Lab: Creating a Custom Task

Training Objective

Learn how to use a specific task-handling requirement by writing your own task-handling implementation.

High Level Steps

- Create the Maven projects.
- Create the Task implementation.
- Deploy the Task in the Micro Integrator.
- Test the Task.

Detailed Instructions

Customizing Task Scheduling

When you create a task using the default task implementation, the task can inject messages to a proxy service, or to a sequence. If you have a specific task-handling requirement, you can write your own task-handling implementation by creating a custom Java Class that implements the `org.apache.synapse.startup.Task` interface.

For example, the below sections demonstrate how you can create and schedule a task to receive stock quotes by invoking a back-end service, which exposes stock quotes. The scheduled task will read stock order information from a text file, and print the stock quotes.

Creating the custom Task implementation

Follow the steps below to create the implementation of the custom Task.

Creating the Maven Project

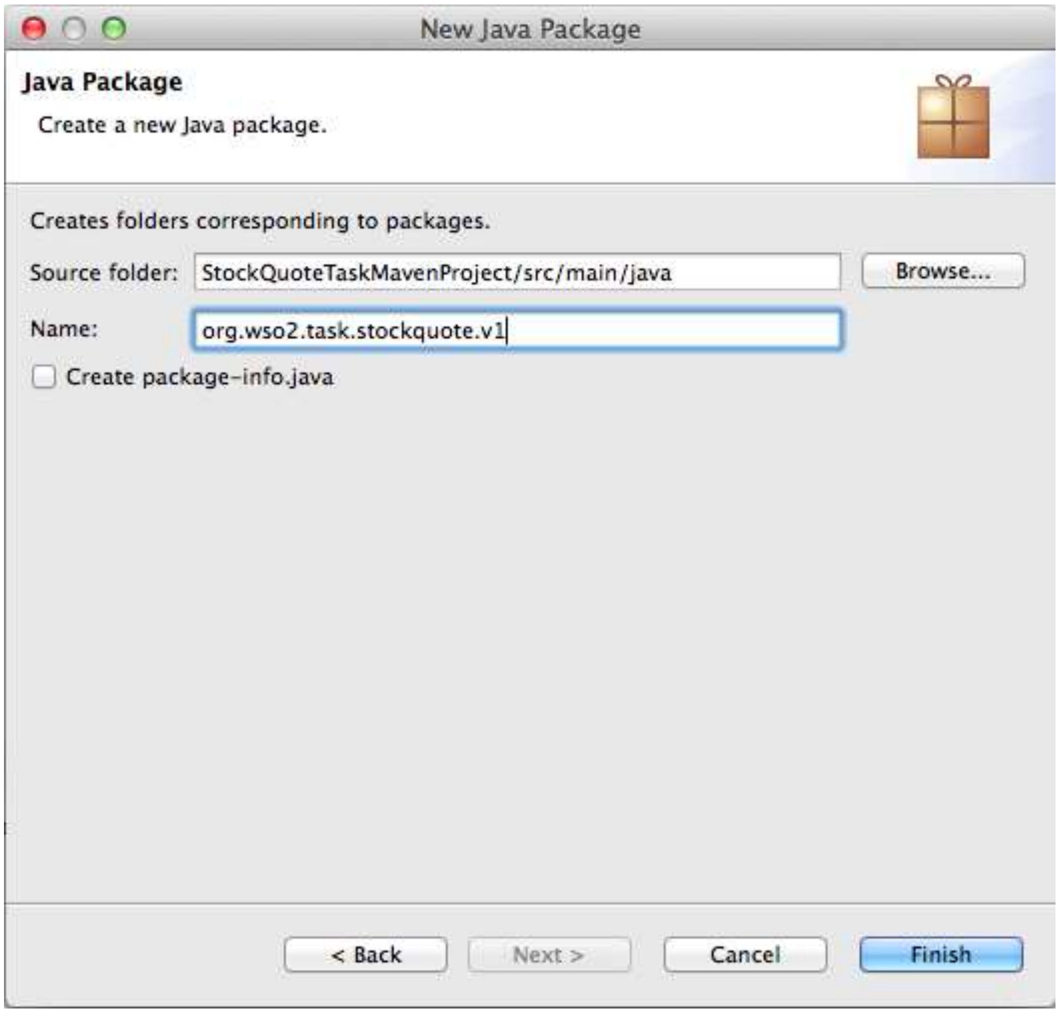
Create a Maven Project using the following information.

Tip

You can skip step 5 since you do not need to add external JAR files in this example. - **Group Id** : `org.wso2.task` - **Artifact Id** : `StockQuoteTaskMavenProject`

Creating the Java Package

Create a Java Package inside the Maven Project using the following name: `org.wso2.task.stockquote.v1`



Creates folders corresponding to packages.

Source folder: StockQuoteTaskMavenProject/src/main/java

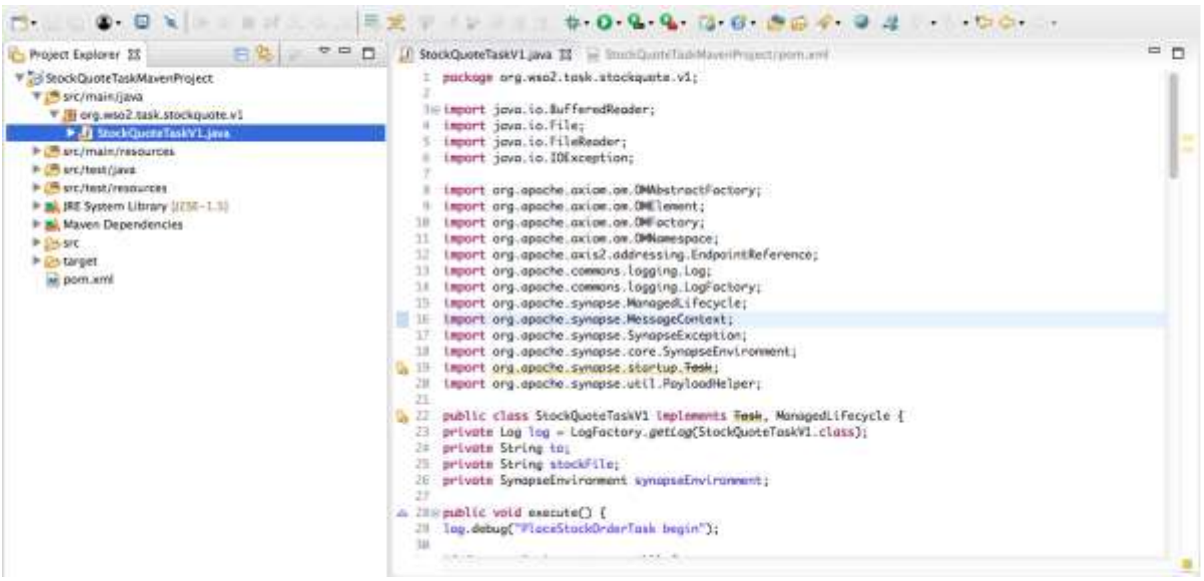
Name: org.wso2.task.stockquote.v1

☐ Create package-info.java

Creating the Java Class

1. Create a Java Class inside the Maven Project using the following name: `StockQuoteTaskV1`
2. In the **Project Explorer**, double-click on the **StockQuoteTaskV1.java** file and replace its source with the below content.

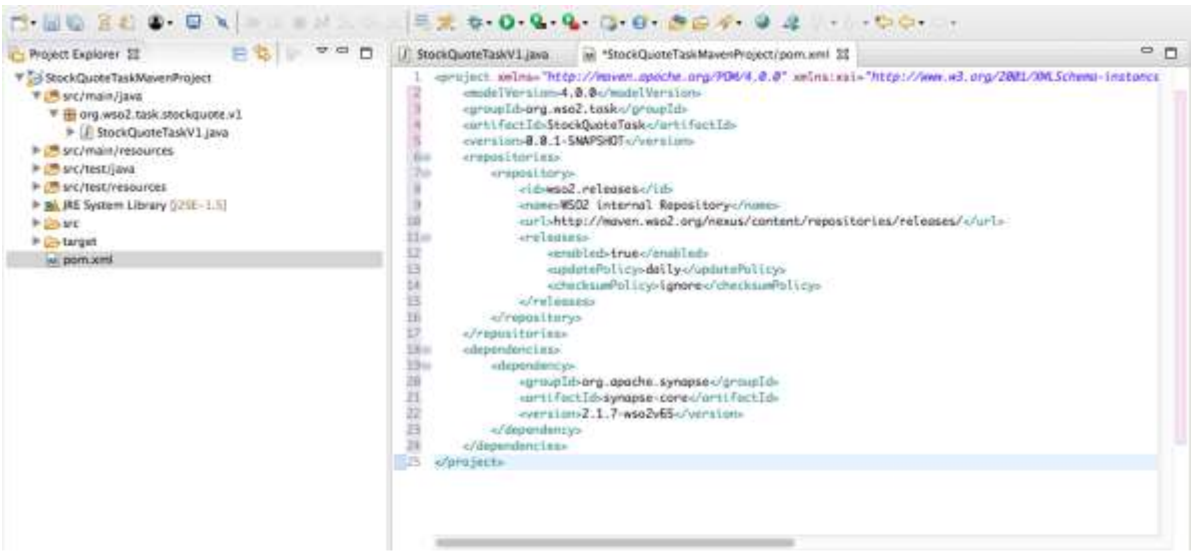
```
package org.wso2.task.stockquote.v1;

import java.io.BufferedReader;
import java.io.File;
import java.io.FileReader;
import java.io.IOException;

    import org.apache.axiom.om.OMAbstractFactory;
    import org.apache.axiom.om.OMElement;
    import org.apache.axiom.om.OMFactory;
    import org.apache.axiom.om.OMNamespace;
    import org.apache.axis2.addressing.EndpointReference;
    import org.apache.commons.logging.Log;
    import org.apache.commons.logging.LogFactory;
    import org.apache.synapse.ManagedLifecycle;
    import org.apache.synapse.MessageContext;
    import org.apache.synapse.SynapseException;
    import org.apache.synapse.core.SynapseEnvironment;
    import org.apache.synapse.startup.Task;
    import org.apache.synapse.util.PayloadHelper;
    public class StockQuoteTaskV1 implements Task, ManagedLifecycle {
        private Log log = LogFactory.getLog(StockQuoteTaskV1.class);
        private String to;
        private String stockFile;
        private SynapseEnvironment synapseEnvironment;

        public void execute() {
            log.debug("PlaceStockOrderTask begin");

            if (synapseEnvironment == null) {
                log.error("Synapse Environment not set");
                return;
            }

            if (to == null) {
                log.error("to not set");
                return;
            }

            File existFile = new File(stockFile);

            if (!existFile.exists()) {
                log.debug("waiting for stock file");
                return;
            }

            try {

                // file format IBM,100,120.50

                BufferedReader reader = new BufferedReader(new
FileReader(stockFile));
                String line = null;

                while ((line = reader.readLine()) != null) {
                    line = line.trim();

                    if (line == "") {
                        continue;
                    }

                    String[] split = line.split(",");
                    String symbol = split[0];
                    String quantity = split[1];
                    String price = split[2];
                    MessageContext mc =
synapseEnvironment.createMessageContext();
                    mc.setTo(new EndpointReference(to));
                    mc.setSoapAction("urn:placeOrder");
                    mc.setProperty("OUT_ONLY", "true");
                    OMElement placeOrderRequest =
createPlaceOrderRequest(symbol, quantity, price);
                    PayloadHelper.setXMLPayload(mc, placeOrderRequest);
                    synapseEnvironment.injectMessage(mc);
                    log.info("placed order symbol:" + symbol + " quantity:"
+ quantity + " price:" + price);
                }

                reader.close();
            } catch (IOException e) {
                throw new SynapseException("error reading file", e);
            }

            File renamefile = new File(stockFile);
            renamefile.renameTo(new File(stockFile + "." +
System.currentTimeMillis()));
            log.debug("PlaceStockOrderTask end");
        }

        public static OMElement createPlaceOrderRequest(String symbol,
String qty, String purchPrice) {
            OMFactory factory = OMAbstractFactory.getOMFactory();
            OMNamespace ns =
factory.createOMNamespace("http://services.samples/xsd", "m0");
            OMElement placeOrder = factory.createOMElement("placeOrder",
ns);
            OMElement order = factory.createOMElement("order", ns);
            OMElement price = factory.createOMElement("price", ns);
            OMElement quantity = factory.createOMElement("quantity", ns);
            OMElement symb = factory.createOMElement("symbol", ns);
            price.setText(purchPrice);
            quantity.setText(qty);
            symb.setText(symbol);
            order.addChild(price);
            order.addChild(quantity);
            order.addChild(symb);
            placeOrder.addChild(order);
            return placeOrder;
        }

        public void destroy() {}

        public void init(SynapseEnvironment synapseEnvironment) {
            this.synapseEnvironment = synapseEnvironment;
        }

        public SynapseEnvironment getSynapseEnvironment() {
            return synapseEnvironment;
        }

        public void setSynapseEnvironment(SynapseEnvironment
synapseEnvironment) {
            this.synapseEnvironment = synapseEnvironment;
        }

        public String getTo() {
            return to;
        }

        public void setTo(String to) {
            this.to = to;
        }

        public String getStockFile() {
            return stockFile;
        }

        public void setStockFile(String stockFile) {
            this.stockFile = stockFile;
        }
    }
```



```java
package org.wso2.task.stockquote.v1;

import java.io.BufferedReader;
import java.io.File;
import java.io.FileReader;
import java.io.IOException;

import org.apache.axiom.om.OMAbstractFactory;
import org.apache.axiom.om.OMElement;
import org.apache.axiom.om.OMFactory;
import org.apache.axiom.om.OMNamespace;
import org.apache.axis2.addressing.EndpointReference;
import org.apache.commons.logging.Log;
import org.apache.commons.logging.LogFactory;
import org.apache.synapse.ManagedLifecycle;
import org.apache.synapse.MessageContext;
import org.apache.synapse.SynapseException;
import org.apache.synapse.core.SynapseEnvironment;
import org.apache.synapse.startup.Task;
import org.apache.synapse.util.PayloadHelper;

public class StockQuoteTaskV1 implements Task, ManagedLifecycle {
private Log log = LogFactory.getLog(StockQuoteTaskV1.class);
private String to;
private String stockFile;
private SynapseEnvironment synapseEnvironment;

public void execute() {
log.debug("PlaceStockOrderTask begin");
```

3. In the **Project Explorer**, double-click on the **pom.xml** file and replace its source with the below content.

```xml
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
            <modelVersion>4.0.0</modelVersion>
            <groupId>org.wso2.task</groupId>
            <artifactId>StockQuoteTask</artifactId>
            <version>0.0.1-SNAPSHOT</version>
            <repositories>
                <repository>
                    <id>wso2.releases</id>
                    <name>WSO2 internal Repository</name>

<url>http://maven.wso2.org/nexus/content/repositories/releases/</url>
                    <releases>
                        <enabled>true</enabled>
                        <updatePolicy>daily</updatePolicy>
                        <checksumPolicy>ignore</checksumPolicy>
                    </releases>
                </repository>
            </repositories>
            <dependencies>
                <dependency>
                    <groupId>org.apache.synapse</groupId>
                    <artifactId>synapse-core</artifactId>
                    <version>2.1.7-wso2v65</version>
                </dependency>
            </dependencies>
</project>
```



```xml
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance
    <modelVersion>4.0.0</modelVersion>
    <groupId>org.wso2.task</groupId>
    <artifactId>StockQuoteTask</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <repositories>
        <repository>
            <id>wso2.releases</id>
            <name>WSO2 internal Repository</name>
            <url>http://maven.wso2.org/nexus/content/repositories/releases/</url>
            <releases>
                <enabled>true</enabled>
                <updatePolicy>daily</updatePolicy>
                <checksumPolicy>ignore</checksumPolicy>
            </releases>
        </repository>
    </repositories>
    <dependencies>
        <dependency>
            <groupId>org.apache.synapse</groupId>
            <artifactId>synapse-core</artifactId>
            <version>2.1.7-wso2v65</version>
        </dependency>
    </dependencies>
</project>
```

Writing the custom Task

Step 1: Writing the Task Class

You can create a custom task class, which implements the `org.apache.synapse.startup.Task` interface as follows. This interface has a single `execute()` method, which contains the code that will be executed according to the defined schedule.
The `execute()` method contains the following actions:

1. Check whether the file exists at the desired location.
2. If it does, then read the file line by line composing place order messages for each line in the text file.
3. Individual messages are then injected to the synapse environment with the given `To` endpoint reference.
4. Set each message as `OUT_ONLY` since it is not expected any response for messages.

In addition to the `execute()` method, it is also possible to make the class implement a `JavaBean` interface.
Also, add the following dependency to the POM file of the custom task project: `WSO2 Carbon - Utilities bundle` (symbolic name: `org.wso2.carbon.utils`)

This is a bean implementing two properties: To and StockFile. These are used to configure the task.

Implementing `ManagedLifecycle` for Initialization and Clean-up

Since a task implements `ManagedLifecyle` interface, the Micro Integrator will call the `init()` method at the initialization of a `Task` object and `destroy()` method when a `Task` object is destroyed:

```
public interface ManagedLifecycle {
public void init(SynapseEnvironment se);
public void destroy();
        }
```

The `PlaceStockOrderTask` stores the Synapse environment object reference in an instance variable for later use
with the `init()` method. The `SynapseEnvironment` is needed for injecting messages into the ESB.

Step 2: Customizing the Task

It is possible to pass values to a task at runtime using property elements. In this example, the location of the stock order file and its address was given using two properties within the `Task` object:

- **String type**
- **OMElement type**

Info

For **OMElement** type, it is possible to pass XML elements as values in the configuration file.

When creating a `Task` object, the ESB will initialize the properties with the given values in the configuration file.

```
public String getStockFile() {
return stockFile;
}
public void setStockFile(String stockFile) {
this.stockFile = stockFile;
}
```

For example, the following properties in the `Task` class are initialized with the given values within the property element of the task in the configuration.

```
<syn:property xmlns="http://ws.apache.org/ns/synapse"
name="stockFile"value="/home/upul/test/stock.txt"/>
```

For those properties given as XML elements, properties need to be defined within the `Task` class using the format given below. OMElement comes from Apache AXIOM, which is used by the Micro Integrator. AXIOM is an object model similar to DOM. To learn more about AXIOM, see the tutorial in the AXIOM user guide .

```
public void setMessage(OMElement elem) {
            message = elem;}
```

It can be initialized with an XML element as follows:

```
<property name="message">
    <m0:getQuote xmlns:m0="http://services.samples/xsd">
    <m0:request>
    <m0:symbol>IBM</m0:symbol>
    </m0:request>
    </m0:getQuote>
</property>
```

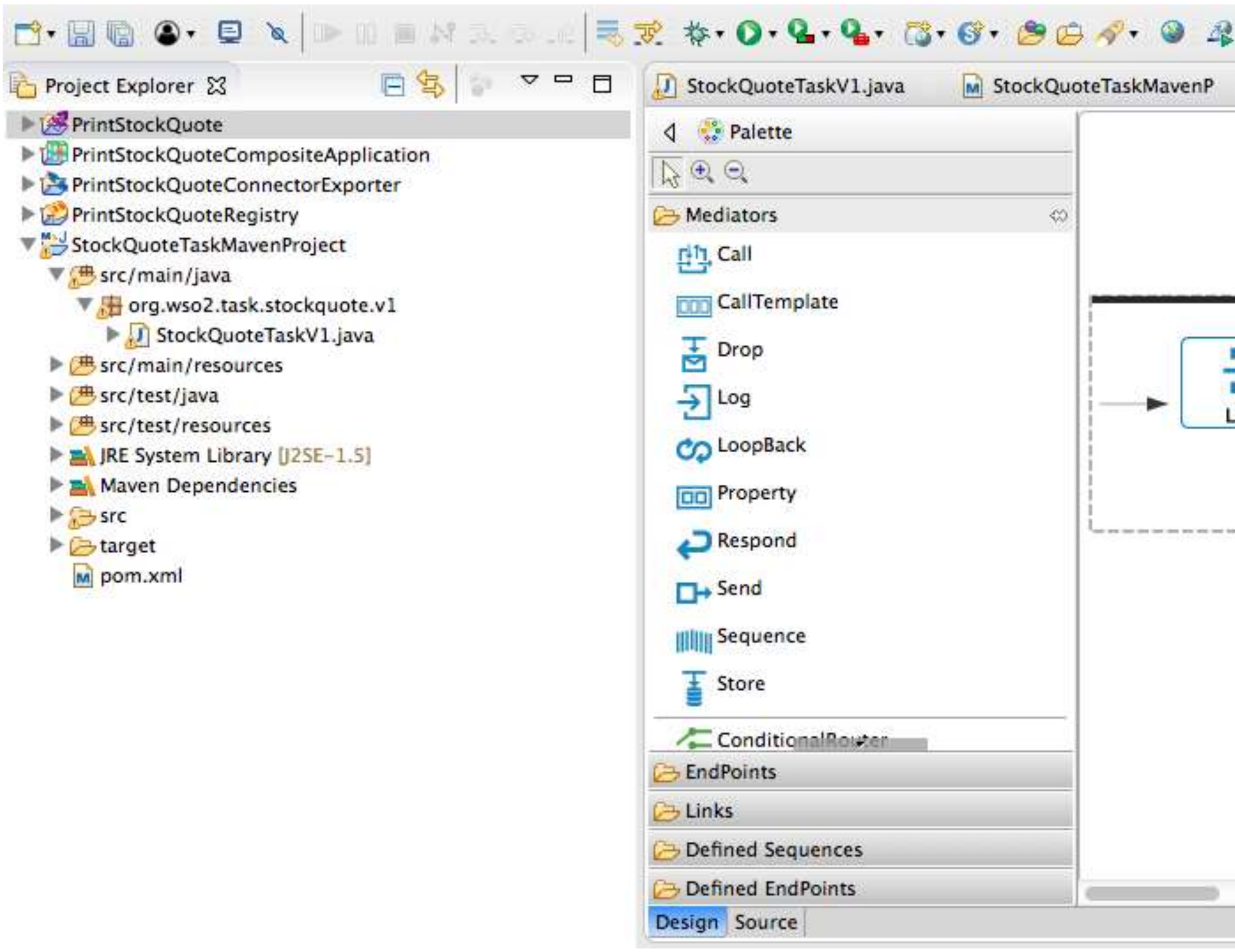
Deploying the custom Task implementation

Deploy the custom Task implementation.

Creating the Task

Follow the steps below to create the task and schedule it.

1. Create a ESB Config project named `PrintStockQuote`.
2. Create a Sequence using the following information named `PrintStockQuoteSequence`.
3. Add a **Log Mediator** and a **Drop Mediator** to the sequence and configure them.

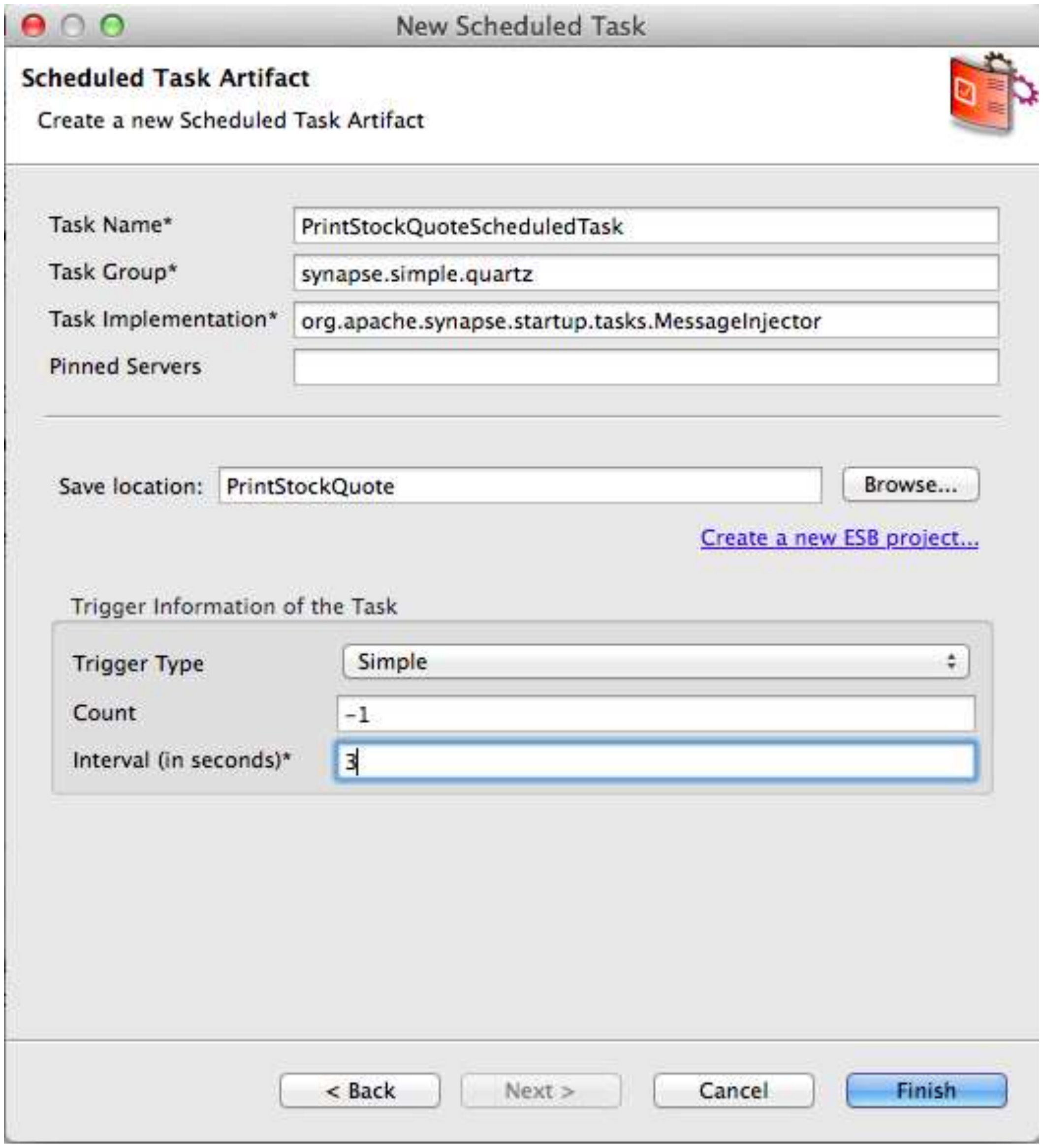


The below is the complete source configuration of the Sequence (i.e., the `PrintStockQuoteSequence.xml` file):

```
<?xml version="1.0" encoding="UTF-8"?>
<sequence name="PrintStockQuoteSequence" trace="disable"
xmlns="http://ws.apache.org/ns/synapse">
    <log level="custom"/>
    <drop/>
</sequence>
```

4. Create a Scheduled Task using the following information:

Task Property	Description
Task Name	`PrintStockQuoteScheduledTask`
Count	1
Interval (in seconds)	5


New Scheduled Task

Scheduled Task Artifact

Create a new Scheduled Task Artifact

Task Name* PrintStockQuoteScheduledTask

Task Group* synapse.simple.quartz

Task Implementation* org.apache.synapse.startup.tasks.MessageInjector

Pinned Servers

Save location: PrintStockQuote

Trigger Information of the Task

Trigger Type Simple

Count -1

Interval (in seconds)* 3

5. Defining the properties of the Task: In the **Project Explorer** , double-click the **Print StockQuoteScheduledTask.xml** file and replace its source with the below content.

```
<task class="org.apache.synapse.startup.tasks.MessageInjector"
group="synapse.simple.quartz" name="PrintStockQuoteScheduledTask"
xmlns="http://ws.apache.org/ns/synapse">
            <trigger count="1" interval="5" />
            <property name="to" value=
`http://localhost:9000/soap/SimpleStockQuoteService` />
            <property name="stockFile"
value="/Users/praneesha/Desktop/stockfile.txt"
xmlns:task="http://www.wso2.org/products/wso2commons/tasks" />
            <property name="synapseEnvironment" value=""
xmlns:task="http://www.wso2.org/products/wso2commons/tasks" />
</task>
```

The task properties will change according to the custom implementation. Therefore, you need to enter values for your custom properties. This sets the below properties.

Parameter Name	Value
to	`http://localhost:9000/soap/SimpleStockQuoteService`
`stockFile`	The directory path to the `stockfile.txt` file.

Parameter Name	Value
`synapseEnvironment`	Do not enter a value. This will be used during runtime.

Note

Currently, you cannot set properties of a custom task using the **Design View** due to a known issue, which will be fixed in future versions.

The below is the complete source configuration of the Task (i.e., the `PrintStockQuoteScheduledTask.xml` file).

```
<?xml version="1.0" encoding="UTF-8"?>
<task class="org.apache.synapse.startup.tasks.MessageInjector"
group="synapse.simple.quartz" name="PrintStockQuoteScheduledTask"
xmlns="http://ws.apache.org/ns/synapse">
        <trigger interval="3" />
        <property name="to"
value=`http://localhost:9000/soap/SimpleStockQuoteService`
xmlns:task="http://www.wso2.org/products/wso2commons/tasks" />
        <property name="stockFile"
value="/Users/praneesha/Desktop/stockfile.txt"
xmlns:task="http://www.wso2.org/products/wso2commons/tasks" />
</task>
```

Deploying the Task

Deploy the Task.

Testing the Custom Task

Starting the back-end service

Download the backend service from GitHub and run it.

Creating the text file

Create a text file named `stockfile.txt` with the following content and save it to a preferred location on your machine. This will include the information to be read by the scheduled task to pass to the backend service.

stockfile.txt

```
IBM,100,120.50
MSFT,200,70.25
SUN,400,60.758
```

Info

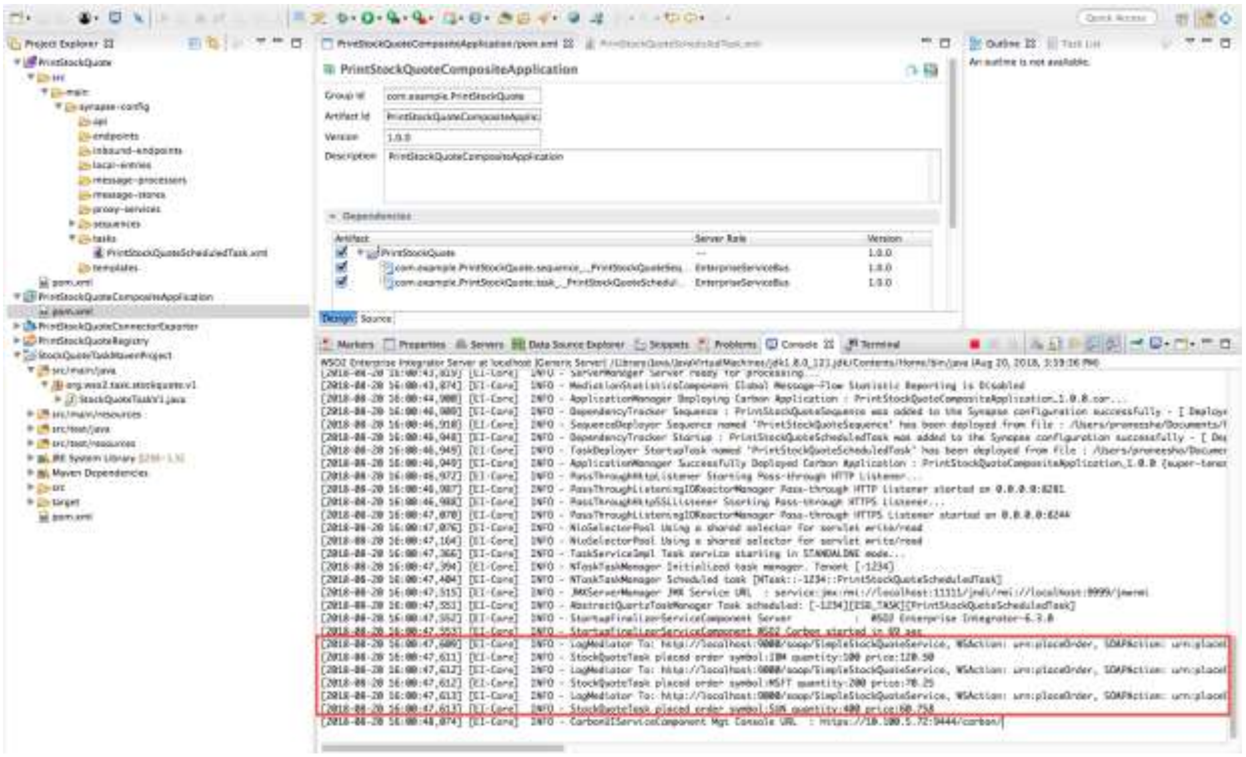
Each line in the text file contains details for a stock order: - `symbol` - `quantity` - `price`

A task that is scheduled using this custom implementation will read the text file, a line at a time, and create orders using the given values to be sent to the back-end service. The text file will then be tagged as processed to include a system time stamp. The task will be scheduled to run every 15 seconds.

Viewing the output

You will view the stock quotes sent by the backend service printed every 3 seconds by the scheduled task in the below format.

```
INFO - StockQuoteTask placed order symbol:IBM quantity:100 price:120.50
```



```
WSO2 Enterprise Integrator Server at localhost [Generic Server] /Library/Java/JavaVirtualMachines/jdk1.8.0_121.jdk/Contents/Home/bin/java (Aug 20, 2018, 3:59:36 PM)
[2018-08-20 16:00:43,819] [EI-Core]  INFO - ServerManager Server ready for processing...
[2018-08-20 16:00:43,874] [EI-Core]  INFO - MediationStatisticsComponent Global Message-Flow Statistic Reporting is Disabled
[2018-08-20 16:00:44,900] [EI-Core]  INFO - ApplicationManager Deploying Carbon Application : PrintStockQuoteCompositeApplication_1.0.0.car...
[2018-08-20 16:00:46,080] [EI-Core]  INFO - DependencyTracker Sequence : PrintStockQuoteSequence was added to the Synapse configuration successfully - [ Deploye
[2018-08-20 16:00:46,918] [EI-Core]  INFO - SequenceDeployer Sequence named 'PrintStockQuoteSequence' has been deployed from file : /Users/praneesha/Documents/
[2018-08-20 16:00:46,948] [EI-Core]  INFO - DependencyTracker Startup : PrintStockQuoteScheduledTask was added to the Synapse configuration successfully - [ Dep
[2018-08-20 16:00:46,949] [EI-Core]  INFO - TaskDeployer StartupTask named 'PrintStockQuoteScheduledTask' has been deployed from file : /Users/praneesha/Documen
[2018-08-20 16:00:46,949] [EI-Core]  INFO - ApplicationManager Successfully Deployed Carbon Application : PrintStockQuoteCompositeApplication_1.0.0 {super-tenan
[2018-08-20 16:00:46,972] [EI-Core]  INFO - PassThroughHttpListener Starting Pass-through HTTP Listener...
[2018-08-20 16:00:46,987] [EI-Core]  INFO - PassThroughListeningIOReactorManager Pass-through HTTP Listener started on 0.0.0.0:8281
[2018-08-20 16:00:46,988] [EI-Core]  INFO - PassThroughHttpSSLListener Starting Pass-through HTTPS Listener...
[2018-08-20 16:00:47,070] [EI-Core]  INFO - PassThroughListeningIOReactorManager Pass-through HTTPS Listener started on 0.0.0.0:8244
[2018-08-20 16:00:47,076] [EI-Core]  INFO - NioSelectorPool Using a shared selector for servlet write/read
[2018-08-20 16:00:47,164] [EI-Core]  INFO - NioSelectorPool Using a shared selector for servlet write/read
[2018-08-20 16:00:47,366] [EI-Core]  INFO - TaskServiceImpl Task service starting in STANDALONE mode...
[2018-08-20 16:00:47,394] [EI-Core]  INFO - NTaskTaskManager Initialized task manager. Tenant [-1234]
[2018-08-20 16:00:47,404] [EI-Core]  INFO - NTaskTaskManager Scheduled task [NTask::-1234::PrintStockQuoteScheduledTask]
[2018-08-20 16:00:47,515] [EI-Core]  INFO - JMXServerManager JMX Service URL  : service:jmx:rmi://localhost:11111/jndi/rmi://localhost:9999/jmxrmi
[2018-08-20 16:00:47,551] [EI-Core]  INFO - AbstractQuartzTaskManager Task scheduled: [-1234][ESB_TASK][PrintStockQuoteScheduledTask]
[2018-08-20 16:00:47,552] [EI-Core]  INFO - StartupFinalizerServiceComponent Server           :  WSO2 Enterprise Integrator-6.3.0
[2018-08-20 16:00:47,553] [EI-Core]  INFO - StartupFinalizerServiceComponent WSO2 Carbon started in 69 sec
[2018-08-20 16:00:47,600] [EI-Core]  INFO - LogMediator To: http://localhost:9000/soap/SimpleStockQuoteService, WSAction: urn:placeOrder, SOAPAction: urn:placeO
[2018-08-20 16:00:47,611] [EI-Core]  INFO - StockQuoteTask placed order symbol:IBM quantity:100 price:120.50
[2018-08-20 16:00:47,612] [EI-Core]  INFO - LogMediator To: http://localhost:9000/soap/SimpleStockQuoteService, WSAction: urn:placeOrder, SOAPAction: urn:placeO
[2018-08-20 16:00:47,612] [EI-Core]  INFO - StockQuoteTask placed order symbol:MSFT quantity:200 price:70.25
[2018-08-20 16:00:47,613] [EI-Core]  INFO - LogMediator To: http://localhost:9000/soap/SimpleStockQuoteService, WSAction: urn:placeOrder, SOAPAction: urn:placeO
[2018-08-20 16:00:47,613] [EI-Core]  INFO - StockQuoteTask placed order symbol:SUN quantity:400 price:60.758
[2018-08-20 16:00:48,074] [EI-Core]  INFO - CarbonUIServiceComponent Mgt Console URL  : https://10.100.5.72:9444/carbon/
```