Actions

Actions are used to describe activities applied to objects or the environment. They may contain so-called action scripts, or can be stereotyped signaling a special kind of an action, like for instance the stereotype <<SQLAdapter>> that is used to access a database. All stereotyped actions are implemented as add-ons. This modular architecture makes it simple to enhance the Bridge UML profile, because new adapter modules add additional stereotypes and tagged-values can be added in a simple way.

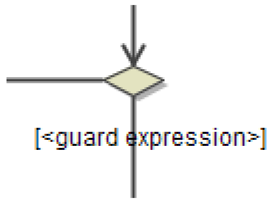
On this Page:

Related Pages:

- xUML Service Adapters

Action Scripts

Actions can hold action scripts. Action scripts execute actions following the Action Semantics for the UML defined in [7]. Action scripts can be used in the script section of actions or in guard expressions of decision transitions:



The Server implements parts of the Action Semantics for the UML in its E2E Action Language (EAL). This language is described in xUML Action Language.
Action scripts execute actions on objects. However, if you want to integrate systems and platforms having action semantics different to the Action Semantics for the UML (databases, legacy systems, operating systems, and so on), it is desirable to model these actions as well in UML actions. This can be done using adapters (see below).

Add-ons

While action scripts execute actions on objects it is frequently necessary to perform actions in systems not directly modeled in UML using languages native to these systems. For example, querying and updating relational databases is best done using SQL statements. Executing batch programs or system utilities is often done most naturally using means of the operating system. To describe such actions working on the environment we use action scripts having a stereotype. In this context, the environment is called E2E backend because it is logically situated behind the Bridge that offers Services for clients before it – i.e. in the E2E frontend. Using the MDA® terminology, stereotype actions implement platform dependent behavior. This behavior is not only dependent on the platform the Bridge is running on but also on the platforms, the Bridge is accessing.

However, all these actions consume and create objects like actions defined using an Action Language.

In order to access a backend system, we still use an action with an action script. The difference lies in the fact that you must type the action state with a certain stereotype, e.g. <<SQLAdapter>>. The action script will contain an SQL statement or the name of a stored procedure. If you wish to use a <<SOAPServ ice>>, you will not have an action script, but tagged values ("call" and "alias") which are references to the called service.

Each add-on communicating with external systems is called adapter.

Refer to section xUML Service Adapters for more information on the available adapters and their features.

Each adapter has at least an **alias** tagged value. This tagged value is an UML artifact referring to a dependency in the deployment diagram. This dependency points to an object (e.g. a database, Web Service, file, etc.) that holds all physical configuration information for the adapter. Thus we separate the logical use of an adapter from its deployment configuration – for instance on which machine, port listens to a given database.

The tagged values of other add-ons than adapters depend on the usage, that is, there is no common meaning. Examples are timer or cryptology add-ons.

Each add-on requires a set of input parameters. Either these parameters are given as input variables or they are defined directly in the model. In the first approach, the input parameters are calculated dynamically. The second approach gives the parameters as static model information. For example, the SQL Adapter can have an **sql** input parameter holding the SQL statement, or the SQL statement can be given as an action script of the <<SQLAdapter>> action. Generally, each adapter has one input parameter that defines the message sent to the target system. If this parameter is defined statically in the model, it is put into the action script field. An example of this would be the SQL template (SQL Adapter), system commands (System Adapter), post message (URL Adapter), email message (SMTP Adapter) and so on. All input parameters that define the connection to the target system are given via information in the deployment diagram. The link between the adapter and the component diagram is given by the **alias** tagged value. All input parameters that do not fall into the above categories can be statically given as tagged values. For example, the SMTP adapter has tagged values to define the **TO**, **CC** and **BCC** header fields.

	Connection parameter to the target system	**Message or content that will be sent to the target system**	**Additional parameter and link between Component Diagram and Adapter**
Static	Component diagram on the service as tagged values (e.g. port, protocol, charset, user, path, method, alias)	Action script of an adapter defined in an action	Tagged Values in the action (e.g. alias, call)
Dynamic		Input out of data item or file controlled by arguments (e.g. key, value, server, port, domainName, sender, recipients, content, headerParameters, command, inputCount, outputCount, input, output, requestName, requestMessage, threaded, timeout, firstOccurrence, repeatInterval, occurrences, requestMessage, url, method, response)	