Monitoring Software AG webMethods Broker

This topic explains how to configure monitoring for events that are transmitted through Software AG webMethods Broker. This monitoring requires admin client group privileges. Sections include:

Adding Required Jar Files to the SOAtest Classpath

The following jar files need to be added to the SOAtest classpath:

- wmbrokerclient.jar
- g11nutils.jar

The jar files can be found under [webmethods install dir]/Broker/lib/. For more details, please refer to webMethods Broker Client Java API Programmer's Guide> Getting Started> Using the webMethods Broker Java API.

To add these jar files to SOAtest's classpath, complete the following:

1. Choose **Parasoft> Preferences**.
2. Open the **Parasoft> System Properties** page.
3. Click the **Add JARS** button and choose and select the necessary JAR files to be added.

Configuring Event Monitor

1. Double-click the Event Monitor tool to open up the tool configuration panel.
2. Click the Event Source tab and configure the following settings.
3. Choose **Software AG webMethods Broker** from the platform drop-down menu. Adjust the configuration field values as needed. The fields are the same as those used in the webMethods tool, and are described in webMethods**.**

Using Filters and Wildcards

The Event Monitor uses BrokerAdminClient to monitor events. By default, it subscribes to the "admin" client group. Thus, if you wish to filter events based on their content, you should change the client group value to one that allows for subscription to regular event types (instead of Trace). This is because when using `Broker::Trace::* events`, the filter string will be applied to the fields in the trace BrokerEvents—not the original events that they represent.

When completing the Event Type field, remember that wildcards are not allowed for Broker::Trace or Broker::Activity event types according to WebMethods Broker Client Java API Programmer's Guide. If you wish to monitor a set of event types, specify a client group name that allows subscription access to the desired event types (possibly other than the default "admin" client group setting), then provide event types with wildcards. For example, you can use `Sample::*`

You may also use `${data bank column name}` variables in your string. SOAtest will replace that with the data bank value so filter strings can have dynamic values based on the output of other tests.

For more details on filters and using wildcards in event type names, please refer to the WebMethods Broker Client Java API Programmer's Guide.

Configuring the Event Monitor Options

Clear the event viewer before each event monitor run	Enable this option to automatically clear the Event Monitor event view (both text and graphical) whenever Event Monitor starts monitoring.
Include test execution events in the XML event output to chained tools	Enable this option to show only the monitored messages and events in the Event Viewer tab and XML output display. This option also indicates when each test started and completed. Enabling this option is helpful if you have multiple tests in the test suite and you want to better identify the events and correlate them to your test executions.

Wrap monitored messages with CDATA to ensure well-formedness of the XML event output	Enable this option if you do not expect the monitored events' message content to be well-formed XML. Disabling this option will make the messages inside the events accessible via XPaths, allowing the message contents to be extracted by XML Transformer or validated with XML Assertor tools. Enable this option if the message contents are not XML. This ensures that the XML output of the Event Monitor tool (i.e., the XML Event Output for chaining tools to the Event Monitor, not what is shown under the Event Viewer) is well-formed XML by escaping all the message contents. This will make the content of these messages inaccessible by XPath since the message technically becomes just string content for the parent element. The Diff tool's XML mode supports string content that is XML. As a result, the Diff tool will still be able to diff the messages as XML, including the ability to use XPaths for ignoring values, even if this option is disabled.
Maximum time to wait for the monitor to start (milliseconds)	Specify the maximum length of time the Event Monitor should wait to finish connecting to the event source before SOAtest runs the other tests in the suite. This enables SOAtest to capture events for those tests and prevents SOAtest from excessively blocking the execution of the other tests if the Event Monitor is having trouble connecting to its event source. Increase the value if connecting to the event source takes more time than the default. The default is `3000`.
Maximum monitor execution duration (milliseconds)	Specify the point at which the test should timeout if, for example, another test in the test suite hangs or if no other tests are being run (e.g., if you execute the Event Monitor test apart from the test suite, then use a custom application to send messages to system).
Event polling delay after each test finishes execution (milliseconds)	This field is not applicable to Software AG webMethods Broker.

Supported Event Types

With admin client group privileges, the Event Monitor tool can subscribe to the following event types:

- Broker::Ping
- Adapter::ack
- Adapter::adapter
- Adapter::error
- Adapter::errorNotify
- Adapter::refresh
- Broker::Trace::Publish
- Broker::Trace::Enqueue
- Broker::Trace::Drop
- Broker::Trace::Receive
- Broker::Trace::PublishRemote
- Broker::Trace::EnqueueRemote
- Broker::Trace::ReceiveRemote
- Broker::Activity::TerritoryChange
- Broker::Activity::ClientChange,
- Broker::Activity::ClientGroupChange
- Broker::Activity::EventTypeChange
- Broker::Trace::Insert
- Broker::Trace::Delete
- Broker::Trace::Peek
- Broker::Trace::DropRemote
- Broker::Trace::Modify
- Broker::Activity::ClientSubscriptionChange
- Broker::Activity::RemoteSubscriptionChange

Note that you can also configure the webMethods tool (described in webMethods) to subscribe to the desired event type—the only difference is that with the webMethods tool you need to provide the specific event type name.