# Backend Components

Components utilized by E2E services are called backend components or backend services. Their configuration is depicted in the component diagram.

Component Diagram

The following component diagram shows an E2E service that uses an SQLite database as backend. This backend is modeled as an alias <<SQLAlias>> **Employee** connecting to the <<E2EComposite>> **SqlQueriesExample**. The <<SQLAlias>> contains configuration information of the specific database or more generally speaking: it configures the backend component accessed by the xUML service. SQL databases and other service types accessible by the Bridge are described in the xUML Service Adapters section.
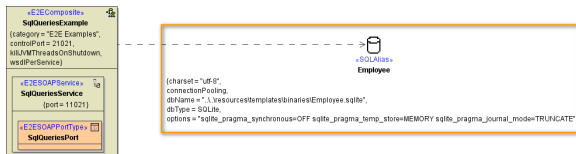
Aliases

Aliases are stereotyped UML artifacts. Their purpose is to link E2E adapters (defined in the activity diagram) with configuration settings of accessed backend systems. The aliases are defined and configured in the component diagram. For instance, when accessing database systems, the alias is a name that points to the physical database. When the backend system is a Web Service, the alias is a pointer to the port definition. It is recommended to put all aliases into a top level package called **Aliases**:

*Figure: Package Aliases in Containment Tree*
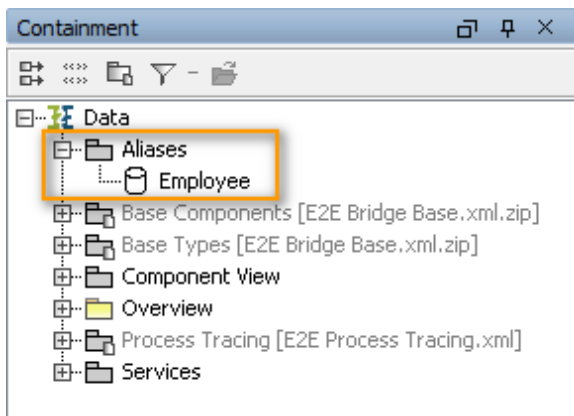
In activity diagrams, only the alias artifacts are assigned to backend adapters. This means that the logical view is separated from the physical view. For example, the following SQL adapter inserts data into the database referred to by the alias **Employee**.

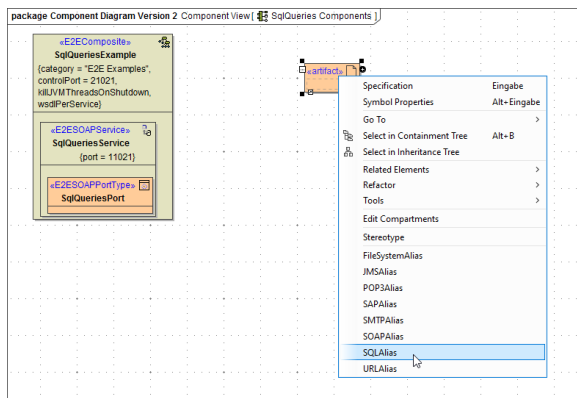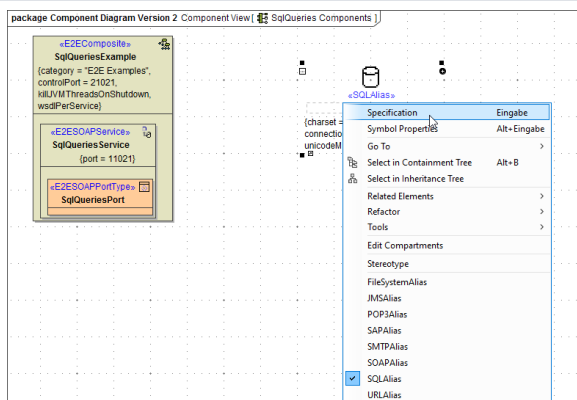*Figure: Example of an Alias Reference in an Adapter*

Creating an Alias

There are two different ways to create a new alias: During the Creation of the Component Diagram in the Components Wizard, or manually on the diagram pane of the component diagram.
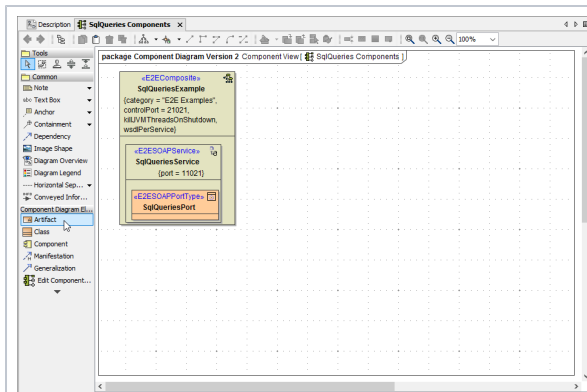
To create the alias, drag and drop an artifact on the diagram pane.

Right-click the artifact and assign the stereotype corresponding to the alias you need, in our example **SQLAlias**.
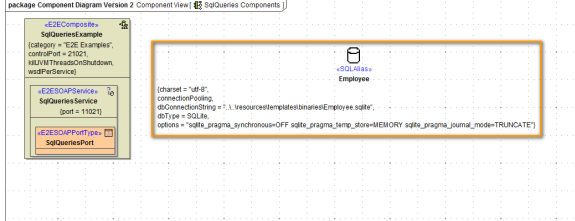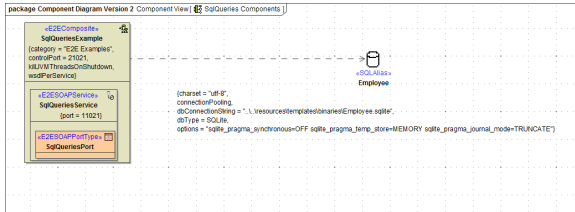

Open the specification dialog of the new alias.

You are now able to enter all necessary data such as alias name, database type etc. in the specification dialog.

Close the window, if you have finished.

package Component Diagram Version 2 Component View [SqlQueries Components] «E2EComposite» **SqlQueriesExample** (category = "E2E Examples", controlPort = 21021, killJVMThreadsOnShutdown, wsdlPerService) «E2ESOAPService» **SqlQueriesService** [port = 11021] «E2ESOAPPortType» **SqlQueriesPort** «SQLAlias» **Employee** {charset = "utf-8", connectionPooling, dbConnectionString = "..\..\resources\templates\binaries\Employee.sqlite", dbType = SQLite, options = "sqlite_pragma_synchronous=OFF sqlite_pragma_temp_store=MEMORY sqlite_pragma_journal_mode=TRUNCATE"}	The alias is displayed on the diagram pane including its specifications.
	Finally, you need to connect the <<E2EComposite>> with the <<SQLAlias>> . Select the dependency icon from the diagram toolbar and draw a dependency from the composite to the alias.
package Component Diagram Version 2 Component View [SqlQueries Components] «E2EComposite» **SqlQueriesExample** (category = "E2E Examples", controlPort = 21021, killJVMThreadsOnShutdown, wsdlPerService) «E2ESOAPService» **SqlQueriesService** [port = 11021] «E2ESOAPPortType» **SqlQueriesPort** «SQLAlias» **Employee** {charset = "utf-8", connectionPooling, dbConnectionString = "..\..\resources\templates\binaries\Employee.sqlite", dbType = SQLite, options = "sqlite_pragma_synchronous=OFF sqlite_pragma_temp_store=MEMORY sqlite_pragma_journal_mode=TRUNCATE"}	The created alias is now connected to the composite.