Adding and Scheduling Tasks

A task runs a piece of code triggered by a timer, allowing you to run scheduled jobs at specified intervals. A task can be scheduled in the following ways:

1. Using `count` and `interval` attributes to run the task a specified number of times at a given interval.
2. Giving the scheduled time as a cron style entry.
3. Making the task run only once after the ESB starts by using the `once` attribute.

Task handling in a clustered environment

In a clustered environment, tasks are distributed among server nodes according to the **round-robin** method, by default. If required, you can change this default task handling behavior so that tasks are distributed randomly, or according to a specific rule. This is a server-level setting that is configured in the `tasks-config.xml` file.

- See Configuring the Task Scheduling Component for instructions on configuring the task handling behavior at server-level.
- You can also configure the task handling behavior at task-level, by specifying the Pinned Servers for a task. Note that this setting overrides the server-level configuration.

Also, note that a scheduled task will only run on one of the nodes (at a given time) in a clustered environment. The task will failover to another node, only if the first node fails.

Having deployed a task implementation to the ESB runtime (see Writing Tasks), you can use the ESB Management Console to add a task to the “Tasks” list and schedule various instances of the task. You can use either UI configuration or XML configuration to add and schedule tasks, as explained below.

UI Configuration

Follow the instructions below to add and schedule a task in ESB Management Console.

1. Sign in. Enter your user name and password to log in to the ESB Management Console.
2. Click **Main** in the left menu to access the Manage menu.
3. In the Manage menu, click **Scheduled Tasks** under Service Bus.

![Scheduled Tasks](image)

4. The Scheduled Tasks page appears, where you can add, **edit**, and **delete** tasks.

![Scheduled Tasks](image)

5. Click **Add Task**.

![Scheduled Tasks](image)

6. The New Scheduled Task page appears. Enter the required details into the fields.
   a. **Task Name** - Name of a scheduled task.
   b. **Task Group** - The group name to grouping tasks. The group name `synapse.simple.quartz` belongs to ESB - Synapse. All available groups are displayed as a drop-down menu. If there are tasks belong to some other domains, for example WSO2 Mashups tasks, then those will be shown here as a separate group names.
   c. **Task Implementation** - The implementation class of the task. To use the default task implementation that is available with the ESB (and therefore can be used without downloading any third-party libraries or custom JARs), specify `org.apache.synapse.startup.tasks.MessageInjector`. This class simply injects a specified message into the Synapse environment at ESB startup. For more information on writing custom task implementations, see [Writing Tasks](#).
   d. **Trigger Type** - Trigger type for the task. This can be selected as either "Simple" or "Cron."
      - **Simple Trigger** - Defined by specifying a **count** and an **interval**, implying that the task will run a **count** number of times at specified intervals.
      - **Count** - The number of times the task will be executed.
      - **Interval** - The interval between consecutive executions of a task.
      - **Cron Trigger** - Defined using a cron expression.
   e. **Pinned Servers** - The list of ESB server nodes that will run the task. You can specify the IP addresses of the required nodes.
This setting can be used if you want the task to run on a selected set of nodes in an ESB cluster. Note that the task will only run on one of the nodes at a time. It will failover to another node, only if the first node fails.

As explained above, pinned servers will override the default task handling behavior defined at server-level (for this particular task). However, if rule-based task handling is specified at server-level, you need to ensure that the same server nodes you specify as pinned servers for the task are also specified for the task handling rule at server-level.

7. Click **Load Task Properties** to see the instance properties of the task implementation.

8. Use the instance properties fields as follows:
   - **Property Name** - The unique name of the task property.
   - **Property Type** - The type of property, either Literal or XML.
   - **Property Value** - The value of the property.
   - **Action** - Allows you to delete a property.

   For more information on setting the properties for the default task implementation, see Examples and Injecting the message to a named sequence or proxy service below.

The org.apache.synapse.startup.tasks.MessageInjector implementation takes the following properties:
   - **format** - defines the format of the message similar to Address Endpoint formats: soap11, soap12, pox, get
   - **message** - you can provide an XML or literal value depending on message format.
When you add a scheduled task, it is mandatory to provide a value for the message property. Therefore, even if you do not want to send a message body, you have to provide an empty payload as the value to avoid an exception being thrown.

- **soapAction** - specify the SOAP Action to use when sending the message to the endpoint.
- **to** - specify the endpoint address.
- **injectTo** - specify whether to inject a message to a proxy service or sequence. This field takes values 'sequence' or 'proxy' and 'main' to inject to main sequence.
- **proxyName** - if injectTo contains 'proxy' then the name of the proxy to inject the message to is specified here.
- **sequenceName** - if injectTo contains 'sequence' then the name of the sequence to inject the message to is specified here.

9. Click **Schedule** to apply the settings.

**XML Configuration**

Follow the instructions below to add and schedule tasks using XML configuration.

1. Sign in. Enter your user name and password to log on to the ESB Management Console.
2. Click on "Main" in the left menu to access the "Manage" menu.
3. In the "Manage" menu, click on "Source View" under "Service Bus."
4. In the source view, add the task configuration based on your requirement.

The syntax of the task configuration is as follows:

```xml
<task class="string" name="string" [group="string"] [pinnedServers="(serverName)+"]>
  <property name="string" value="String"/>
  <property name="string"><somexml>config</somexml></property>
  <trigger ([(count="int"]? interval="int"] | [cron="string"] | [once=(true | false)])>
</task>
```

Examples

Following are examples of configuring some common use cases. For an example of configuring a task with a simple trigger, see Sample 300: Introduction to Tasks with a Simple Trigger. To see a complete example of writing a new task and configuring it in the UI, see Writing Tasks Sample.

To run every 5 seconds continuously:

```xml
<task name="CheckPrice" class="org.wso2.esb.tutorial.tasks.PlaceStockOrderTask">
  <trigger interval="5"/>
</task>
```

To run every 5 seconds for 10 times:

```xml
<task name="CheckPrice" class="org.wso2.esb.tutorial.tasks.PlaceStockOrderTask">
  <trigger interval="5" count="10"/>
</task>
```

You can also give cron-style values. To run daily at 1:30 AM:

```xml
<task name="CheckPrice" class="org.wso2.esb.tutorial.tasks.PlaceStockOrderTask">
  <trigger cron="0 30 1 * * ?"/>
</task>
```

To run only once after ESB starts:
Injecting the message to a named sequence or proxy service

By default, the message is sent to the Main sequence. To send it to a different sequence or to a proxy service, set the `injectTo` property to `sequence` or `proxy`, and then add the `sequenceName` or `proxyName` property to specify the name of the sequence or proxy service to use. For example:

**Injecting to a sequence other than Main**

```xml
<task name="SampleInjectToSequenceTask" class="org.apache.synapse.startup.tasks.MessageInjector" group="synapse.simple.quartz">
  <trigger count="2" interval="5"/>

  <property xmlns:task="http://www.wso2.org/products/wso2commons/tasks" name="injectTo" value="sequence"/>

  <property xmlns:task="http://www.wso2.org/products/wso2commons/tasks" name="message">
    <m0:getQuote xmlns:m0="http://services.samples">
      <m0:request>
        <m0:symbol>IBM</m0:symbol>
      </m0:request>
    </m0:getQuote>
  </property>

  <property xmlns:task="http://www.wso2.org/products/wso2commons/tasks" name="sequenceName" value="SampleSequence"/>

</task>
```

**Injecting to a proxy service**

```xml
<task name="SampleInjectToProxyTask" class="org.apache.synapse.startup.tasks.MessageInjector" group="synapse.simple.quartz">
  <trigger count="2" interval="5"/>

  <property xmlns:task="http://www.wso2.org/products/wso2commons/tasks" name="message">
    <m0:getQuote xmlns:m0="http://services.samples">
      <m0:request>
        <m0:symbol>IBM</m0:symbol>
      </m0:request>
    </m0:getQuote>
  </property>

</task>
```
Injecting messages to RESTful Endpoints

In order to use the Message Injector to inject a message to a RESTful endpoint, we can specify the injector with the required payload and inject the message to sequence or proxy service as defined above. The sample below shows a RESTful message injection through a ProxyService.

Injecting to a sequence other than Main

```xml
<definitions xmlns="http://ws.apache.org/ns/synapse">
  <registry provider="org.wso2.carbon.mediation.registry.WSO2Registry">
    <parameter name="cachableDuration">15000</parameter>
  </registry>
  <proxy name="SampleProxy" transports="https http" startOnLoad="true" trace="disable">
    <description/>
    <target>
      <inSequence>
        <property name="uri.var.city" expression="//request/location/city"/>
        <property name="uri.var.cc" expression="//request/location/country"/>
        <log>
          <property name="Which city?" expression="get-property('uri.var.city')"/>
          <property name="Which country?" expression="get-property('uri.var.cc')"/>
        </log>
        <send>
          <endpoint name="EP">
            <http method="get">
              <uri-template>http://api.openweathermap.org/data/2.5/weather?q={uri.var.city},{uri.var.cc}</uri-template>
            </http>
          </endpoint>
        </send>
      </inSequence>
    </target>
  </proxy>
</definitions>
```
<log level="full"/>
<drop/>
</outSequence>
</target>
</proxy>

<task name="SampleInjectToProxyTask"
   class="org.apache.synapse.startup.tasks.MessageInjector"
   group="synapse.simple.quartz">
   <trigger count="2" interval="5"/>
   <property xmlns:task="http://www.wso2.org/products/wso2commons/tasks"
     name="injectTo"
     value="proxy"/>
   <property xmlns:task="http://www.wso2.org/products/wso2commons/tasks" name="message">
     <request xmlns="">
       <location>
         <city>London</city>
         <country>UK</country>
       </location>
     </request>
   </property>
   <property xmlns:task="http://www.wso2.org/products/wso2commons/tasks"
     name="proxyName"
     value="SampleProxy"/>

</task>
</definitions>