

# 1 SWS-E

## 1.1 Overview of the Semantic Web Services Executor

The Semantic Web Service Executor (SWS-E) is responsible for executing the Semantic Web Service descriptions based on WSMO. This component handles the interface descriptions of the WSMO Web Service descriptions. Input data is received from the client via the SAM component. This input data is used to evaluate the transition rules within the choreography descriptions. When a rule (or more than one) evaluates to true, an invocation to the underlying service occurs. The result of this invocation is then sent back via SAM. The instance data fed in to the executor is of course at the semantic level, that is, ontology-based. This instance data is also transformed to the equivalent XML data. In a similar way, the XML data which is received after invocation is lifted back to the equivalent ontological level. The executor also makes use of the QoS Monitor during the execution of the services.

## 1.2 Getting started

There are two basic things which are needed by the executor, namely, the configuration file and the required XSLT files. The configuration file is nothing but a parameter-value text file which can be easily parsed with the `java.util.Properties` utility class. This file should be placed in the working directory of the executor. The executor contains also the QoS Monitor. This component has its own configuration file and is further described in Section 2.

### 1.2.1 Installing the SWS-E

The SWS-E is packed up within a jar file. It has also dependencies on other jar files which should be kept in the same working directory as for the executor. In addition to that, the configuration file named `swse_config.txt` should also be in the same working directory. The dependent jar files are the following:

<code>axis.jar</code>	The axis library
<code>choreography.jar</code>	WSMO choreography library
<code>commons-discovery.jar</code>	Needed by Axis
<code>commons-logging.jar</code>	Needed by Axis
<code>hsqldb.jar</code>	HSQLDB library (note that if mysql is used, then it should be <code>mysql.jar</code> )
<code>log4j.jar</code>	Needed by Axis
<code>saaj.jar</code>	Needed for SOAP invocations
<code>wsdl4j.jar</code>	Java API for WSDL
<code>wsmllparser.jar</code>	Latest WSMML parser
<code>wsmo4j.jar</code>	Latest WSMO4J API
<code>wsmx-common.jar</code>	Library which defines some components which are WSMX dependent but used also by the executor

qos.jar	QoS Monitor
sfs-adapter.jar	Adapters of the SFS Testbed
egov-adapter.jar	Adapters of the eGove Testbed
infrawebs_if.jar	Infrawebs Integrated Framework API
if.jar	Core API of the Integrated Framework
xercesImpl.jar	Xerces XML Parser
jaxrpc.jar	Required for RPC invocations

### 1.2.2 Configuration

The configuration file contains the following main parameters:

xslts_dir	The directory where the required XSLT files are
monitor_config	Configuration file of the QoS Monitor

The first parameter defines the directory where the required XSLT files for the adapters can be found. The second parameter defines the filename or full path of the configuration file for the QoS-Monitor. If only a filename is given, it is then assumed to be in the same working directory of the executor. For further details about the configuration of the QoS-Monitor, please refer to Section 2.

## 2 QoS MONITOR

### 2.1 Overview of the QoS Monitor

This component is responsible for collecting QoS metric data during the execution of the semantic web services and calculate the respective QoS values. These values are then stored in the non-functional properties of the web service and also as instances in the respective QoS ontologies. The QoS Monitor is tightly coupled with the Executor but it can be shipped as a separate jar file. The respective execution engine must then make sure to make the appropriate calls to the monitor in order for it to correctly perform the metric calculations.

### 2.2 Getting started

As stated, the QoS Monitor is shipped as a separate jar file and in the case of our project, it is used by the SWS-E. Along the monitor is a configuration file. As described in Section 1.2.2, the configuration settings of the SWS-E define from where the configuration file of the QoS Monitor is fetched. If no configuration file is found, the default settings are used. These are described in section 2.2.2.

#### 2.2.1 Installing the QoS Monitor

There is not really an installation procedure apart from setting up a database. The qos.jar file contains the necessary classes to run the monitor. The

configuration file should be (preferably) placed in the same working directory. With respect to the database used and how to install it, the setup of the database on the type of database used. We recommend either HSQLDB or MySQL database systems. The former allows to have an in-memory database and thus a faster access in the case where not so many web services are expected to be executed. On the other hand, MySQL can be used when a large amount of web services are expected to be executed. It will also allow to keep a backup of the monitoring data. The default configuration settings are based on MySQL. In both cases, however, the user is required to setup a database with a login and password. Such settings are then defined in the configuration file.

### 2.2.2 Configuration

In a similar way to the executor, the QoS Monitor is started via a configuration file. As already described, the QoS Monitor makes use of a database to keep the metric data. This database is preferred to be in-memory for performance purposes. However, it can also be a normal database which can be remotely accessed. To this extend, the following parameters in the configuration file must be specified:

JDBC_DRIVER	The name of the jdbc driver used by the database
DB_URL	The URL of the database
LOGIN	The required login to access the database
PASSWORD	The required password to access the database

Further to these settings is the `UPDATE_INTERVAL` which specifies the frequency at which the metric values in the cache have to be updated. If the configuration file is not found, the following default values are used:

JDBC_DRIVER	<code>com.mysql.jdbc.Driver</code>
DB_URL	<code>jdbc:mysql://localhost/qos</code>
LOGIN	<code>infrawebs.qos</code>
PASSWORD	<code>infrawebs</code>
UPDATE_INTERVAL	<code>5000</code>

For convenience, these configuration parameters are specified in an enumeration class calls `QoSMonitorConfig`.

### 2.3 Usage

Once the monitor is started, the usage is simple. Monitor Events have to be created and each one must define the web service identifier, the execution identifier and the status (defined in the `ExecutionEvents` enumeration). The metric value is automatically updated once the status `END`, `SYSTEM_FAULT` or `EXECUTION_FAULT` have been received. These monitor events are fed to the monitor through the `monitor()` method.