise

Syntax

Sec_Confidential=*value*

Parameter	Description
value	Specifies whether transmitted data is encrypted. Values are: • **0** (Default) Do *not* encrypt transmitted data. You can also specify `'No'` or `'False'` to set this value. • **1** Encrypt transmitted data. Requests sent to the server and results returned by the server are encrypted. You can also specify `'Yes'` or `'True'` to set this value.

Default value

Sec_Confidential=0

Usage

When to use Encryption can protect your data if you are sending it over a public network to a nonsecure server. In a networked environment, you might want to set Sec_Confidential to 1 to ensure that all requests sent to the server and all results returned by the server are encrypted.

Set Release parameter For this parameter to take effect, you *must* also set the Release parameter to 11or higher to specify that your application should use the appropriate version of Sybase Open Client Client-Library (CT-Lib) behavior. See the description of the Release parameter for more information.

Requirements for use To use Sec_Confidential or any other parameter supporting Open Client security services, you must meet certain requirements for using security services in your PowerBuilder application. For details, see “Requirements for using Open Client security services” in *Connecting to Your Database*.

Corresponding CT-Lib connection property Specifying a value for Sec_Confidential sets the corresponding Sybase CT-Lib connection property named CS_SEC_CONFIDENTIALITY.

Examples

To specify that transmitted data is encrypted:

- **Database profile** Select the Encrypt All Results check box on the Security page in the Database Profile Setup dialog box.
- **Application** Type the following in code:

```
SQLCA.DBParm="Sec_Confidential=1"
```

See also

Release
Sec_Data_Integrity
Sec_Data_Origin
Sec_Replay_Detection
Sec_Seq_Detection

Sec_Cred_Timeout

Description

When you access a Sybase Adaptive Server Enterprise database in PowerBuilder through Open Client, Sec_Cred_Timeout is one of several parameters that support login authentication for network-based security services. (For other login authentication parameters, see the See Also section.)

Some security mechanisms allow applications to set credential timeout values for connections that use network-based login authentication. Sec_Cred_Timeout specifies the number of seconds remaining before a user’s network credentials expire and become invalid. Users obtain network credentials when they log in to the network.

By default, Sec_Cred_Timeout specifies that there is no credential timeout limit—the credentials do not expire.

You must specify a value for Sec_Cred_Timeout *before* connecting to the database in PowerBuilder.

Using third-party security mechanisms
For information about the third-party security mechanisms and operating system platforms that Sybase has tested with Open Client security services, see the Open Client documentation.

Applies to

ASE, SYC Sybase Adaptive Server Enterprise

Syntax

Sec_Cred_Timeout=*value*

Parameter	Description
value	Specifies the number of seconds remaining before a user's network credentials expire and become invalid. You can also specify `'no_limit'` (the default) to specify that the credentials not expire. A credential timeout value set by the security system's administrator supersedes any value you specify for Sec_Cred_Timeout.

Default value

Sec_Cred_Timeout='no_limit'

Usage

CyberSafe Kerberos If your security mechanism is CyberSafe Kerberos, Sec_Cred_Timeout cannot override the installation default value set for credential timeout.

Set Release parameter For this parameter to take effect, you *must* also set the Release parameter to 11or higher to specify that your application should use the appropriate version of Sybase Open Client Client-Library (CT-Lib) behavior. See the description of the Release parameter for more information.

Requirements for use To use Sec_Cred_Timeout or any other parameter supporting Open Client security services, you must meet certain requirements for using security services in your PowerBuilder application. For details, see "Requirements for using Open Client security services" in *Connecting to Your Database*.

Corresponding CT-Lib connection property Specifying a value for Sec_Cred_Timeout sets the corresponding Sybase CT-Lib connection property named CS_SEC_CREDTIMEOUT.

Examples

To specify 120 seconds (2 minutes) remaining before a user's network credentials expire:

- **Database profile** Type 120 in the Credential Timeout box on the Security page in the Database Profile Setup dialog box.

- **Application** Type the following in code:

```
SQLCA.DBParm="Sec_Cred_Timeout=120;Release=15"
```

See also

Release
Sec_Channel_Bind
Sec_Delegation
Sec_Keytab_File
Sec_Mechanism
Sec_Mutual_Auth
Sec_Network_Auth
Sec_Server_Principal
Sec_Sess_Timeout

Sec_Data_Integrity

Description

When you access a Sybase Adaptive Server Enterprise database in PowerBuilder through Open Client, Sec_Data_Integrity is one of several parameters that support per-packet security for network-based security services. (For other per-packet security parameters, see the See Also section.)

Sec_Data_Integrity controls whether your connection's security mechanism checks the integrity of data transmitted to and from the server. When Sec_Data_Integrity is set to 1, the security mechanism analyzes all packets to ensure that their content was not modified during transmission.

When Sec_Data_Integrity is set to 0 (the default), no integrity checking is performed.

You must specify a value for Sec_Data_Integrity *before* connecting to the database in PowerBuilder.

Using third-party security mechanisms
For information about the third-party security mechanisms and operating system platforms that Sybase has tested with Open Client security services, see the Open Client documentation.

Applies to

ASE, SYC Sybase Adaptive Server Enterprise

Syntax

Sec_Data_Integrity=*value*

Parameter	Description
value	Specifies whether your connection's security mechanism performs integrity checking on data transmitted to and from the server. Values are: • **0** (Default) Do *not* check data integrity. You can also specify `'No'` or `'False'` to set this value. • **1** Check data integrity by analyzing all packets to ensure that their content was not modified during transmission. You can also specify `'Yes'` or `'True'` to set this value.

Default value

Sec_Data_Integrity=0

Usage

When to use Your connection's security mechanism can check data integrity only when your connection is also using network-based login authentication. For information, see your Sybase Open Client/Server documentation.

Set Release parameter For this parameter to take effect, you *must* also set the Release parameter to 11or higher to specify that your application should use the appropriate version of Sybase Open Client Client-Library (CT-Lib) behavior. See the description of the Release parameter for more information.

Requirements for use To use Sec_Data_Integrity or any other parameter supporting Open Client security services, you must meet certain requirements for using security services in your PowerBuilder application. For details, see "Requirements for using Open Client security services" in *Connecting to Your Database*.

Corresponding CT-Lib connection property Specifying a value for Sec_Data_Integrity sets the corresponding Sybase CT-Lib connection property named CS_SEC_INTEGRITY.

Examples

To specify that your connection's security mechanism checks data integrity:

- **Database profile** Select the Ensure Data Integrity check box on the Security page in the Database Profile Setup dialog box.
- **Application** Type the following in code:

```
SQLCA.DBParm="Sec_Data_Integrity=1;Release=15"
```

See also

Release
Sec_Confidential
Sec_Data_Origin
Sec_Replay_Detection
Sec_Seq_Detection

Sec_Data_Origin

Description

When you access a Sybase Adaptive Server Enterprise database in PowerBuilder through Open Client, Sec_Data_Origin is one of several parameters that support per-packet security for network-based security services. (For other per-packet security parameters, see the See Also section.)

Sec_Data_Origin controls whether your connection's security mechanism performs data origin stamping. When Sec_Data_Origin is set to 1, the security mechanism attaches a digital signature to each packet that verifies the packet's origin and contents.

When Sec_Data_Origin is set to 0 (the default), no data origin stamping is performed.

You must specify a value for Sec_Data_Origin *before* connecting to the database in PowerBuilder.

Using third-party security mechanisms
For information about the third-party security mechanisms and operating system platforms that Sybase has tested with Open Client security services, see the Open Client documentation.

Applies to

ASE, SYC Sybase Adaptive Server Enterprise

Syntax

Sec_Data_Origin=*value*

Parameter	Description
value	Specifies whether your connection’s security mechanism performs data origin stamping. Values are: • **0** (Default) Do *not* perform data origin stamping. You can also specify `'No'` or `'False'` to set this value. • **1** Perform data origin stamping by attaching a digital signature to each packet that verifies the packet's origin and contents. You can also specify `'Yes'` or `'True'` to set this value.

Default value

Sec_Data_Origin=0

Usage

Not supported with CyberSafe Kerberos Sec_Data_Origin is *not supported* if your security mechanism is CyberSafe Kerberos.

Set Release parameter For this parameter to take effect, you *must* also set the Release parameter to 11or higher to specify that your application should use the appropriate version of Sybase Open Client Client-Library (CT-Lib) behavior. See the description of the Release parameter for more information.

Requirements for use To use Sec_Data_Origin or any other parameter supporting Open Client security services, you must meet certain requirements for using security services in your PowerBuilder application. For details, see “Requirements for using Open Client security services” in *Connecting to Your Database*.

Corresponding CT-Lib connection property Specifying a value for Sec_Data_Origin sets the corresponding Sybase CT-Lib connection property named CS_SEC_DATAORIGIN.

Examples

To specify that your connection's security mechanism performs data origin stamping:

- **Database profile** Select the Verify Packet Origin check box on the Security page in the Database Profile Setup dialog box.
- **Application** Type the following in code:

```
SQLCA.DBParm="Sec_Data_Origin=1;Release=15"
```

See also

Release
Sec_Confidential
Sec_Data_Integrity
Sec_Replay_Detection
Sec_Seq_Detection

Sec_Delegation

Description

When you access a Sybase Adaptive Server Enterprise database in PowerBuilder through Open Client, Sec_Delegation is one of several parameters that support login authentication for network-based security services. (For other login authentication parameters, see the See Also section.)

For applications that are using network-based login authentication to connect to a Sybase Open Server gateway, Sec_Delegation controls whether the gateway server is allowed to connect to a remote SQL Server using delegated credentials. When Sec_Delegation is set to 1, the gateway can connect to a remote server using the client’s delegated credentials. The remote server must also be using network-based authentication and an identical security mechanism.

When Sec_Delegation is set to 0 (the default), the gateway server cannot connect to a remote server using delegated credentials.

You must specify a value for Sec_Delegation *before* connecting to the database in PowerBuilder.

Using third-party security mechanisms
For information about the third-party security mechanisms and operating system platforms that Sybase has tested with Open Client security services, see the Open Client documentation.

Applies to

ASE, SYC Sybase Adaptive Server Enterprise

Syntax

Sec_Delegation=*value*

Parameter	Description
value	Specifies whether the Sybase Open Server gateway is allowed to connect to a remote SQL Server using the client's delegated credentials. Values are: • **0** (Default) Prohibit the gateway from connecting to a remote server using delegated credentials. You can also specify `'No'` or `'False'` to set this value. • **1** Allow the gateway to connect to a remote server using delegated credentials. You can also specify `'Yes'` or `'True'` to set this value.

Default value

Sec_Delegation=0

Usage

Not supported with CyberSafe Kerberos Sec_Delegation is *not supported* if your security mechanism is CyberSafe Kerberos.

Set Release parameter For this parameter to take effect, you *must* also set the Release parameter to 11or higher to specify that your application should use the appropriate version of Sybase Open Client Client-Library (CT-Lib) behavior. See the description of the Release parameter for more information.

Requirements for use To use Sec_Delegation or any other parameter supporting Open Client security services, you must meet certain requirements for using security services in your PowerBuilder application. For details, see "Requirements for using Open Client security services" in *Connecting to Your Database*.

Corresponding CT-Lib connection property Specifying a value for Sec_Delegation sets the corresponding Sybase CT-Lib connection property named CS_SEC_DELEGATION.

Examples

To allow the Open Server gateway to connect to a remote server using delegated credentials:

- **Database profile** Select the Use Delegated Credentials check box on the Security page in the Database Profile Setup dialog box.
- **Application** Type the following in code:

```
SQLCA.DBParm="Sec_Delegation=1;Release=15"
```

See also

Release
Sec_Channel_Bind
Sec_Cred_Timeout
Sec_Keytab_File
Sec_Mechanism
Sec_Mutual_Auth
Sec_Network_Auth
Sec_Server_Principal
Sec_Sess_Timeout

Sec_Keytab_File

Description

When you access a Sybase Adaptive Server Enterprise database in PowerBuilder through Open Client, Sec_Keytab_File is one of several parameters that support login authentication for network-based security services. (For other login authentication parameters, see the See Also section.)

Sec_Keytab_File applies only to connections using Distributed Computing Environment (DCE) Kerberos as their security mechanism and requesting network-based login authentication. For those connections, Sec_Keytab_File specifies the name of the keytab file containing the security key for the DCE user.

You *must* set Sec_Keytab_File if the login ID specified in the database profile or Application is *different* from the user name of the DCE user currently running the application.

You must specify a value for Sec_Keytab_File *before* connecting to the database in PowerBuilder.

Using third-party security mechanisms
For information about the third-party security mechanisms and operating system platforms that Sybase has tested with Open Client security services, see the Open Client documentation.

Applies to

ASE, SYC Sybase Adaptive Server Enterprise

Syntax

Sec_Keytab_File='*keytab_filename*'

Parameter	Description
keytab_filename	The name of the keytab file containing the security key for the DCE user

Default value

None

PowerBuilder does not set Sec_Keytab_File or the corresponding Sybase Open Client Client-Library (CT-Lib) connection parameter CS_SEC_KEYTAB if you do not specify a value.

Usage

Supported only with Distributed Computing Environment Only Distributed Computing Environment (DCE) security servers and clients support the use of keytab files. Therefore, Sec_Keytab_File is supported only when your security mechanism is DCE Kerberos.

When to use If you want your application to be able to connect to a server with a different user name (login ID) than the DCE user currently running the application, set Sec_Keytab_File to specify the name of the keytab file containing the security key for the appropriate user. For details, see “Requirements for using Open Client security services” in *Connecting to Your Database*.

Set Release parameter For this parameter to take effect, you *must* also set the Release parameter to 11or higher to specify that your application should use the appropriate version of Sybase Open Client Client-Library (CT-Lib) behavior. See the description of the Release parameter for more information.

Requirements for use To use Sec_Keytab_File or any other parameter supporting Open Client security services, you must meet certain requirements for using security services in your PowerBuilder application. For details, see “Requirements for using Open Client security services” in *Connecting to Your Database*.

Corresponding CT-Lib connection property Specifying a value for Sec_Keytab_File sets the corresponding Sybase CT-Lib connection property named CS_SEC_KEYTAB.

Examples

To specify *C:\DCE_KEY* as the name of the DCE keytab file:

- **Database profile** Type the following in the Keytab File box on the Security page in the Database Profile Setup dialog box:

  ```
  C:\DCE_KEY
  ```

- **Application** Type the following in code:

  ```
  SQLCA.DBParm="Sec_Keytab_File='C:\DCE_KEY';Release=
  15"
  ```

See also

Release
Sec_Channel_Bind
Sec_Cred_Timeout
Sec_Delegation

Sec_Mechanism
Sec_Mutual_Auth
Sec_Network_Auth
Sec_Server_Principal
Sec_Sess_Timeout

Sec_Mechanism

Description

When you access a Sybase Adaptive Server Enterprise database in PowerBuilder through Open Client, Sec_Mechanism is one of several parameters that support login authentication for network-based security services. (For other login authentication parameters, see the See Also section.)

When you use Open Client security services, you must specify the name of the security mechanism you want to use in the Open Client/Open Server Configuration utility so that the required drivers can be loaded. The default security mechanism is the one currently specified as active in the Configuration utility.

Sec_Mechanism lets you specify a security mechanism name listed in the Open Client/Open Server Configuration utility *other than* the default (active) mechanism.

You must specify a value for Sec_Mechanism *before* connecting to the database in PowerBuilder.

Using third-party security mechanisms
For information about the third-party security mechanisms and operating system platforms that Sybase has tested with Open Client security services, see the Open Client documentation.

Applies to

ASE, SYC Sybase Adaptive Server Enterprise

Syntax

Sec_Mechanism='*mechanism_name*'

Parameter	Description
mechanism_name	The security mechanism name you want to use to establish a connection. The security mechanism name is case sensitive. You must specify it *exactly as it appears* in the Open Client/Open Server Configuration utility.

Default value

The default value for Sec_Mechanism is the security mechanism name currently specified as active in the Open Client/Open Server Configuration utility. If there is no security mechanism specified, no security service is used to establish the connection.

Usage

When to use Set Sec_Mechanism to use a security mechanism specified in the Open Client/Open Server Configuration utility *other than* the default (active) security mechanism. For instructions on using the Open Client/Open Server Configuration utility, see your Sybase Open Client/Server configuration guide.

Set Release parameter For this parameter to take effect, you *must* also set the Release parameter to 11or higher to specify that your application should use the appropriate version of Sybase Open Client Client-Library (CT-Lib) behavior. See the description of the Release parameter for more information.

Requirements for use To use Sec_Mechanism or any other parameter supporting Open Client security services, you must meet certain requirements for using security services in your PowerBuilder application. For details, see “Requirements for using Open Client security services” in *Connecting to Your Database*.

Corresponding CT-Lib connection property Specifying a value for Sec_Mechanism sets the corresponding Sybase CT-Lib connection property named CS_SEC_MECHANISM.

Examples

To specify KERBEROS as your security mechanism name:

- **Database profile** Type the following in the Security Mechanism box on the Security page in the Database Profile Setup dialog box:

```
KERBEROS
```

- **Application** Type the following in code:

```
SQLCA.DBParm="Sec_Mechanism='KERBEROS';Release=15"
```

See also

Release
Sec_Channel_Bind
Sec_Cred_Timeout
Sec_Delegation
Sec_Keytab_File
Sec_Mutual_Auth
Sec_Network_Auth
Sec_Server_Principal
Sec_Sess_Timeout

Sec_Mutual_Auth

Description

When you access a Sybase Adaptive Server Enterprise database in PowerBuilder through Open Client, Sec_Mutual_Auth is one of several parameters that support login authentication for network-based security services. (For other login authentication parameters, see the See Also section.)

Sec_Mutual_Auth controls whether your connection's security mechanism performs mutual authentication. When Sec_Mutual_Auth is set to 1, the server must prove its identity to the client before connecting by sending a credential token containing the server's principal name and proof that this name is authentic.

When Sec_Mutual_Auth is set to 0 (the default), no mutual authentication is performed.

You must specify a value for Sec_Mutual_Auth *before* connecting to the database in PowerBuilder.

Using third-party security mechanisms
For information about the third-party security mechanisms and operating system platforms that Sybase has tested with Open Client security services, see the Open Client documentation.

Applies to

ASE, SYC Sybase Adaptive Server Enterprise

Syntax

Sec_Mutual_Auth=*value*

Parameter	Description
value	Specifies whether your connection's security mechanism performs mutual authentication. Values are: • **0** (Default) Does *not* perform mutual authentication. You can also specify `'No'` or `'False'` to set this value. • **1** Performs mutual authentication. The server must prove its identity to the client before connecting by sending a credential token containing the server's principal name and proof that this name is authentic. You can also specify `'Yes'` or `'True'` to set this value.

Default value

Sec_Mutual_Auth=0

Usage

Set Release parameter For this parameter to take effect, you *must* also set the Release parameter to 11or higher to specify that your application should use the appropriate version of Sybase Open Client Client-Library (CT-Lib) behavior. See the description of the Release parameter for more information.

Requirements for use To use Sec_Mutual_Auth or any other parameter supporting Open Client security services, you must meet certain requirements for using security services in your PowerBuilder application. For details, see “Requirements for using Open Client security services” in *Connecting to Your Database*.

Corresponding CT-Lib connection property Specifying a value for Sec_Mutual_Auth sets the corresponding Sybase CT-Lib connection property named CS_SEC_MUTUALAUTH.

Examples

To specify that your connection’s security mechanism performs mutual authentication:

- **Database profile** Select the Mutual Authentication check box on the Security page in the Database Profile dialog box.
- **Application** Type the following in code:

```
SQLCA.DBParm="Sec_Mutual_Auth=1;Release=15"
```

See also

Release
Sec_Channel_Bind
Sec_Cred_Timeout
Sec_Delegation
Sec_Keytab_File
Sec_Mechanism
Sec_Network_Auth
Sec_Server_Principal
Sec_Sess_Timeout

Sec_Network_Auth

Description

When you access a Sybase Adaptive Server Enterprise database in PowerBuilder through Open Client, Sec_Network_Auth is one of several parameters that support login authentication for network-based security services. (For other login authentication parameters, see the See Also section.)

Sec_Network_Auth controls whether your connection uses network-based login authentication. When Sec_Network_Auth is set to 1, your connection uses network-based authentication when connecting to a secure SQL Server. **Network-based authentication** means that the security mechanism—not the application—confirms that the specified user name represents the authenticated user running the application.

Since the security mechanism rather than the application authenticates your user name (login ID), you need *not* supply a login password for authentication purposes in the database profile or Application if Sec_Network_Auth is set to 1.

When Sec_Network_Auth is set to 0 (the default), your connection does not use network-based login authentication to connect to the server. You must specify a value for Sec_Network_Auth *before* connecting to the database in PowerBuilder.

Using third-party security mechanisms
For information about the third-party security mechanisms and operating system platforms that Sybase has tested with Open Client security services, see the Open Client documentation.

Applies to

ASE, SYC Sybase Adaptive Server Enterprise

Syntax

Sec_Network_Auth=*value*

Parameter	Description
value	Specifies whether your connection uses network-based login authentication when connecting to a secure SQL Server. Values are: • **0** (Default) Does *not* use network-based login authentication when connecting to the server. You can also specify `'No'` or `'False'` to set this value. • **1** Uses network-based login authentication when connecting to the server. Since the security mechanism rather than the application authenticates your user name (login ID), you need *not* supply a login password for authentication purposes in the database profile or Application. You can also specify `'Yes'` or `'True'` to set this value.

Default value

Sec_Network_Auth=0

Usage

When to use Setting Sec_Network_Auth to 1 to enable network-based login authentication provides three important benefits for PowerBuilder users, because you do not have to specify a login password in the database profile or Application to authenticate the login ID when Sec_Network_Auth is set to 1:

- **Password not stored in registry file** Since you do not specify a login password, no login password is stored in the Windows registry.
- **Password not transmitted across network** Since you do not specify a login password, no login password is transmitted across the network to Adaptive Server.

- **Same user ID and password for different servers** You can use the same network user ID and password to connect to many different Adaptive Server database servers. You can change your password for the network security mechanism and have this change apply on all servers to which your application connects.

Set Release parameter For this parameter to take effect, you *must* also set the Release parameter to 11or higher to specify that your application should use the appropriate version of Sybase Open Client Client-Library (CT-Lib) behavior. See the description of the Release parameter for more information.

Requirements for use To use Sec_Network_Auth or any other parameter supporting Open Client security services, you must meet certain requirements for using security services in your PowerBuilder application. For details, see “Requirements for using Open Client security services” in *Connecting to Your Database*.

Corresponding CT-Lib connection property Specifying a value for Sec_Network_Auth sets the corresponding Sybase CT-Lib connection property named CS_SEC_NETWORKAUTH.

Examples

To specify that your connection uses network-based login authentication when connecting to the server:

- **Database profile** Select the Network Based Authentication check box on the Security page in the Database Profile Setup dialog box.
- **Application** Type the following in code:

```
SQLCA.DBParm="Sec_Network_Auth=1;Release=15"
```

See also

Release
Sec_Channel_Bind
Sec_Cred_Timeout
Sec_Delegation
Sec_Keytab_File
Sec_Mechanism
Sec_Mutual_Auth
Sec_Server_Principal
Sec_Sess_Timeout

Sec_Replay_Detection

Description

When you access a Sybase Adaptive Server Enterprise database in PowerBuilder through Open Client, Sec_Replay_Detection is one of several parameters that support per-packet security for network-based security services. (For other per-packet security parameters, see the See Also section.)

Sec_Replay_Detection controls whether your connection's security mechanism can detect and reject unauthorized attempts to capture and replay transmitted data. When Sec_Replay_Detection is set to 1, the security mechanism detects and subsequently rejects any unauthorized attempts by third parties to capture packets sent to the server and repeat (replay) the commands in the packets at a later time.

When Sec_Replay_Detection is set to 0 (the default), the security mechanism cannot detect unauthorized attempts to capture and replay data.

You must specify a value for Sec_Replay_Detection *before* connecting to the database in PowerBuilder.

Using third-party security mechanisms
For information about the third-party security mechanisms and operating system platforms that Sybase has tested with Open Client security services, see the Open Client documentation.

Applies to

ASE, SYC Sybase Adaptive Server Enterprise

Syntax

Sec_Replay_Detection=*value*

Parameter	Description
value	Specifies whether your connection's security mechanism can detect and reject unauthorized attempts to capture and replay transmitted data. Values are: • **0** (Default) Prohibits your security mechanism from detecting unauthorized attempts to capture and replay transmitted data. You can also specify `'No'` or `'False'` to set this value. • **1** Allows your security mechanism to detect and reject unauthorized attempts to capture and replay transmitted data. You can also specify `'Yes'` or `'True'` to set this value.

Default value

Sec_Replay_Detection=0

Usage

When to use In a nonsecure network, unauthorized third parties might capture the commands sent to a server in order to repeat (replay) these commands at a later date. For example, if packets are sent from the client to the server in the order P1, P2, P3 and the server receives the packets in the order P1, P3, P2, this is considered an attempt to replay the data. Setting Sec_Replay_Detection to 1 ensures that your security mechanism can detect and subsequently reject all such unauthorized attempts to capture and replay data transmitted over the network.

Set Release parameter For this parameter to take effect, you *must* also set the Release parameter to 11or higher to specify that your application should use the appropriate version of Sybase Open Client Client-Library (CT-Lib) behavior. See the description of the Release parameter for more information.

Requirements for use To use Sec_Replay_Detection or any other parameter supporting Open Client security services, you must meet certain requirements for using security services in your PowerBuilder application. For details, see “Requirements for using Open Client security services” in *Connecting to Your Database*.

Corresponding CT-Lib connection property Specifying a value for Sec_Replay_Detection sets the corresponding Sybase CT-Lib connection property named CS_SEC_DETECTREPLAY.

Examples

To allow your security mechanism to detect and reject unauthorized attempts to capture and replay transmitted data:

- **Database profile** Select the Detect Replayed Commands check box on the Security page in the Database Profile Setup dialog box.
- **Application** Type the following in code:

```
SQLCA.DBParm="Sec_Replay_Detection=1;Release=12.5"
```

See also

Release
Sec_Confidential
Sec_Data_Integrity
Sec_Data_Origin
Sec_Seq_Detection

Sec_Seq_Detection

Description

When you access a Sybase Adaptive Server Enterprise database in PowerBuilder through Open Client, Sec_Seq_Detection is one of several parameters that support per-packet security for network-based security services. (For other per-packet security parameters, see the See Also section.)

Sec_Seq_Detection controls whether your connection's security mechanism can detect and reject transmitted packets that arrive at the server in a different order than was originally sent from the client. When Sec_Seq_Detection is set to 1, the security mechanism detects and rejects packets that arrive at the server out of sequence.

When Sec_Seq_Detection is set to 0 (the default), the security mechanism cannot detect packets that arrive at the server out of sequence.

You must specify a value for Sec_Seq_Detection *before* connecting to the database in PowerBuilder.

Using third-party security mechanisms
For information about the third-party security mechanisms and operating system platforms that Sybase has tested with Open Client security services, see the Open Client documentation.

Applies to

ASE, SYC Sybase Adaptive Server Enterprise

Syntax

Sec_Seq_Detection=*value*

Parameter	Description
value	Specifies whether your connection's security mechanism can detect and reject packets that arrive at the server in a different order than the one in which they were sent from the client. Values are: • **0** (Default) Prohibit your security mechanism from detecting packets that arrive at the server out of sequence. You can also specify `'No'` or `'False'` to set this value. • **1** Allow your security mechanism to detect and reject packets that arrive at the server out of sequence. You can also specify `'Yes'` or `'True'` to set this value.

Default value

Sec_Seq_Detection=0

Usage

When to use When transmitting data over a network, commands sent to a server might arrive out of sequence. For example, if packets are sent from the client to the server in the order P1, P2, P3 and the server receives the packets in the order P1, P3, P2, this is considered an out-of-sequence error.

Setting Sec_Seq_Detection to 1 ensures that your security mechanism can detect and subsequently reject packets that arrive at the server out of sequence.

Set Release parameter For this parameter to take effect, you *must* also set the Release parameter to 11or higher to specify that your application should use the appropriate version of Sybase Open Client Client-Library (CT-Lib) behavior. See the description of the Release parameter for more information.

Requirements for use To use Sec_Seq_Detection or any other parameter supporting Open Client security services, you must meet certain requirements for using security services in your PowerBuilder application. For details, see "Requirements for using Open Client security services" in *Connecting to Your Database*.

Corresponding CT-Lib connection property Specifying a value for Sec_Seq_Detection sets the corresponding Sybase CT-Lib connection property named CS_SEC_DETECTSEQ.

Examples

To allow your security mechanism to detect and reject packets that arrive at the server out of sequence:

- **Database profile** Select the Detect Sequence Errors check box on the Security page in the Database Profile Setup dialog box.
- **Application** Type the following in code:

```
SQLCA.DBParm="Sec_Seq_Detection=1;Release=15"
```

See also

Release
Sec_Confidential
Sec_Data_Integrity
Sec_Data_Origin
Sec_Replay_Detection

Sec_Server_Principal

Description

When you access a Sybase Adaptive Server Enterprise database in PowerBuilder through Open Client, Sec_Server_Principal is one of several parameters that support login authentication for network-based security services. (For other login authentication parameters, see the See Also section.)

Sec_Server_Principal specifies the principal name of the server that you want to access. The **server principal name** is the name by which your security mechanism identifies each server.

If the server name (specified in the database profile or Application) is *different* from the server principal name for the server you want to access, you *must* set Sec_Server_Principal to the correct server principal name in order to connect.

You must specify a value for Sec_Server_Principal *before* connecting to the database in PowerBuilder.

Using third-party security mechanisms
For information about the third-party security mechanisms and operating system platforms that Sybase has tested with Open Client security services, see the Open Client documentation.

Applies to

ASE, SYC Sybase Adaptive Server Enterprise

Syntax

Sec_Server_Principal='*server_principal_name*'

Parameter	Description
server_principal_name	Specifies the principal name of the server you want to access

Default value

None

If you do not specify a value, the security mechanism uses the server's directory entry name, which is the same as the server name specified in the database profile or Application.

Usage

When to use When you use Open Client security services with PowerBuilder, the server's directory entry name (which you specify as the server name in the database profile or Application) might differ from the server principal name. In this case, you *must* set Sec_Server_Principal to the correct server principal name so that the security mechanism can identify the server you want to access.

Set Release parameter For this parameter to take effect, you *must* also set the Release parameter to 11or higher to specify that your application should use the appropriate version of Sybase Open Client Client-Library (CT-Lib) behavior. See the description of the Release parameter for more information.

Requirements for use To use Sec_Server_Principal or any other parameter supporting Open Client security services, you must meet certain requirements for using security services in your PowerBuilder application. For details, see "Requirements for using Open Client security services" in *Connecting to Your Database*.

Corresponding CT-Lib connection property Specifying a value for Sec_Server_Principal sets the corresponding Sybase CT-Lib connection property named CS_SEC_SERVERPRINCIPAL.

Examples

To specify SYS12NT as the principal name of the server you want to access:

- **Database profile** Type `SYS12NT` in the Server Principal Name box on the Security page in the Database Profile Setup dialog box.
- **Application** Type the following in code:

```
SQLCA.DBParm="Sec_Server_Principal='SYS12NT';Releas
e=12"
```

See also

Release
Sec_Channel_Bind
Sec_Cred_Timeout
Sec_Delegation
Sec_Keytab_File
Sec_Mechanism
Sec_Mutual_Auth
Sec_Network_Auth
Sec_Sess_Timeout

Sec_Sess_Timeout

Description

When you access a Sybase Adaptive Server Enterprise database in PowerBuilder through Open Client, Sec_Sess_Timeout is one of several parameters that support login authentication for network-based security services. (For other login authentication parameters, see the See Also section.)

Some security mechanisms allow applications to set session timeout values for connections using network-based login authentication. For these connections, Sec_Sess_Timeout specifies the number of seconds remaining before a session expires. The session timeout period begins when the connection is opened.

By default, Sec_Sess_Timeout specifies that there is no session timeout limit; the session does not expire. You must specify a value for Sec_Sess_Timeout *before* connecting to the database in PowerBuilder.

Using third-party security mechanisms
For information about the third-party security mechanisms and operating system platforms that Sybase has tested with Open Client security services, see the Open Client documentation.

Applies to

ASE, SYC Sybase Adaptive Server Enterprise

Syntax

Sec_Sess_Timeout=*value*

Parameter	Description
value	Specifies the number of seconds remaining before a session expires. You can also specify `'no_limit'` (the default) to indicate that the session does not expire. A session timeout value set by the security system's administrator supersedes any value you specify for Sec_Sess_Timeout.

Default value

Sec_Sess_Timeout='no_limit'

Usage

CyberSafe Kerberos If your security mechanism is CyberSafe Kerberos, Sec_Sess_Timeout cannot override the installation default value set for session timeout.

Set Release parameter For this parameter to take effect, you *must* also set the Release parameter to 11or higher to specify that your application should use the appropriate version of Sybase Open Client Client-Library (CT-Lib) behavior. See the description of the Release parameter for more information.

Requirements for use To use Sec_Sess_Timeout or any other parameter supporting Open Client security services, you must meet certain requirements for using security services in your PowerBuilder application. For details, see "Requirements for using Open Client security services" in *Connecting to Your Database*.

Corresponding CT-Lib connection property Specifying a value for Sec_Sess_Timeout sets the corresponding Sybase CT-Lib connection property named CS_SEC_SESSTIMEOUT.

Examples

To specify 14,400 seconds (4 hours) remaining before a session expires:

- **Database profile** Type `14400` in the Session Timeout box on the Security page in the Database Profile Setup dialog box.
- **Application** Type the following in code:

```
SQLCA.DBParm="Sec_Sess_Timeout=14400"
```

See also

Release
Sec_Channel_Bind
Sec_Cred_Timeout
Sec_Delegation
Sec_Keytab_File
Sec_Mechanism
Sec_Mutual_Auth
Sec_Network_Auth
Sec_Server_Principal

ServerName

Description
Specifies the server name for Oracle client interface (OCI) pooling.

Applies to
ORA Oracle 11*g*

Syntax
ServerName =*value*

Parameter	Description
value	Sets the name of the server to use for OCI pooling

Default value
None

Usage
Oracle client interface pooling for PowerBuilder applications is created when you connect to an Oracle server for the first time. The pooling is identified by the server name and character set that are specified in the ServerName and NLS_Charset parameters, respectively. If two Oracle connections are connected to the same Oracle server but use different character sets, the connections must reside in different connection or session pools. All pooling-related DBParm parameters must be set before the initial database connection.

Examples
To specify the server name for OCI pooling:

- **Database profile** Type the server name you want to set in the Server text box on the Connection page in the Database Profile Setup dialog box.
- **Application** Type the following in code:

```
my_trans.Server = "myServerName"
```

ServiceComponents

Description
Specifies the global services the OLE DB interface can use.

When to specify ServiceComponents
You must specify the ServiceComponents parameter *before* connecting to the database.

Applies to
OLE DB

Syntax
ServiceComponents='*servicecomponent_name*'

Default value
None

Examples
To enable the resource pooling service component:

- **Database profile** Select Resource Pooling from the Service Component Support box on the System page in the Database Profile Setup dialog box.
- **Application** Type the following in code:

```
SQLCA.DBParm="ServiceComponents='DBPROPVAL_OS_RESOU
RCEPOOLING'"
```

SessionHomogeneous

Description

Authenicates all sessions in a session pool with the user name and password in effect when the session pool was created.

Applies to

ORA Oracle 11*g*

Syntax

SessionHomogeneous =*value*

Parameter	Description
value	Specifies restrictions on authentication values for a session pool. Values are: • **True** Sessions in the session pool can be authenticated only with the user name and password of the session pool creator. • **False** (Default) The sessions in the session pool can use the user name and password of a connection request even if they differ from the user name and password of the session pool creator.

Default value

False, but this value is meaningless if session pooling is not set.

Usage

When set to True, all sessions in the session pool are authenticated with the user name and password in effect when the pool was created. The user name and password in later connection requests are ignored. Proxy sessions cannot be created in homogeneous session mode.

Examples

The following example enables homogeneous session pooling:

- **Database profile** Select Session Pooling from the Pooling Type drop-down list on the Pooling page in the Database Profile Setup dialog box, select the Homogeneous Session Pooling check box on the same page.
- **Application** Type the following in code:

```
my_trans.dbparm="Pooling='session',
  SessionHomogeneous='Yes'"
```

See also

Pooling

ShowTempTables

Description	Specifies whether temporary tables are displayed when you request a list of tables from the Database painter or SQL Select painter.
Applies to	ASE, SYC Sybase Adaptive Server Enterprise
Syntax	ShowTempTables=*value*

Parameter	**Description**
value	Specifies whether temporary tables are displayed when you request a list of tables. Values are: • **1** (Default) Display temporary tables. • **0** Do not display temporary tables.

Default value	ShowTempTables=1
Usage	The ShowTempTables database parameter applies only to database connections from within the development environment. When ShowTempTables=1, a request for a list of database tables from within a painter causes both sp_pb100table and sp_pb100temptab to execute. The union of these two result sets is displayed to the user. If ShowTempTables=0, only sp_pb100table is executed. The behavior is the same if the stored procedures are not installed on the ASE server.
Examples	To specify that you want to display temporary tables, select the Show Temp Tables check box on the System page in the Database Profile Setup dialog box.

ShowWarnings

Description	Specifies whether warning message text can be concatenated to existing error messages.
Applies to	DIR Sybase DirectConnect
Syntax	ShowWarnings=*value*

Parameter	**Description**
value	Specifies whether warning message text can be concatenated to existing error messages. Values are: • **0** (Default) Does not allow the concatenation of warning message text to existing error messages. • **1** Allows the concatenation of warning message text to existing error messages.

Default value	ShowWarnings=0

Usage

The ShowWarnings parameter allows the DirectConnect interface to use warning and error processing similar to that formerly available in the InformationConnect DB2 Gateway (MDI) interface. For example, if a single warning message appears on the DIR error queue, the default behavior is to discard warnings. If ShowMessages is set, `sqlca.sqlcode=-1` and `sqlca.sqlerrtext="`***`text_of_warning_message`***`"` are returned to the application.

`sqlca.sqlerrtext` cannot exceed 254 characters. Consequently, if multiple warning messages are returned together, message text might be truncated.

For Access Service connections, must be issued before connecting to ensure its correct operation.

Examples

To specify that you want to show warning messages as errors:

- **Database profile** Select the Show Warning Messages as Errors check box on the Syntax page in the Database Profile Setup dialog box.
- **Application** Type the following in code:

```
SQLCA.DBParm="ShowWarnings=1"
```

SPCache

Description

Specifies the number of stored procedures for which the driver caches information.

When to specify SPCache
If you want to change the default value for SPCache, you must specify a new value *before* connecting to the database. The value cannot be changed at runtime.

Applies to

ADO.NET
OLE DB
O90 Oracle9*i*
O10 Oracle 10*g*
ORA Oracle 11*g*
SNC SQL Native Client for Microsoft SQL Server

Syntax

SPCache=*value*

Parameter	Description
value	A number that specifies how many stored procedures are added to a cache that contains information about each stored procedure's parameters. To turn off caching, specify 0.

Default value

SPCache=50 (ADO.NET), SPCache=100 (others)

Usage

By default, the driver retrieves information from the server about a stored procedure's parameters the first time the stored procedure is called and caches that information. The next time the procedure is called, the driver retrieves the information from the cache to improve performance. The information is retrieved based on the stored procedure's name. The name is case sensitive.

If you call two different stored procedures with the same name, you can turn off caching by setting the value of SPCache to 0.

Examples

To turn off caching of stored procedure parameter information:

- **Database profile** Specify 0 in the Maximum Procedures to Cache box on the System page in the Database Profile Setup dialog box.
- **Application** Type the following in code:

```
SQLCA.DBParm="SPCache=0"
```

See also

BindSPInput

SQLCache

Description

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

PowerBuilder caches:

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

Applies to

ODBC

Syntax

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, PowerBuilder removes the statement that was least recently executed.

SQLCache and bind variables Caching SQL statements that you execute frequently improves their performance. Statements with bind variables are often 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 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, PowerBuilder 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, PowerBuilder stores the parsed statement, result set description, and memory buffer allocation and binding in the SQL cache. The next time you execute this statement, PowerBuilder finds it in the cache and avoids the overhead of repeating these steps.

If PowerBuilder 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 PowerBuilder found a matching statement in the SQL cache
- **Misses** The number of times PowerBuilder 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 page in the Database Profile Setup dialog box.
- **Application** Type the following in code:

```
SQLCA.DBParm="SQLCache=25"
```

See also

DisableBind

SQLQualifiers

Description

Sets the level of qualification for identifiers (table and column names) in SQL statements when you connect to a database. This affects behavior in DataWindow objects.

When PowerBuilder **qualifies** a table or column name, it prefixes it with the name of the owner. For example, if a user named Fran owns a table named Sales, the qualified table name is Fran.Sales.

Applies to

DIR Sybase DirectConnect

Syntax

SQLQualifiers=*value*

Parameter	Description
value	Sets the level of qualification for identifiers in SQL statements when you connect to a database. Values are: • **0** (Default) Do not qualify identifiers with owner names in SQL statements. • **1** Qualify identifiers with owner names in SQL statements.

Default value

SQLQualifiers=0

Usage

When PowerBuilder qualifies identifiers If the name of the table owner is the same as the name of the user logged in to the database, PowerBuilder does not qualify identifiers with owner names in the SQL statements it generates. If you set the SQLQualifiers parameter to 1, PowerBuilder qualifies identifiers with an owner name in SQL statements.

Examples

To specify that you want PowerBuilder to qualify identifiers with owner names in SQL statements:

- **Database profile** Select the Qualify Identifiers With Owner Names check box on the Syntax page in the Database Profile Setup dialog box.

- **Application** Type the following in code:

```
SQLCA.DBParm="SQLQualifiers=1"
```

StatementCache

Description

Specifies whether statement caching is enabled and the maximum number of statements to cache.

Applies to

O10 Oracle 10*g*
ORA Oracle 11*g*

Syntax

StatementCache='*value*'

Parameter	Description
value	Specifies whether statement caching is enabled and the maximum number of statements to cache. Values are: • **0** Statement caching is disabled. • ***n*** Statement caching is enabled and *n* statements may be cached, where *n* is a positive integer.

Default value

StatementCache=0

Usage

Statement caching in Oracle provides and manages a cache of statements for each session. On the server, cursors are ready to be used without the need to parse the statement again before execution. Statement caching can be used with connection or session pooling to improve performance and scalability.

Examples

To enable statement caching and specify that five statements can be cached:

- **Database profile** Type 5 in the Number of Oracle Statements Cached box on the Transaction page in the Database Profile Setup dialog box.
- **Application** Type the following in code:

```
SQLCA.DBParm="StatementCache=5"
```

See also

CnnPool
Pooling

StaticBind

Description

When you retrieve data from a database into a DataWindow object or report, PowerBuilder does not get a result set description to validate the SELECT statement against the database server before retrieving the data. This means the retrieval should be faster, 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 PowerBuilder get a description of the result set before retrieving data, set the StaticBind parameter to 0 or No.

Applies to

ADO.NET
ASE, SYC and SYJ Sybase Adaptive Server Enterprise
DIR Sybase DirectConnect
JDB JDBC
ODBC
OLE DB
O90 Oracle9*i*
O10 Oracle 10*g*
ORA Oracle 11*g*
SNC SQL Native Client for Microsoft SQL Server

Syntax

StaticBind=*value*

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

Default value

StaticBind=1

Usage

Validation When StaticBind is set to 1 (the default), PowerBuilder does not validate the SELECT statement against the database server before retrieving data. It assumes that the result set matches the column format of the DataWindow object or report into which it is being retrieved. If a mismatch occurs, PowerBuilder displays an error.

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

- The database definition changes after you build the DataWindow object or report.
- You build the DataWindow object or report 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 or report 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 or report and DBMS result set descriptions do not match and you want to avoid errors, set StaticBind to 0 or No to specify that PowerBuilder should *always* get a result set description before retrieving data into a DataWindow object or report.

Examples

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

- **Database profile** Clear the Static Bind check box on the Transaction page (or System page in the case of the OLE DB interface) in the Database Profile Setup dialog box.
- **Application** To specify this statement in code, type the following:

```
SQLCA.DBParm="StaticBind=0"
```

StmtCache

Description

Specifies whether SQL statement caching is enabled on the client.

Applies to

I10 Informix

Syntax

StmtCache='*value*'

Parameter	Description
value	Specifies whether SQL statement caching is enabled on the client. Values are: • **0** SQL statement caching is disabled on the client. • **1** SQL statement caching is enabled on the client.

Default value

StmtCache=0

Usage

In IDS 9.2.1 and later, the database server uses the SQL statement cache (SSC) to store SQL statements across user sessions. When any user executes a statement already stored in the SQL statement cache, the database server does not parse and optimize the statement again, resulting in improved performance. The statement must be a SELECT, UPDATE, DELETE, or INSERT statement, and it cannot contain user-defined routines.

There are several ways to configure caching on the server. The SET STATEMENT CACHE statement takes precedence over the STMT_CACHE environment variable and the STMT_CACHE configuration parameter. You must enable the SQL statement cache, either by setting the STMT_CACHE configuration parameter or by using the Informix onmode utility, *before* the SET STATEMENT CACHE statement can execute successfully.

You can set the StmtCache DBParm on the System tab page in the Database Profile Setup dialog box for I10 connections to turn SQL statement caching on or off on the client. However, the server must be configured to support SQL statement caching before you can access the cache from the client.

For more information about Informix SQL statement caching, see the IBM Informix Dynamic Server Performance Guide at http://publib.boulder.ibm.com/epubs/pdf/25122960.pdf.

Examples

To specify that the client can access data from the SQL statement cache if it is configured on the server:

- **Database profile** Select the Enable SQL Statement Cache check box on the System page in the Database Profile Setup dialog box.
- **Application** Type the following in code:

```
SQLCA.DBParm="StmtCache=1"
```

StrByCharset

Description

Specifies how to convert string data between PowerBuilder Unicode strings and multibyte strings on the client.

Applies to

I10 Informix
O90 Oracle9*i*
O10 Oracle 10*g*
ORA Oracle 11*g*

Syntax

StrByCharset='*value*'

Parameter	Description
value	Specifies how to convert string data between PowerBuilder Unicode strings and multibyte strings on the client. Values are: • **0** String conversion is based on the current OS code page, or, for UTF-8 code sets on Informix, on the UTF-8 code set. • **1** For Informix, string conversion is based on the code set specified in the Client_Locale database parameter. For Oracle, string conversion is based on the Oracle character set specified in the NLS_Charset database parameter.

Default value

StrByCharset=0

Usage

Informix The StrByCharset DBParm specifies how to convert string data between PowerBuilder Unicode strings and Informix client multibyte strings. By default, string conversion for UTF-8 code sets is based on the UTF-8 code set, and string conversion for non-UTF-8 code sets is based on the current OS code page. If StrByCharset is set to 1 (true), string conversion is based on the code set specified in the DBParm Client_Locale.

Example 1
With these settings:

```
StrByCharset=0
Informix Server DB_LOCALE='EN_US.8859-1'
PowerBuilder Informix client CLIENT_LOCALE='EN_US.8859-
1'
OS code page=1252
```

The Informix client and server character sets match, so all string data from the client can be sent to the server and fetched back directly, even if some characters do not belong to the EN_US.8859-1 character set but are within code page 1252, because no string conversion happens between the Oracle client and server. All string data can be displayed using the OS code page. If StrByCharset is set to 1, when string data is converted between the Informix client (ANSI) and PowerBuilder (Unicode), characters that are not in the EN_US.8859-1 character set are not converted correctly.

Example 2
With these settings:

```
StrByCharset=0
Informix Server DB_LOCALE='DE_DE.ASCII'
PowerBuilder Informix client
CLIENT_LOCALE='DE_DE.ASCII'
OS code page=949 for Korean
```

The client and server character sets match. All string data, including Korean characters, are sent or received one byte at a time so no data is lost. Using the Korean OS code page, all Korean string data can be converted from ANSI to Unicode safely with StrByCharset set to 0.

Example 3
With these settings:

```
StrByCharset=1
Informix Server DB_LOCALE='ZH_TW.BIG5'
PowerBuilder Informix client CLIENT_LOCALE='ZH_TW.BIG5'
OS code page=1252
```

Because characters in the ZH_TW.BIG5 character set do not belong to code page 1252, string data conversion must be based on the ZH_TW.BIG5 character set.

Oracle The StrByCharset DBParm specifies how to convert string data between PowerBuilder Unicode and OCI client multibyte strings. By default, string conversion is based on the current OS code page or an Oracle character set. The StrByCharset database parameter is ignored if NLS_Charset is set to Unicode because both PowerBuilder and the OCI client use the UTF-16 format. When NLS_Charset is set to another value, you must set StrByCharset to 1 if the character set on the client is incompatible with the OS code page.

Example 1 With these settings:

```
StrByCharset=0
Oracle Server NLS_CHARACTER='WE8ISO8859P1'
PowerBuilder OCI client NLS_CHARSET='WE8ISO8859P1'
OS code page=1252
```

The Oracle client and server character sets match, so all string data from the client can be sent to the server and fetched back directly, even if some characters do not belong to the WE8ISO8859P1 character set but are within code page 1252, because no string conversion happens between the Oracle client and server. All string data can be displayed using the OS code page. If StrByCharset is set to 1, when string data is converted between the OCI client (ANSI) and PowerBuilder (Unicode), characters that are not in the WE8ISO8859P1character set are not converted correctly.

Example 2 With these settings:

```
StrByCharset=0
Oracle Server NLS_CHARACTER='US7ASCII'
PowerBuilder OCI client NLS_CHARSET='US7ASCII'
OS code page=949 for Korean
```

The client and server character sets match. All string data, including Korean characters, is sent or received one byte at a time so no data is lost. Using the Korean OS code page, all Korean string data can be converted from ANSI to Unicode safely with StrByCharset set to 0.

Example 3 With these settings:

```
StrByCharset=1
Oracle Server NLS_CHARACTER='ZHT16BIG5'
PowerBuilder OCI client NLS_CHARSET='ZHT16BIG5'
OS code page=1252
```

Because characters in the ZHT16BIG5 character set do not belong to code page 1252, string data conversion must be based on the ZHT16BIG5 character set.

Examples

To specify that the Informix client should use the fr_FR.8859-1 character set defined by the Client_Locale parameter to handle string datatypes:

- **Database profile** Specify fr_FR.8859-1 in the Client Locale box and select the String Conversion Based on Client Locale box on the Regional Settings page in the Database Profile Setup dialog box.
- **Application** Type the following in code:

```
SQLCA.DBParm="Client_Locale='fr_FR.8859-
1',StrByCharset=1"
```

To specify that the OCI client should use the character set defined by the NLS_LANG parameter on the local computer to handle string datatypes:

- **Database profile** Select Local from the NLS Charset drop-down list and select the Use String Conversion Based on Oracle Character Set box on the System page in the Database Profile Setup dialog box.
- **Application** Type the following in code:

```
SQLCA.DBParm="NLS_Charset='Local',StrByCharset=1"
```

See also

Client_Locale
Db_Locale
NLS_Charset

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, PowerBuilder retains parameter names if 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 page in the Database Profile Setup dialog box.
- **Application** Type the following in code:

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

SvrFailover

Description

Specifies whether you want PowerBuilder to recognize and participate in failover to a designated backup database server if the current database server goes down.

When to specify SvrFailover
You must specify the SvrFailover parameter *before* connecting to the database.

Applies to

ASE, SYC or SYJ Sybase Adaptive Server Enterprise (12.0 and higher database connections only)
O90 Oracle9*i*
O10 Oracle 10*g*

ORA Oracle 11*g*

Syntax

SvrFailover='*value*'

Parameter	Description
value	Specifies whether you want PowerBuilder to recognize and participate in failover to a designated backup database server if the current database server goes down. Values are: • **No** (Default) PowerBuilder should not recognize or participate in failover. • **Yes** PowerBuilder should recognize and participate in failover.

Default value

SvrFailover='No'

Usage

Both Oracle and Sybase support database server failover in Oracle Version 8.1.5 or later and Adaptive Server 12 or later. For information about how to configure failover for these database servers, see your Oracle or Adaptive Server documentation. To avoid losing your PowerBuilder database connection (as the result of a timeout) when a failover takes place, set the SvrFailover parameter so that PowerBuilder recognizes and participates in the database server failover.

Oracle also allows you some control over the failover process. Three additional parameters allow you to specify the number of times you want the database server to which you are connected to attempt a failover, how long to wait between failover attempts, and whether PowerBuilder should display a runtime dialog box indicating when a failover occurs.

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

Examples

To tell PowerBuilder to recognize and participate in failover:

- **Database profile** Select the Allow Server Failover check box on the Network page in the Database Profile Setup dialog box.
- **Application** Type the following in code:

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

See also

FoDelay
FoDialog
FoRetryCount

SystemOwner

Description

Specifies the owner of the IBM DB2 system tables that you want PowerBuilder to use. PowerBuilder accesses the DB2 system tables to get information about the tables and columns in your database.

Applies to

DIR Sybase DirectConnect

Syntax

SystemOwner='*owner_name*'

Parameter	Description
owner_name	Specifies the owner of the DB2 system tables that you want PowerBuilder to use (default=SYSIBM)

Default value

SystemOwner='SYSIBM'

Usage

When you use the SystemOwner parameter to specify a nondefault system owner, PowerBuilder uses the set of system tables belonging to this owner instead of the default system tables owned by SYSIBM.

If your site has a large DB2 system catalog, it might be useful to create local copies of the catalog tables and populate them with a subset of the information in the default system catalog. These local copies are sometimes called **shadow catalogs**.

You can then set the value of SystemOwner to the owner of the shadow catalogs. This tells PowerBuilder to access the smaller shadow catalogs instead of the larger default system tables, resulting in faster performance. However, you must make sure to keep the shadow catalogs synchronized with the default system catalog owned by SYSIBM.

For more about creating shadow catalogs, see your DB2 system administrator or check whether there is a technical document that describes how to do it. Updated information about connectivity issues is available from the Sybase Customer Service and Support Web site at http://www.sybase.com/support.

Examples

To specify MYAPP as the owner of the system tables that you want PowerBuilder to use:

- **Database profile** Type `MYAPP` in the CSP Catalog Qualifier box on the System page in the Database Profile Setup dialog box.
- **Application** Type the following in code:

```
SQLCA.DBParm="SystemOwner='MYAPP'"
```

SystemProcs

Description

Specifies whether you want PowerBuilder to display both system-stored procedures and user-defined stored procedures in the connected database when you request a list of stored procedures.

By default, PowerBuilder displays both system and user-defined stored procedures in the connected database. If you set SystemProcs to 0 or No, only user-defined stored procedures are displayed.

Applies to

ASE, SYC Sybase Adaptive Server Enterprise

Syntax

SystemProcs=*value*

Parameter	Description
value	Specifies whether you want PowerBuilder to display both system-stored procedures and user-defined stored procedures in the connected database when you request a list of stored procedures. Values are: • **0** Display only user-defined stored procedures. You can also specify `'No'` to set this value. • **1** (Default) Display both system-stored procedures and user-defined stored procedures. You can also specify `'Yes'` to set this value.

Default value

SystemProcs=1

Usage

Setting SystemProcs to 0 or No speeds response time if you want to work only with user-defined stored procedures.

Examples

To specify that you want PowerBuilder to display only user-defined stored procedures in the connected database when you request a list of stored procedures:

- **Database profile** Clear the Display System Stored Procedures check box on the System page in the Database Profile Setup dialog box.
- **Application** To specify this statement in code, type the following:

```
SQLCA.DBParm="SystemProcs=0"
```

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 PowerBuilder. Setting this parameter can be useful if you are working with a very large database in the PowerBuilder development environment.

When to specify TableCriteria
You must specify the TableCriteria parameter *before* connecting to the database.

The TableCriteria parameter has no effect in a PowerBuilder application script.

Applies to

ASE, SYC Sybase Adaptive Server Enterprise
DIR Sybase DirectConnect
JDB JDBC
ODBC
O90 Oracle9*i*
O10 Oracle 10*g*
ORA Oracle 11*g*

Syntax

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

Default value

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

Oracle If you do not specify a value for TableCriteria, all Oracle tables, views, and synonyms that you have permission to access display in the Installed Database Interfaces Tables list by default.

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. **Default for DirectConnect interface** If you omit this value when connected through the DirectConnect interface, PowerBuilder displays all tables that you have permission to access in the current database, as defined in the DirectConnect server configuration file. **Default for Adaptive Server Enterprise interface** If you omit this value when connected through the Adaptive Server Enterprise interface, PowerBuilder displays all tables in the current database.
Table Owner	Displays only those tables belonging to the specified table owner. You can use wildcard characters. If you omit this value, PowerBuilder 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.

Field	Description
Include System Tables	Specifies that system tables should be displayed.

Adaptive Server Enterprise and DirectConnect These Sybase database interfaces use stored procedures to create the table list:

- **DirectConnect interface** Uses the sp_tables stored procedure.
- **Adaptive Server Enterprise interface** Uses the version of the sp_pb125table stored procedure installed by you or your database administrator.

  For information about which version of sp_pb125table to install when connecting to an Adaptive Server Enterprise database, see "Installing PowerBuilder stored procedures in Adaptive Server Enterprise databases" in *Connecting to Your Database*.

PowerBuilder uses the TableCriteria parameter to supply the arguments to sp_tables or sp_pb125table and build the table list based on your search criteria.

Examples

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

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

ThreadSafe

Description

Specifies whether your connection should take advantage of the database server threadsafe client libraries.

By default, ThreadSafe is set to No to specify that your connection does not use the threadsafe client libraries. If you set ThreadSafe to Yes, your connection takes advantage of the threadsafe client libraries.

When to specify ThreadSafe
You must specify a value for ThreadSafe *before* connecting to the database.

Applies to

I10 Informix
IN9 Informix
O90 Oracle9*i*
O10 Oracle 10*g*
ORA Oracle 11*g*

Syntax

ThreadSafe='*value*'

Parameter	Description
value	Specifies whether a connection uses the database server threadsafe client libraries. Values are: • **Yes** Your connection uses the threadsafe client libraries. Use this setting when building distributed applications that require a multi-threaded environment. • **No** (Default) Your connection does not use the threadsafe client libraries. Use this setting when building nondistributed applications that require a single-threaded environment.

Default value

ThreadSafe='No'

Usage

When to use Oracle and Informix provide support for thread safety in their client libraries. When you are using the Oracle or Informix database interface to build multi-threaded applications in PowerBuilder, set the ThreadSafe parameter to Yes to use threadsafe client libraries. This prevents possible side effects among multiple threads of execution making calls to the database server. Your application might incur a performance penalty when you use the threadsafe client libraries.

By default, the client software (and, thus, PowerBuilder) assumes that you are building an application in a single-threaded environment that does not need the threadsafe client libraries. This default ensures that single-threaded applications do not incur the performance penalty associated with using threadsafe libraries. Therefore, if you are building single-threaded applications, you can leave the ThreadSafe parameter set to No (the default).

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

Examples

To specify that your connection uses the threadsafe client libraries:

- **Database profile** Select the Thread Safe check box on the Connection page in the Database Profile Setup dialog box.
- **Application** Type the following in code:

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

Time

Description

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

Applies to

JDB JDBC
ODBC
O90 Oracle9*i*
O10 Oracle 10*g*
ORA Oracle 11*g*

Syntax

The syntax you use to specify the Time parameter 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 time format.

In code, you must use the following syntax:

JDBC and ODBC syntax PowerBuilder parses the backslash followed by two single quotes (\'') as a single quote when it builds the SQL UPDATE statement:

Time=' \''*time_format*\'' '

Oracle syntax PowerBuilder parses each set of four consecutive single quotes ('''') as a single quote when it builds the SQL UPDATE statement:

Time=' ''''*time_format*'''' '

Parameter	Description
' \''	**JDBC and ODBC syntax** 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.
' ''''	**Oracle syntax** Type a single quote, followed by one space, followed by four single quotes. There is no space between the four single quotes and the beginning of the time format.
time_format	The time format you want PowerBuilder 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*.

Parameter	Description
\'' '	**JDBC and ODBC syntax** 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.
'''' '	**Oracle syntax** Type four single quotes, followed by one space, followed by a single quote. There is no space between the end of the time format and the four single quotes.

Default value

The default value for Time depends on the DBMS you are accessing:

DBMS	Date default value
JDBC	If no value is specified for the Time parameter, PowerBuilder looks for a time format in the section for your JDBC driver in the registry. If no time format is found in the registry, PowerBuilder uses the JDBC time format escape sequence.
ODBC	If no value is specified for the Time parameter, PowerBuilder looks for a time format in the section for your ODBC driver in the PBODB125 initialization file. If no time format is found in the PBODB125 initialization file, PowerBuilder uses the ODBC time format escape sequence.
Oracle	The default Oracle date format. For information, see your Oracle documentation.

Examples

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

```
hh:mm
```

Example 1 (JDBC and ODBC syntax) To specify that PowerBuilder 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 page in the Database Profile Setup dialog box:

  ```
  hh:mm
  ```

- **Application** Type the following in code:

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

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

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

Example 2 (Oracle syntax) To specify that PowerBuilder should use this format for the time datatype when it builds the SQL UPDATE statement:

- **Database profile** Type `hh:mm` in the Time Format box on the Syntax page in the Database Profile Setup dialog box.
- **Application** Type the following in code:

```
SQLCA.DBParm="Time=' ''''hh:mm'''' '"
```

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

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

See also

Date
DateTime

TimeFormat

Description

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

Applies to

ADO.NET
OLE DB
SNC SQL Native Client for Microsoft SQL Server

Syntax

TimeFormat ='*time_format*'

Parameter	Description
time_format	The time format you want PowerBuilder 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*.

Default value

If no value is specified for the TimeFormat parameter, PowerBuilder does not use a time datatype.

Usage

When you call stored procedures, the database server might not accept the time format built by PowerBuilder. If this occurs, you can try to use another format. For example, for Microsoft SQL Server, try this format:

```
TimeFormat='\''hh:mm:ss\'''
```

Examples

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

```
hh:mm
```

To specify that PowerBuilder 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 page in the Database Profile Setup dialog box:

  ```
  hh:mm
  ```

- **Application** Type the following in code:

  ```
  SQLCA.DBParm="TimeFormat='hh:mm'"
  ```

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

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

See also

DateFormat
DateTimeFormat

Timeout

Description

Specifies the number of seconds the interface should wait for a connection to the data provider to complete.

When to specify TimeOut
You must specify a value for TimeOut *before* connecting to the database.

Applies to

ADO.NET
OLE DB
SNC SQL Native Client for Microsoft SQL Server

Syntax

TimeOut=*value*

Parameter	Description
value	The number of seconds the interface waits for a connection to complete.

Default value

None

Usage

The default value for the TimeOut parameter is driver-specific.

Examples

To set the TimeOut value to wait 10 seconds for a connection to complete:

- **Database profile** Type `10` in the Timeout box on the System page in the Database Profile Setup dialog box.
- **Application** Type the following in code:

```
SQLCA.DBParm="TimeOut=10"
```

TimeStamp

Description

Specifies whether PowerBuilder should map DateTime and Time datatypes to the Oracle TimeStamp datatype.

Applies to

OLE DB
O90 Oracle9*i*
O10 Oracle 10*g*
ORA Oracle 11*g*
SQLNCLI10

Syntax

TimeStamp=*value*

Parameter	Description
value	Specifies whether PowerBuilder should map DateTime and Time datatypes to the Oracle TimeStamp datatype. Values are: • **0** Map DateTime and Time datatypes to the Oracle Date datatype. • **1** (Default) Map DateTime and Time datatypes to the Oracle TimeStamp datatype.

Default value

TimeStamp=1

Usage

Oracle9*i* and later databases and the O90 and O10 interfaces support the Oracle timestamp datatype. This datatype includes the date and the time including milliseconds. The existing Oracle Date datatype does not include millisecond information. In a DataWindow object, both the Oracle Timestamp and Date datatypes are mapped to the PowerBuilder DateTime datatype, which supports millisecond information.

If you use the O90 or O10 interface with an Oracle9*i* or higher server, DateTime and Time datatypes are mapped to the Oracle TimeStamp datatype by default. If you want DateTime and Time to be mapped to the Oracle Date datatype, set the TimeStamp database parameter to 0.

In PowerBuilder 8 and earlier, millisecond information was truncated when used with the Oracle Date datatype. In PowerBuilder 9.0 and later, millisecond information is not truncated. As a result, when performing multiple updates to a DateTime field that maps to a Date column, the first update succeeds, but subsequent updates fail with a “row changed between retrieve and update error.”

If you are using an interface with a database that uses only the TimeStamp datatype, PowerBuilder handles updates correctly. If the database uses the Oracle Date datatype only, set the DBParm TimeStamp to 0 to truncate millisecond information.

If you are using a database that uses both Date and TimeStamp datatypes, you must determine which columns use each datatype, and strip the milliseconds from the Date columns using code like the following:

```
datetime dt
dt=datetime (date(string( today() ,"dd/mm/yyyy")),  &
   time(string ( today() ,"hh:mm:ss ")))
   dw_1.setitem(1,3,dt)
```

Examples

To set the TimeStamp value to treat DateTime and Time DataWindow datatypes as Oracle Date datatypes:

- **Database profile** Clear the Map DateTime/Time to Oracle Timestamp check box on the Syntax page in the Database Profile Setup dialog box.
- **Application** Type the following in code:

```
SQLCA.DBParm="TimeStamp=0"
```

TraceFile

Description

Specifies that the JDBC Driver Manager Trace tool should trace a connection to the database you access through the PowerBuilder JDBC interface.

Applies to

JDB JDBC

Syntax

TraceFile='*tracefile_name*'

Default value

None

Usage

The JDBC Driver Manager Trace logs errors and informational messages originating from the Driver object currently loaded (such as the Sybase jConnect JDBC driver) when PowerBuilder connects to a database through the JDBC interface. It writes this information to a log file (default is JDBC.LOG) or to a file you specify. The amount of trace output varies depending on the JDBC driver being used.

You can start and stop the JDBC Driver Manager Trace in the PowerBuilder development environment by editing the database profile for the connection you want to trace. You can also start and stop the JDBC Driver Manager Trace in a PowerBuilder application by specifying the TraceFile parameter in the appropriate script.

For more information about using the JDBC Driver Manager Trace tool, see *Connecting to Your Database*.

Examples

To start the JDBC Driver Manager Trace and specify a log file:

- **Database profile** Select the Trace JDBC Calls check box and type the following in the Trace File box on the Options page in the Database Profile Setup dialog box:

```
c:\temp\jdbctrce.log
```

- **Application** Type the following in code:

```
SQLCA.DBParm="TraceFile='c:\temp\jdbctrce.log'"
```

TrimSpaces

Description

Specifies whether PowerBuilder 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

ADO.NET
ASE, SYC and SYJ Sybase Adaptive Server Enterprise
DIR Sybase DirectConnect
I10 Informix
IN9 Informix
JDB JDBC
ODBC
O90 Oracle9*i*
O10 Oracle 10*g*
SNC SQL Native Client for Microsoft SQL Server

Syntax

TrimSpaces=*value*

Parameter	Description
value	Specifies whether PowerBuilder 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=0 (ADO.NET, O90, O10, SNC, and IN9) or TrimSpaces=1 (DIR, ASE, and SYC). For JDBC, the default values depend on the PBTrimCharColumns value in the registry. For ODBC, the default values depend on the PBTrimCharColumns value in the PBODB*nnn*.INI file. (If the PBTrimCharColumns keyword is missing for a particular database connection, the default value for the ODBC interface is TrimSpaces=0.)

Usage

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

ODBC database interface
Some ODBC drivers, such as SQL Anywhere, trim trailing spaces before the data reaches the fetch buffer—even when TrimSpaces is set to 0.

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 might receive the message `Row changed between retrieve and update` when your DataWindow object's update properties are set to "Key and updateable columns." To prevent this, change your DataWindow object's 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 PowerBuilder should not trim trailing spaces:

- **Database profile** Clear the Trim Trailing Spaces In CHAR Data check box on the Syntax page in the Database Profile Setup dialog box.
- **Application** Type the following in code:

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

TrustedConnection

Description

Specifies whether the current Windows account credentials can be used for authentication.

When to specify TrustedConnection
You must specify the TrustedConnection parameter *before* connecting to the database.

Applies to

ADO.NET

SNC SQL Native Client for Microsoft SQL Server

Syntax

TrustedConnection=*value*

Parameter	Description
value	Specifies whether the current Windows account credentials can be used for authentication. Values are: • **0** (Default) The User ID and Password are specified in the connection. • **1** The current Windows account credentials are used for authentication. The User ID and Password supplied in the connection are ignored.

Default value

TrustedConnection=0

Examples

To specify that PowerBuilder should trust the connection:

- **Database profile** Select the Trusted Connection check box on the General page in the Database Profile Setup dialog box.
- **Application** Type the following in code:

```
SQLCA.DBParm="TrustedConnection=1"
```

TrustServerCertificate

Description

Specifies whether encryption occurs if there is no verifiable server certificate.

When to specify TrustServerCertificate
You must specify the TrustServerCertificate parameter *before* connecting to the database.

Applies to

SNC SQL Native Client for Microsoft SQL Server

Syntax

TrustServerCertificate=*value*

Parameter	Description
value	Specifies whether encryption occurs if there is no verifiable server certificate. The value of this parameter is ignored if the Encrypt DBParm is not set to 1. Values are: • **0** (Default) Encryption occurs only if there is a verifiable server certificate, otherwise the connection attempt fails. • **1** Encryption always occurs, but may use a self-signed server certificate.

Default value

TrustServerCertificate=0

Usage

SQL Server 2005 always encrypts network packets associated with logging in to the server. If no certificate is provided on the server when it starts up, SQL Server generates a self-signed certificate that is used to encrypt login packets.

SQL Server Configuration Manager can be used to configure the SQL Native Client to request an encrypted connection using the Secure Sockets Layer (SSL), and to accept a self-signed certificate without validation.

You can also request encryption by setting the Encrypt DBParm to 1, which sets the SQL Native Client connection string keyword Encrypt. To enable encryption to be used when a certificate has not been provided on the server, set both Encrypt and TrustServerCertificate. The value of TrustServerCertificate is ignored if Encrypt is not set.

Examples

To specify that PowerBuilder should encrypt data and accept the server certificate without validation:

- **Database profile** Select the Encrypt Data and Trust Server Certificate check boxes on the System page in the Database Profile Setup dialog box.
- **Application** Type the following in code:

```
SQLCA.DBParm="Encrypt=1,TrustServerCertificate=1"
```

See also

Encrypt

TRS

Description

Specifies whether you want your application to connect to a database through the DirectConnect server using:

DirectConnect Access Service for DB2/MVS
DirectConnect TRS
Gatewayless connection using OpenServerConnect

Applies to

DIR Sybase DirectConnect

Syntax

TRS=*value*

Parameter	Description
value	Specifies how your application should connect to a database through the DirectConnect server. Values are: • **0** (Default) Use an Access Service connection. • **1** Use a TRS connection. • **2** Use a gatewayless or OpenServerConnect connection.

Default value

TRS=0

Usage

If you have chosen to make a gatewayless database connection, you can then set the UseProcSyntax parameter to specify whether PowerBuilder should convert the syntax for invoking a Remote Stored Procedure (RSP) or host-resident request before executing that procedure.

Examples

To specify that you want to connect to a database using OpenServerConnect:

- **Database profile** Select Gatewayless from the Choose Gateway drop-down list on the Connection page in the Database Profile Setup dialog box.
- **Application** Type the following in code:

```
SQLCA.DBParm="TRS=2"
```

See also

UseProcSyntax
HostReqOwner

UnicharBind

Description

Specifies whether PowerBuilder binds string input parameters to the Char or Unichar datatype.

Applies to

ASE, SYC, and SYJ Sybase Adaptive Server Enterprise

Syntax

UnicharBind=*value*

Parameter	Description
value	Specifies whether PowerBuilder binds string input parameters to the Char or Unichar datatype. Values are: **0** Binds string input parameters as Char datatypes. **1** Binds string input parameters as Unichar datatypes.

Default value

UnicharBind=0

Usage

The default setting, UnicharBind=0, is recommended for binding Char, Varchar, and Text data. This binding encodes the string data as ANSI strings determined by the current client operating system's code page. UnicharBind=1 is recommended for binding strings as Unichar, Univarchar, and Unitext data.

DisableBind must be set to 0
For UnicharBind to take effect, the DisableBind parameter must be set to 0. DisableBind=1 overrides the UnicharBind setting.

Examples

To bind strings to unichar:

- **Database profile** Select the Bind String parameters as Unichar Type check box on the Transaction page in the Database Profile Setup dialog box.
- **Application** Type the following in code:

```
SQLCA.DBParm="UnicharBind=1"
```

See also

DisableBind
BindSPInput
NCharBind
RPCRebind

URL

Description

The location of the database to which you want to connect using the JDBC interface.

When to specify URL
You must specify the URL parameter *before* connecting to the database.

Applies to

JDB JDBC

Syntax

URL='*URL_address*'

Default value

None

Usage

The database URL is obtained from the database JDBC driver documentation. A list of registered Driver classes, with their URLs, is maintained by the driver's JDBC DriverManager class. When a connection request is made, the DriverManager attempts to locate a suitable driver from those listed.

The URL uses this general format:

jdbc:*vendor*:*driverprotocol*:*servername*:*port/database*

Argument	Description
jdbc	Driver
vendor	Database vendor (such as Sybase or Oracle)
driverprotocol	Database communications protocol
servername	DNS machine name or database host name
port	TCP/IP port number configured for accessing the database server
database	(optional) Name of a specific database

The database URL can also include the user ID and password as follows:

jdbc:*vendor*:*driverprotocol*:*userid*/*password*@*servername*:*port*:*database*

Examples

Example 1 To set the URL to a database accessed through jConnect:

- **Database profile** Type the following in the URL box on the Connection page in the Database Profile Setup dialog box:

```
jdbc:sybase:Tds:199.93.178.151:5007/tsdata
```

- **Application** Type the following in code:

```
SQLCA.DBParm="URL='jdbc:sybase:Tds:199.93.178.151:5
007/tsdata'"
```

Example 2 To set the URL to a database accessed through the Oracle JDBC driver:

- **Database profile** Type the following in the URL box on the Connection page in the Database Profile Setup dialog box:

```
jdbc:oracle:thin:@ora80nt:1521:orcl
```

- **Application** Type the following in code:

```
SQLCA.DBParm="URL='jdbc:oracle:thin:@ora80nt:1521:o
rcl'"
```

Example 3 To set the URL, which includes the user ID and password, to a database accessed through the Oracle JDBC driver:

- **Database profile** Type the following in the URL box on the Connection page in the Database Profile Setup dialog box:

```
jdbc:oracle:thin:system/manager@ora80nt:1521:orcl
```

- **Application** Type the following in code:

```
SQLCA.DBParm="URL='jdbc:oracle:thin:system/manager@
ora80nt:1521:orcl'"
```

See also

Driver

UseContextObject

Description

Specifies that PowerBuilder controls a transaction using the transaction service context object TransactionServer. This parameter applies *only* when a PowerBuilder custom class user object is deployed as an EAServer or COM+ component.

Applies to

JDB JDBC
ODBC

O90 Oracle9*i*
SYJ Sybase Adaptive Server Enterprise

Database interface restrictions

EAServer requires the SYJ interface Sybase EAServer uses a slightly different version of the CT-Lib software. Therefore, *at runtime*, you need to use the SYJ database interface rather than ASE or SYC to connect to an Adaptive Server Enterprise database. The SYJ Database Profile Setup dialog box provides a convenient way to set the appropriate connection parameters and then copy the syntax from the Preview page into the script for your Transaction object.

You cannot use the SYJ interface, however, to connect to the database in the PowerBuilder development environment. Therefore, *during the development phase* (before the component has been deployed to EAServer), you must use ASE or SYC to connect to the database.

Syntax

UseContextObject='*value*'

Default value

UseContextObject=No

Usage

PowerBuilder provides a transaction service context object called TransactionServer that gives you access to the transaction state primitives that influence whether EAServer or COM+ commits or aborts the current transaction. If you use the TransactionServer context object by setting the UseConnectObject parameter to Yes, COMMIT and ROLLBACK statements called on the Transaction object result in a runtime error.

By default, the TransactionServer context object is not used. Instead, you can use COMMIT and ROLLBACK statements to manage transactions. In this case, COMMIT is interpreted as a SetComplete function and ROLLBACK is interpreted as a SetAbort function. This approach is recommended only when you want to migrate PowerBuilder 6.x or prior objects to EAServer or COM+ without modifying the code.

Setting UseContextObject to Yes is incompatible with the use of the SetTrans function. The SetTrans function is used when you want the DataWindow engine to manage database connections, transaction state primitives, and related EAServer component deactivation.

Because the default Web DataWindow component uses SetTrans to specify database connection information, you must not set UseContextObject to Yes in your database profile or in the EAServer properties for the component.

For information on how to use PowerBuilder to build EAServer and COM+ components, see *Application Techniques*.

Examples

To use the TransactionServer context object:

- **Database profile** Select the Use Transaction Context Object check box on the EAServer or EAServer/COM+ page in the Database Profile Setup dialog box.
- **Application** Type the following in code:

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

See also

CacheName
GetConnectionOption
ReleaseConnectionOption

UseProcSyntax

Description

Specifies whether PowerBuilder should convert the syntax for invoking a Remote Stored Procedure (RSP) or host-resident request before executing that procedure.

Applies to

DIR Sybase DirectConnect (applies only to Open ServerConnect)

Syntax

UseProcSyntax=*value*

Parameter	Description
value	Specifies whether PowerBuilder should convert RSP or host-resident request syntax. Values are: • **0** (Default) Do not convert syntax. • **1** Convert syntax to USE PROCEDURE.

Default value

UseProcSyntax=0

Usage

UseProcSyntax applies *only* if you are using Open ServerConnect to make a gatewayless database connection to a DB2/MVS database. To indicate that you want to use Open ServerConnect, select Gatewayless from the Choose Gateway drop-down list on the Connection page. If you select Gatewayless, and set the UseProcSyntax parameter to 1, PowerBuilder converts the EXECUTE procedure syntax it normally uses to the USE PROCEDURE syntax required to invoke RSP and host-resident requests.

Examples

To specify that PowerBuilder should convert RSP or host-resident syntax:

- **Database profile** Select the Use Procedure Syntax for RSPs check box on the Syntax page in the Database Profile Setup dialog box.

- **Application** Type the following in code:

```
SQLCA.DBParm="UseProcSyntax=1;TRS=2"
```

See also

TRS
HostReqOwner

UTF8

Description

The UTF8 database parameter specifies whether the database server you are accessing will handle conversion between the character sets on the client and server when they are different.

When to specify UTF8
You must specify a value for UTF8 *before* connecting to the database in PowerBuilder.

Applies to

ASE, SYC and SYJ Sybase Adaptive Server Enterprise
DIR Sybase DirectConnect

Syntax

UTF8=*value*

Parameter	Description
value	Specifies whether the database server you are accessing will perform conversion between the character sets used on the client and server. Values are: **0** (Default) The PowerBuilder database interface performs the conversion. Multiple languages are not supported. **1** Your database server performs the conversion. Multiple languages are supported.

Default value

UTF8=0

Usage

If UTF8 is set to 1, PowerBuilder always uses UTF-8 as the client character set when connecting to an Adaptive Server database. When UTF8 is set to 0, if the client and database server character sets are different, the database interface converts Transact-SQL, identifiers, parameters, and Char and VarChar data to and from the character set used on the server. Multiple languages are not supported with this setting.

To enable multilanguage support, you must set the UTF8 database parameter to 1 and the database server must have the UTF-8 character set installed, or, for Adaptive Server, it must be configured to support Unicode conversions. To do so, the database administrator must run the following command on the server:

```
sp_configure "enable Unicode conversion", 2
```

This enables the server to perform the conversion to and from Unicode.

Examples

To specify that the database server you are accessing with PowerBuilder uses UTF-8 as its default character set:

- **Database profile** Select the UTF8 Character Set Installed or Unicode Conversion Enabled check box on the Regional Settings page in the Database Profile Setup dialog box.
- **Application** Type the following in code:

```
SQLCA.DBParm="UTF8=1"
```

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 PowerBuilder.

Contents

The database preferences are listed in alphabetical order.

Database preferences and supported database interfaces

The following table lists each supported database interface and the connection-related database preferences you can use with that interface in PowerBuilder. The preferences listed in the table pertain to the database connection, and not to the behavior of the Database painter itself.

Database interface	Database preferences
ADO.NET	AutoCommit Connect to Default Profile Keep Connection Open Read Only Shared Database Profiles SQL Terminator Character Use Extended Attributes
ASE Sybase Adaptive Server Enterprise	AutoCommit Connect to Default Profile Keep Connection Open Lock Read Only Shared Database Profiles SQL Terminator Character Use Extended Attributes

Database interface	Database preferences
DIR Sybase DirectConnect	AutoCommit Connect to Default Profile Keep Connection Open Lock Read Only Shared Database Profiles SQL Terminator Character Use Extended Attributes
IN9 and I10 Informix	AutoCommit Connect to Default Profile Keep Connection Open Lock Read Only Shared Database Profiles SQL Terminator Character Use Extended Attributes
JDBC	AutoCommit Connect to Default Profile Keep Connection Open Lock Read Only Shared Database Profiles SQL Terminator Character Use Extended Attributes
ODBC **Using AutoCommit and Lock with ODBC** The AutoCommit and Lock database preferences are supported by the ODBC interface only if *both* the ODBC driver you are using and the back-end DBMS support the feature.	AutoCommit Connect to Default Profile Keep Connection Open Lock Read Only Shared Database Profiles SQL Terminator Character Use Extended Attributes
OLE DB	AutoCommit Connect to Default Profile Keep Connection Open Read Only Shared Database Profiles SQL Terminator Character Use Extended Attributes

Database interface	Database preferences
O90 Oracle9*i* O10 Oracle 10*g* ORA Oracle 11*g*	Connect to Default Profile Keep Connection Open Read Only Shared Database Profiles SQL Terminator Character Use Extended Attributes
SNC SQL Native Client	AutoCommit Connect to Default Profile Keep Connection Open Lock Read Only Shared Database Profiles SQL Terminator Character Use Extended Attributes
SYC Sybase Adaptive Server Enterprise	AutoCommit Connect to Default Profile Keep Connection Open Lock Read Only Shared Database Profiles SQL Terminator Character Use Extended Attributes
SYJ Sybase Adaptive Server Enterprise	AutoCommit Connect to Default Profile Keep Connection Open Lock Read Only Shared Database Profiles SQL Terminator Character Use Extended Attributes

AutoCommit

Description

For those DBMSs and database interfaces that support it, AutoCommit controls whether PowerBuilder issues SQL statements outside or inside the scope of a transaction.

When AutoCommit is set to False (the default), PowerBuilder issues SQL statements *inside* the scope of a transaction. When AutoCommit is set to True, PowerBuilder 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. AutoCommit takes effect only when the database connection occurs. Changes to AutoCommit after the connection occurs have no effect on the current connection.

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

Applies to

ADO.NET
ASE, SYC and SYJ Sybase Adaptive Server Enterprise
DIR Sybase DirectConnect
I10 Informix
IN9 Informix
JDB JDBC
ODBC (if driver and back-end DBMS support this feature)
OLE DB
SNC SQL Native Client for Microsoft SQL Server

In an application

For those DBMSs and database interfaces that support it, 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 PowerBuilder issues SQL statements outside or inside the scope of a transaction. Values are: • **True** PowerBuilder issues SQL statements *outside the scope of a transaction*. The statements are not part of a logical unit of work (LUW). If the SQL statement is successful, the DBMS updates the database immediately as if a COMMIT statement had been issued. • **False** (Default) PowerBuilder issues SQL statements *inside the scope of a transaction*. PowerBuilder 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.

For instructions, see “Setting Additional Connection Parameters” in *Connecting to Your Database*.

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.

Executing SQL DDL statements Some DBMSs require you to execute certain SQL statements outside the scope of a transaction. For example, when connected to a SQL Server 7 or earlier database, you must execute SQL Data Definition Language (DDL) statements such as CREATE TABLE and DROP TABLE outside a transaction. There are two reasons for this:

- It ensures that the structure of your database cannot change during a transaction.
- It improves database performance, because DDL statements are costly operations to recover.

Therefore, to execute DDL statements or stored procedures containing DDL statements in a SQL Server database, you must set AutoCommit to true to issue the DDL statements outside the scope of a transaction. You should, however, set AutoCommit back to false immediately after executing the DDL statements.

When you change the value of AutoCommit from false to true, PowerBuilder 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*.

DirectConnect interface As part of the Connect process, the DIR interface automatically issues TransactionMode=short to override the access service default configuration. It then issues begin transaction at connect time and after every Commit and Rollback whenever AutoCommit=False. Most developers should start their connections with AutoCommit=True, switch to False only when the application demands transaction processing, and then switch back to AutoCommit=True after the transaction is committed or rolled back.

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.
- **Application** Type the following in a script:

```
SQLCA.AutoCommit=True
```

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

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 not selected, the Database painter opens without establishing a connection to a database.

Applies to

All database interfaces

In an application

You *cannot* set the Connect to Default Profile database preference in code.

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, PowerBuilder opens a database connection the first time you open a painter requiring a connection, and stays connected throughout the session until you exit.

When you connect to a database in the PowerBuilder development environment without using a database profile, you can set the Keep Connection Open database preference to specify when PowerBuilder closes the database connection.

Keep Connection Open applies only when connecting to a database in the PowerBuilder development environment without using a database profile. The setting of Keep Connection Open has no effect when you use a database profile to connect in PowerBuilder.

Applies to

All database interfaces (only in the development environment)

In an application

You *cannot* set the Keep Connection Open database preference in code.

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 PowerBuilder 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 the development environment** You must be working in the development environment.
- **Using default connection information** PowerBuilder 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, and closes the connection at other times. This can save you money if you are accessing a database that charges for connect time.

Lock

Description

For those DBMSs and database interfaces that support the use of lock values and isolation levels, the Lock preference sets the isolation level to use when connecting to the database.

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.

PowerBuilder 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. 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.

Applies to

ASE, SYC and SYJ Sybase Adaptive Server Enterprise
DIR Sybase DirectConnect
I10 Informix
IN9 Informix
JDB JDBC
ODBC (if driver and back-end DBMS support this feature)
OLE DB
SNC SQL Native Client for Microsoft SQL Server

In an application

For those DBMSs and database interfaces that support it, you can set the Lock value in code 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 each database interface that supports locking. You set the lock value in code, and the isolation level in a database profile.

For more about the isolation levels that your DBMS supports, see your DBMS documentation.

Database interface	Lock values	Isolation levels
IN9 and I10 Informix (for OnLine databases only)	Dirty Read	Dirty Read
	Committed Read	Committed Read
	Cursor Stability	Cursor Stability
	Repeatable Read	Repeatable Read
JDB JDBC	RU	Read Uncommitted
	RC	Read Committed
	RR	Repeatable Read
	TS	Serializable Transactions
	TN	Transaction None

Database interface	Lock values	Isolation levels
ODBC	RU	Read Uncommitted
	RC	Read Committed
	RR	Repeatable Read
	TS	Serializable Transactions
	TV	Transaction Versioning
OLE DB	RU	Read Uncommitted
	RC	Read Committed
	RR	Repeatable Read
	TS	Serializable Transactions (default)
	TC	Chaos
SNC SQL Native Client	RU	Read Uncommitted
	RC	Read Committed (default)
	RR	Repeatable Read
	SS	Snapshot
	TS	Serializable Transactions
	TC	Chaos
Sybase Adaptive Server Enterprise	0	Read Uncommitted
	1	Read Committed (default)
	3	Serializable Transactions
Sybase DirectConnect	0	Read Uncommitted
	1	Read Committed (default)
	2	Repeatable Read
	3	Serializable Transactions

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.

For instructions, see “Setting Additional Connection Parameters” in *Connecting to Your Database*.

Default value

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

Usage

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

OLE DB The default value for Lock in the discontinued MSS native interface and the SNC interface for Microsoft SQL Server 2005 is Read Committed, but for OLE DB the default is Serializable Transactions. If you want to connect to SQL Server 2000 using OLE DB, you can override the default value by specifying a value for Lock in the *PBODB125.INI* file. For example:

```
[Microsoft SQL Server]
...
LOCK='RC'
...
```

The value in the *PBODB125.INI* file is used if you do not change the default in the database profile or set the Lock parameter of the Transaction object in code.

Sybase Adaptive Server Enterprise Sybase Adaptive Server Enterprise supports the following lock values, which correspond to SQL Server isolation levels:

- **0—Read Uncommitted (dirty reads)** Isolation level 0 prevents other transactions from changing data that an uncommitted transaction has already modified (through SQL statements such as UPDATE).

  Other transactions cannot modify the data until the transaction commits, but they can still read the uncommitted data (perform dirty reads). Isolation level 0 prohibits retrieval locks on tables or pages.

  Isolation level 0 is valid only for Sybase System 10 or higher databases.

- **1—Read Committed** (Default) Isolation level 1 prevents dirty reads by issuing shared locks on tables or pages.

  A **dirty read** occurs when one transaction modifies a table row and a second transaction reads that row before the first transaction commits the change. If the first transaction rolls back the change, the information read by the second transaction becomes invalid.

- **3—Serializable Transactions (HOLDLOCK behavior)** Isolation level 3 prevents dirty reads, nonrepeatable reads, and phantoms for the duration of a transaction.

  A **nonrepeatable read** occurs when one transaction reads a row and then a second transaction modifies that row. If the second transaction commits the change, subsequent reads by the first transaction produce different results than the original read.

A phantom occurs when one transaction reads a set of rows that satisfy a search condition, and then a second transaction modifies that data through a SQL INSERT, UPDATE, or DELETE statement. Subsequent reads by the first transaction using the same search conditions produce a different set of rows than the original read.

Dynamically controlling the isolation level PowerBuilder makes a second connection to implement either of the following while connected to a Sybase Adaptive Server Enterprise database:

- The Retrieve.AsNeeded property to specify that a DataWindow should retrieve only as many rows as needed from the database
- A SELECTBLOB embedded SQL statement to select a single blob column in a specified table row

The lock value you set before making the first Adaptive Server Enterprise connection is automatically inherited by the second connection, and *cannot be changed for the second connection*.

However, you can dynamically control the isolation level for the first (original) Adaptive Server Enterprise connection in an application by coding the following PowerScript embedded SQL statement, where *n* is 0, 1, or 3 for the isolation level you want to set for the first connection:

```
EXECUTE IMMEDIATE "set transaction isolation level n"
```

For example, the following PowerScript embedded SQL code specifies isolation level 0 (dirty read behavior) for the second connection, and isolation level 1 (read committed behavior) for the first connection:

```
// Isolation level inherited by second connection
SQLCA.Lock="0"
CONNECT USING SQLCA;
// Override lock value 0 for first connection only
EXECUTE IMMEDIATE "set transaction isolation level 1";
```

Use in three-tier applications If an ASE connection on an application server, such as EAServer, is used by a component with a specified isolation level and cached by the server, it is released back into the connection pool with the isolation level set by the component. If that connection is then used by another component that has no specified isolation level, the isolation level may not be the default level expected by the component (1). This could result in the occurrence of deadlocks. To avoid this, always set the SQLCA.Lock property explicitly in application server components.

Examples

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

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

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

Example 2   To set the Lock value to 3 (Serializable Transactions) for a Sybase Adaptive Server Enterprise database:

- **Development environment**   Select Serializable Transactions from the Isolation Level drop-down list in the Database Profile Setup dialog box.
- **Application**   Type the following in a script:

```
SQLCA.Lock="3"
```

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

Read Only

Description

Read Only specifies whether PowerBuilder 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 PowerBuilder updates the extended attribute system tables and other tables in your database when you make changes.

If you select the Read Only check box, PowerBuilder *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.

Applies to

All database interfaces

In an application

You *cannot* set the Read Only database preference in code.

In the development environment

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

- **Select the check box** PowerBuilder 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) PowerBuilder 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 painter
- SELECT statements in embedded SQL

See also

Use Extended Attributes

Shared Database Profiles

Description

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

For instructions on sharing database profiles in the PowerBuilder development environment, see “Managing Database Connections” in *Connecting to Your Database*.

Applies to

All database interfaces

In an application

You *cannot* set the Shared Database Profiles database preference in code.

In the development environment

In the Database painter, supply the path name of the PowerBuilder 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.

For instructions, see “Setting Additional Connection Parameters” in *Connecting to Your Database*.

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\PB.INI* on the Windows platform, type or browse to the following in the Shared Database Profiles box in the Database Preferences dialog box:

```
I:\SHARE\PB.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 (`) character.

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

Applies to

All database interfaces

In an application

You *cannot* set the SQL Terminator Character database preference in code.

In the development environment

In the Database Preferences dialog box in the Database painter, type the terminator character you want to use in the SQL Terminator Character box. For instructions, see "Setting Additional Connection Parameters" in *Connecting to Your Database*.

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 a 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 PowerBuilder 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.

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 PowerBuilder.

Applies to

All database interfaces

In an application

You *cannot* set the Use Extended Attributes database preference in code.

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 connecting 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 painter use the appropriate default values for extended attributes (such as headers, labels, and text color). If the extended attribute system tables already exist, PowerBuilder 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, PowerBuilder *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