SQL

Stereotype Attributes

SQL Alias

Attribute	Description	Allowed Values / Examples
Name	Specify a name for the alias.	any string
Standard		
user	Specify the database user. Optionally, the password can be given after a '/', however, this is recommended for development purposes only.	**db_user/db_password**
dbType	Type of the database.	Oracle, SQLServer, InterBase, SQLBase, ODBC, DB2, Informix, Sybase, MySQL, PostgreSQL, SQLite, DBTypeVariable
options	This attribute can hold a comma separated list of <name>=<value> pairs. These list elements are interpreted as native options. The possible name-value pairs depend on the database type. A comprehensive list can be found at https://www.sqlapi.com/ApiDoc/servers/	**SSPROP_INIT_ENCRYPT=VARIANT_TRUE**
dbTypeVariable	If the attribute **dbType** is set to **DBTypeValue**, the **dbTypeVariable** attribute is used to define the type of the database. The **dbType** then can be defined by a setting variable. This is to handle the case, that you not want to hard code the **dbType** , but to configure it at runtime via the PAS integration component (Bridge). See Bridge Guide > Using Global Setting Variables. Use one of the listed **dbTypes** in the settings variable. If you configure an unknown **dbType** via the PAS integration component (Bridge), the xUML Runtime will throw an error on service startup.	Any global setting variable from the PAS integration component (Bridge): **{{my_setting_variable}}**
Advanced		
tableQualifier	Specify a string that prefixes tables. For example, if **tableQualifier** is set to TQ1, all tables accessing the current database are prefixed by "TQ1", e.g. `TQ1Customers`. If schema and table qualifier are both given, all tables will become: <schema>.<tableQualifier><tableName>. This works only if the tables are marked using the `TABLE::` keyword, e.g `TABLE::Customers` in SQL statements. If you do not prefix the table name by `TABLE::`, the table name is used as it is.	
schema	Specify a string that prefixes tables and stored procedures. For example, if schema is set to S1, all tables accessing the current database are prefixed by "S1.". This works only if the tables are marked using the `TABLE::` keyword, e.g `TABLE::Customers` in SQL statements. If you do not prefix the table name by `TABLE::`, the table name is used as it is.	
charset	Any database uses a charset to encode strings. If the database uses UNICODE charsets (like UTF-8, UTF-16, UTF-32), encoding is handled automatically. If the database is not UNICODE compliant, the xUML Runtime assumes 7-bit ASCII by default. However, in many cases it necessary to define the charset explicitly. This is done by the attribute **charset**. The charset needs to be the same as defined in the database settings. All possible charset definitions are listed in section Charset Definitions.	**UTF-8** See Charset Definitions for a list of possible values.

On this Page:

SQLAdapter_CustomerData_Example

Click the icon to download a simple example model that shows the usage of the SQL adapter in **Scheer PAS** *Designer*.

Related Pages:

- SQL Adapter
- Database Specifics
  - Database Server-Specific Notes for SQL Adapters
  - Database-Specific Mappings

timezone	Specify a valid time zone or the value **local**, which uses the time zone of the xUML service. See Time Zones for a list of possible values. If **timezone** does not contain any content (is NULL), UTC is used.	**local** (default) **"Australia/Melbourne", "CET", "Etc/GMT+10"**	
dbConnectionString	Specify the database connection string. The format depends on the type of the database. For more details see Database Server-Specific Notes for SQL Adapters.		
transactionIsolationLevel	Specify here the required transaction isolation level of the SQL connection according to SQL-92 standard. Refer to Wikipedia for a detailed description of the available isolation levels. Please note: - that not all databases support all levels. In this case a database-specific mapping will occur. - that for persistent state databases no other than **<UNSPECIFIED>** and **DBMS default** are allowed.	**DBMS default**	Use the default isolation level of the connected database system (default).
		Read uncommitted	Lowest isolation level. Dirty reads allowed, SQL adapter may fetch not-yet-committed changes of other transactions.
		Read committed	Lock-based concurrency control.
		Repeatable read	Lock-based concurrency control.
		Serializable	Highest isolation level. Lock-based concurrency control.
unicodeMode	Specify the encoding for database access. We recommend to use the **Platform default** unless you suspect an encoding incompatibility (see Troubleshooting the SQL Adapter).	**Platform default**	Use the platform default mode (default). This is - Unicode: for Windows systems - non-Unicode: for all others This option is backwards compatible to older Runtimes.
		Unicode	Force Unicode mode.
		non-Unicode	Force non-Unicode mode.
Pooling			
maxConnectionAge	Specify a maximum connection age in minutes. After this period of time, the connection will be closed and removed from the pool.	an integer value	
		15	Keep the connection for 15 minutes (default).
		-1	Keep the connection forever.
maxConnectionIdleTime	Specify a maximum idle time in minutes. Connections not used for the time specified will be closed and removed from the pool. This is useful for connections going through firewalls because such connections might be cut off after some time.	an integer value	

		60	Close the connection after 60 minutes of idle time.
maxConnectionReuse	This attribute controls how often a connection can be reused. After the connection has been reused the number of times specified with **maxConnectionReuse**, it will be closed and not put back into the pool. This feature has been introduced because some databases might have problems if one connection is reused too often. Value **-1** means the connection will be reused forever. In this case you should define reasonable values for **maxConnectionAge** or **maxConnectionIdleTime** (see above). Note that a pooling defined with **connectionPooling** (see below) is implicitly switched off, if **maxConnectionReuse** is set to 0.	an integer value	The number of connections to be pooled.
		1000	Allow for a pool size of 1000 connections (default).
		0	Pooling is implicitly switched off.
		-1	Connections are pooled forever.
connectionPooling	This attribute controls the connection pooling. If true, each connection is put into a pool after use. If an SQL adapter requires a connection, it is taken from the pool. If no connection is available, a new connection is being created and put into the pool after use. The time the connection is kept in the pool depends on the other pooling parameters.	**true**	Database connections are pooled (default).
		false	Database connections are not pooled.

SQL Adapter

Attribute	Description	Allowed Values	
alias	Specify the SQL alias resp. the database the adapter should use to establish the connection.	any valid SQL alias	
sql	Specify the SQL statement to be performed on the database, or specify a **commit** or **rollback** command.	any valid SQL statement as a string	
		commit	Explicitly commit the recent changes to the database.
		rollback	Explicitly rollback the recent changes to the database.
dbType	Overwrite the database type defined in the SQL alias.		
action	The adapter action derives from the used operation. Do not configure this.		

SQL Adapter Operations

closeHandle Operation

- `closeHandle ( handle : SQLHandle )`

Attribute	Type	Direction	Description	Allowed Values / Examples
handle	SQLHandle	in	An SQL connection handle. The connection handle identifies a structure that contains connection information.	

execute Operations

- `execute ( affectedRows : Integer, result : Any, sql : String )`
- `execute ( affectedRows : Integer, result : Any[], sql : String )`

Attribute	Type	Direction	Description	Allowed Values / Examples
sql	String	in	Use this parameter to provide a dynamic SQL statement.	
result	Any or Array of Any	out	Result set of the database query. The SQL adapter tries to match the table column names with the attribute names of the output class. For information on type mapping refer to Database-Specific Mappings. Use `execute` with `result : Any` when you expect only one row to be return, Use `execute` with `result : Any []` when you expect multiple rows.	
affectedRows	Integer	out	This parameter returns the number of rows affected by the SQL statement.	

fetchNext Operation

- `fetchNext ( handle : SQLHandle, result : Any )`

Attribute	Type	Direction	Description	Allowed Values / Examples
handle	SQLHandle	in	An SQL connection handle. The connection handle identifies a structure that contains connection information.	
result	Any	out	Result set of the database query. The SQL adapter tries to match the table column names with the attribute names of the output class. For information on type mapping refer to Database-Specific Mappings.	

getHandle Operation

- `getHandle ( affectedRows : Integer, handle : SQLHandle, sql : String )`

Attribute	Type	Direction	Description	Allowed Values / Examples
sql	String	in	Use this parameter to provide a dynamic SQL statement.	
handle	SQLHandle	out	An SQL connection handle. The connection handle identifies a structure that contains connection information.	
affectedRows	Integer	out	This parameter returns the number of rows affected by the SQL statement.	

SQL Adapter Parameter Types

SQLHandle

Name	Type	Description
id	String	ID of the SQL connection handle.