Getting Started with Persistent State Databases

To save the states of business objects, the Persistent State engine needs a database. This page explains the preliminary steps to get started with Persistent State databases.

On this Page:

Related Pages:

- Creating an Alias
- Creating a Component Diagram

SQLite

In its default configuration, Persistent State does not need any additional components or installation steps. SQLite, an embedded, small footprint, relational database is provided with the Base firmware package. The database model is automatically initialized upon start-up and updated if a new firmware version should require it.

With SQLite, a separate database file is kept in the home directory of the deployed service instance. The filename is **pstate.db** and the command shell provided by the SQLite project (http://www.sqlite.org) can explore its contents. There are also several free or inexpensive visual management tools. We recommend SQLite2008 Pro Enterprise Manager of OsenXPSuite (http://www.osenxpsuite.net/?xp=3).

Other RDBMS

Persistent State does work with any other RDBMS platform that is supported by the Bridge using the <<SQLAdapter>>. The following RDBMS are tested and supported: **Oracle, Microsoft SQL Server** and **MySQL** (since xUML Runtime 6.0.91.1).

Like with normal use of RDBMS in the Bridge, the connection parameters to the database are configured as part of the component diagram. A database user and an empty instance/schema have to be created on the RDBMS before it can be accessed.
With these preconditions, the database model is automatically initialized upon start-up and updated if a new firmware version should require it.

> Even though multiple services may share the same database, it is recommended to use separate schemes/instances for each service. Using separate schemes/instances will increase the performance, especially if a large number of persistent state objects is given.
> Shared schemes are mandatory, however, for load balanced persistent state setups.

In case of an xUML Runtime update that modifies the database schema, you will experience a downtime for all services sharing a common database schema. A modification of the database schema will be announced in the xUML Runtime Release Notes.

Oracle

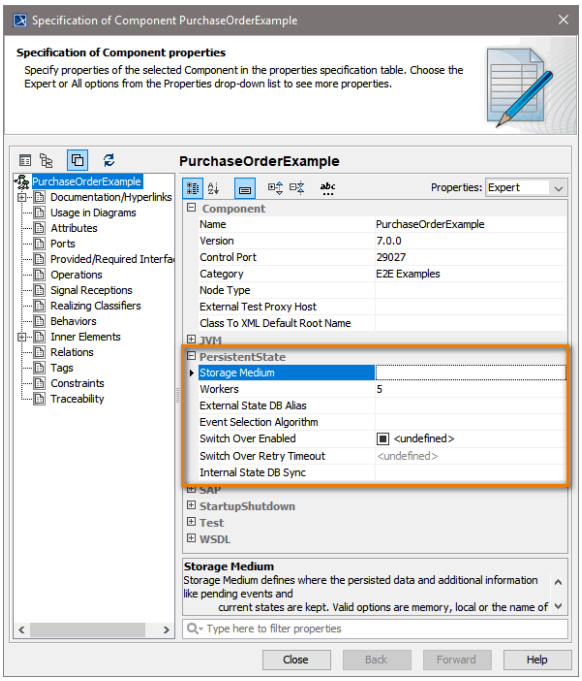
Using Oracle as a persistent state database has some advantages:

- A single xUML service can have several persistent state workers. If you use Oracle, every worker will get his own database connection.
  For other RDBMS like Microsoft SQL Server, MySQL or SQLite, all workers will use the same database connection. This means that all database operations will be serialized.
- With shared databases, Oracle has a better performance because of a better locking implementation for the persistent state use case.
  With Microsoft SQL Server, there may sometimes happen temporary deadlocks which decrease the performance.

Configuring the Persistent State Engine

In the specification dialog of the xUML service composite, you can configure the persistent state engine.

Figure: Specification of xUML Service Composite PurchaseOrderExample



For further details, see Persistent State Components.

A new alias can be created during the Creation of a Component Diagram or manually, as described on Backend Components.

> If the database is unknown at the time of service development, create a placeholder SQL alias anyway. The specifics to be applied at runtime can be edited on the Bridge (see xUML Service Settings) without the need to change the model.