



Import and Export of XML data

using the *featherlite* Integration Framework

Author: Robert von Burg
Type: 5 minute tutorial
Date: 2011-02-18
Version: 0.1.1

Abstract

The goal of 5 minute tutorial is to describe the steps which are necessary to configure an instance of *featherlite* to use the *featherlite* Integration Framework. As a result, the *featherlite* instance will be configured to import and export XML models from and to files, respectively.

Contents

1 Overview	1
2 Prerequisites	1
3 Configuration	2
3.1 IntegrationHandler	2
3.2 Connection	2
4 IOService registration	3
5 Testing	4
5.1 <i>featherlite</i> Startup	4
5.2 Use Case: Export Orders	4
5.3 Use Case: Import Orders	5

1 Overview

This short tutorial is meant to be completed within 5 minutes, and shows the steps which are necessary for two use cases in order to configure an instance to use the *featherlite* Integration Framework.

In the first use case, the *featherlite* receives a file containing an XML model from the outside world (see Figure 1).

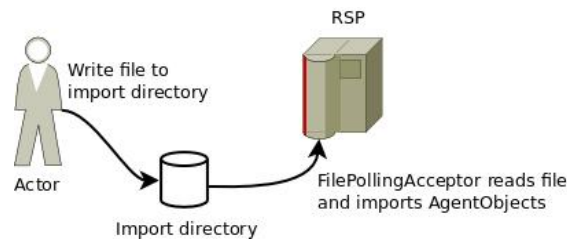


Figure 1: Inbound message

In the second use case, the *featherlite* exports `AgentObjects` to a XML model files in a predefined file system path, see Figure 1.

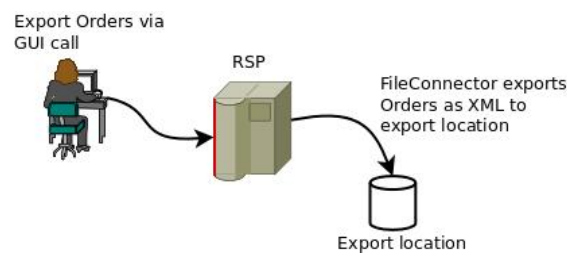


Figure 2: Outbound message

The `IOConverter` needed to convert the data is the `XmlModelConverter` from the package `com.rsp.communication.integration.converter.xml`.

The `IOAcceptor` reading the files from the file system is the `FilePollingAcceptor` and the `IOConnector` writing the files into a predefined file system path is the `FileConnector`. Both are from the `com.rsp.communication.integration.transport.file` package.

2 Prerequisites

To be able to complete this tutorial, you need to have completed the `GettingStarted` tutorial and have an instance of *featherlite* that you can run available. The *ExecutionServer* plug-in is recommended.

To test the configuration and view the results, you need the *ExecutionClient* rich client. This client has a set of actions and views which are needed to trigger the actions and

view the results.

3 Configuration

The following configurations steps refer to the instance of *featherlite* that will be started (e.g., the *ExecutionServer*). Thus, all mentioned configuration files are from that instance of *featherlite*.

By default, the *featherlite* Integration Framework is not started and is in *OFF* mode. To start the integration framework add the following configuration property to the `RSPConfig.xml` configuration file:

```
<entry key="rsp.integration.mode">on</entry>
```

3.1 IntegrationHandler

The `DefaultIntegrationHandler` must be registered as a *featherlite* Component. This is done by adding the configuration to the `RSPContainer.xml` file which is shown in the following listing:

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <container>
3   <component-implementation key='IntegrationHandler'
4     class='com.rsp.communication.integration.base.DefaultIntegrationHandler' />
5 </container>
```

Listing 1: Registration of `DefaultIntegrationHandler` in `RSPContainer.xml`

When adding the lifecycle elements, consider that the `IntegrationHandler` should be *initialized* and *started* as late as possible in the *featherlite* component lifecycle and in the same manner the handler should be *stopped* and *shutdown* as early as possible. Thus, the component shall be declared as late as possible in the `RSPContainer.xml` file.

3.2 Connection

The integration connection is defined in the `IntegrationConfig.xml` file as follows:

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <ic:IntegrationConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:ic="http://www.apixxo.com/
   rsp/IntegrationConfig"
3   xsi:schemaLocation="http://www.apixxo.com/rsp/IntegrationConfig IntegrationConfigSchema.xsd">
4
5   <ic:Connection id="XmlFileTransfer" name="XmlFileTransfer" inbound="FileInboundTransfer" outbound="
     FileOutboundTransfer">
6     <ic:IOConverter class="com.rsp.communication.integration.converter.xml.XmlModelConverter" />
7     <ic:IOConnector class="com.rsp.communication.integration.transport.file.FileConnector" />
8     <ic:IOAcceptor class="com.rsp.communication.integration.transport.file.FilePollingAcceptor" />
9     <ic:Parameters>
10      <ic:Parameter name="receiverKey" value="InboundAgentObjectIOService" />
11      <ic:Parameter name="outboundFilename" value="/tmp/OutboundTest.xml" />
12      <ic:Parameter name="incomingDirectoryPath" value="/tmp/inbound" />
```

```

13     <ic:Parameter name="archiveDirectoryPath" value="/tmp/archive" />
14     </ic:Parameters>
15 </ic:Connection>
16
17 </ic:IntegrationConfig>

```

Listing 2: Integration Connection Configuration in *IntegrationConfig.xml*

In this example, a single connection is defined. It has both an `IOAcceptor` for incoming messages and an `IOConnector` for outgoing messages. These two classes both share an instance of the configured `IOConverter`.

The `XmlModelConverter` needs the parameter *receiverKey* which defines which `IOService` is notified on incoming messages.

The `FileConnector` needs the parameter *outboundFilename* which defines where the exported messages are written to.

The `FilePollingAcceptor` needs two parameters. The parameter *incomingDirectoryPath* is an absolute path where new files are looked for. Parameter *archiveDirectoryPath* is an absolute path where files which have been read are moved to for archivation.

4 IOService registration

The `IOService` receiving the inbound messages has to be registered on the `IntegrationHandler`. This is normally done by using a `RSPPostInitializer`. This component is mostly a project specific *featherlite* Component that handles the initialization after *featherlite* has started.

If no `RSPPostInitializer` exists, a new one has be created. To register the `IOService`, we simply add a call to register the `InboundAgentObjectIOService` which takes care of incoming `IntegrationMessages` and adds any `AgengObjects` it encounters. Listing 4 shows the content of a simple `RSPPostInitializer`.

The `InboundAgentObjectIOService` registers itself on the `IntegrationHandler` with its class name, which explains the parameter value of the *receiverId* in the configuration file of the connection.

```

1 package com.rsp.execution.server.base;
2
3 import com.rsp.communication.integration.service.InboundAgentObjectIOService;
4 import com.rsp.core.base.ExecutionPostInitializer;
5
6 public class PostInitializer extends ExecutionPostInitializer {
7     private static final long serialVersionUID = 1L;
8     public boolean initializeRSP() {
9
10        // register test IOService
11        InboundAgentObjectIOService service = new InboundAgentObjectIOService();
12        service.register();
13
14        // call parent
15        return super.initializeRSP();
16    }
17 }

```

Listing 3: IOService registration in `RSPPostInitializer`

In order to be started, the `RSPPostInitializer` must also be registered as an *featherlite* Component. Listing 4 shows the `RSPContainer.xml` file which both the `IntegrationHandler` and the `RSPPostInitializer` are registered.

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <container>
3   <component implementation key='IntegrationHandler'
4     class='com.rsp.communication.integration.base.DefaultIntegrationHandler' />
5   <component implementation key='RSPPostInitializer' class='com.rsp.execution.server.base.PostInitializer' />
6 </container>

```

Listing 4: Registering of `PostInitializer` in `RSPContainer.xml`

5 Testing

Now that the project is configured for the *featherlite* Integration Framework, it is time to see the results. Before starting *featherlite* and testing if the configuration works, ensure that the directories specified in the `IntegrationConfig.xml` file exist and that they are readable and writable.

5.1 featherlite Startup

Start the *featherlite* server instance. In the log file, look for the following lines. If these lines exist, then the *featherlite* Integration Framework has loaded properly.

```

...
2010-02-02 15:31:18,018 INFO [RSP Startup] IntegrationConnectionPool initialize - Loading IntegrationHandler
connections from file IntegrationConfig.xml
...
2010-02-02 15:31:18,064 INFO [RSP Startup] IntegrationConnection reset - XmlFileTransfer: Loaded
IntegrationConnection: IOConverter: com.rsp.communication.integration.converter.xml.XmlModelConverter /
IOConnector: com.rsp.communication.integration.transport.file.FileConnector / IOAcceptor: com.rsp.
communication.integration.transport.file.FilePollingAcceptor
...
2010-02-02 15:31:18,619 INFO [RSP Startup] IntegrationConnection connect - Starting Integration connection
XmlFileTransfer...
...

```

5.2 Use Case: Export Orders

Start the `ExecutionClient` plug-in. As is shown in Figure 3 export the Orders.



State	Name	Type	Script	Date	
	TransportOrder	-	Template	dp01	2007-01-19T16:00:00.000+01:00
	e01	order1	TransportOrder	dp01	2008-01-19T16:00:00.000+01:00
	e02	order2	TransportOrder	dp02	2008-01-19T16:00:00.000+01:00
	e03	order3	TransportOrder	dp01	2008-01-19T16:00:00.000+01:00
	e04	order4	TransportOrder	dp02	2008-01-19T16:00:00.000+01:00
	e05	order5	TransportOrder	dp01	2008-01-19T16:00:00.000+01:00
	e06	order6	TransportOrder	dp02	2008-01-19T16:00:00.000+01:00

Figure 3: Use Case export Orders

The log from the server should be similar to the following:

```
...
2010-02-02 15:49:23,454 INFO [RMI TCP Connection(6)-127.0.0.1] DefaultServiceHandler doService - start: do
service: ExportOrderIOService with locale en
2010-02-02 15:49:23,468 INFO [RMI TCP Connection(6)-127.0.0.1] DefaultIntegrationHandler process - Adding
message: msgId: msg145167577308 / senderId: ExportOrderIOService / senderTransporterId:
FileOutboundTransfer / receiverId: ImportOrderIOService / receiverTransporterId: FileInboundTransfer / Number
of body elements: 21
2010-02-02 15:49:23,468 INFO [RMI TCP Connection(6)-127.0.0.1] DefaultServiceHandler doService - service
execution of ExportOrderIOService took 14
2010-02-02 15:49:23,471 INFO [IntegrationHandler] DefaultIntegrationHandler internalProcessOutbound - Trying
to send message on transporterId: FileOutboundTransfer
2010-02-02 15:49:23,531 INFO [XmlFileTransfer_Connection] ExportOrderIOService processDone - Yay, the
OrderMap should be exported to a file as is configured
2010-02-02 15:49:23,535 INFO [IntegrationHandler] Log4jIntegrationMsgArchivator archiveOutbound - Archive
Outbound Message with id msg145167577308 on 2010-02-02T15:49:23.534+01:00
```

If this is the case, the orders have been exported to the file specified in the *IntegrationConfig.xml* under the parameter with name *archiveDirectoryPath*.

5.3 Use Case: Import Orders

To test the import of `Orders`, create a file with the following content, representing an example order:

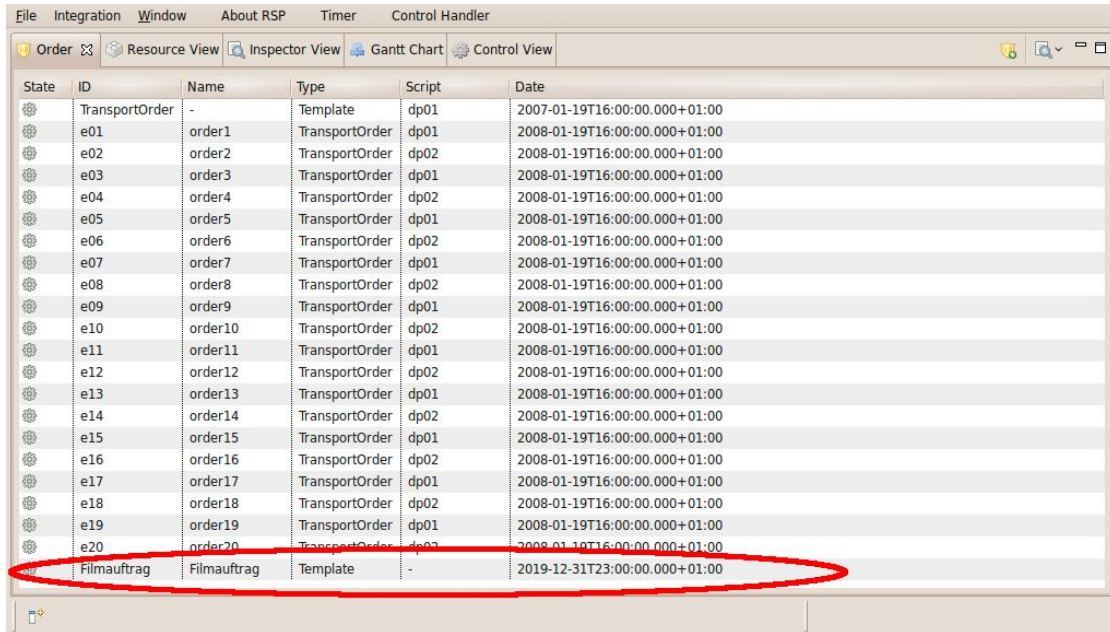
```
<model>
  <Order Id="Filmauftrag" Name="Filmauftrag" Type="Template" Date="2020-01-01T00:00:00.000+02:00">
    <Parameter Id="erzeuger" Name="Erzeuger" Type="String" Value="?" />
    <Parameter Id="startdatum" Name="Startdatum" Type="Time" Value="0" />
    <Parameter Id="enddatum" Name="Enddatum" Type="Time" Value="0" />
  </Order>
</model>
```

Save the file under the path specified in the *incomingDirectoryPath* parameter of the configuration *IntegrationConfig.xml*. For our example, the path is */tmp/inbound*.

As soon as the `FilePollingAcceptor` detects the file, the file is read and the order imported. The log should then contain entries like the following:

```
...
2010-02-02 15:59:51,454 INFO [inbound] InboundAgentObjectIOService notify - Received message:
  XmlIn145167577309 with 1 elements
2010-02-02 15:59:51,468 INFO [inbound] DefaultServiceHandler doService - start: do service:
  InboundAgentObjectIOService with locale en
2010-02-02 15:59:51,494 INFO [inbound] InboundAgentObjectIOService doService - Added order: Filmauftrag
2010-02-02 15:59:51,497 INFO [inbound] DefaultServiceHandler doService - service execution of
  InboundAgentObjectIOService took 29
2010-02-02 15:59:51,497 INFO [inbound] InboundAgentObjectIOService notify - Successfully imported
  AgentObjects from IntegrationMessage: XmlIn145167577309
2010-02-02 15:59:51,498 INFO [inbound] Log4jIntegrationMsgArchivator archiveInbound - Archive Inbound
  Message with id XmlIn145167577309 on 2010-02-02T15:59:51.498+01:00
```

In the *Order View* of the *ExecutionClient* client the new Order should be listed as is shown in Figure 4.



The screenshot shows a software application window with a menu bar (File, Integration, Window, About RSP, Timer, Control Handler) and a toolbar (Order, Resource View, Inspector View, Gantt Chart, Control View). Below the toolbar is a table with the following data:

State	ID	Name	Type	Script	Date
	TransportOrder	-	Template	dp01	2007-01-19T16:00:00.000+01:00
	e01	order1	TransportOrder	dp01	2008-01-19T16:00:00.000+01:00
	e02	order2	TransportOrder	dp02	2008-01-19T16:00:00.000+01:00
	e03	order3	TransportOrder	dp01	2008-01-19T16:00:00.000+01:00
	e04	order4	TransportOrder	dp02	2008-01-19T16:00:00.000+01:00
	e05	order5	TransportOrder	dp01	2008-01-19T16:00:00.000+01:00
	e06	order6	TransportOrder	dp02	2008-01-19T16:00:00.000+01:00
	e07	order7	TransportOrder	dp01	2008-01-19T16:00:00.000+01:00
	e08	order8	TransportOrder	dp02	2008-01-19T16:00:00.000+01:00
	e09	order9	TransportOrder	dp01	2008-01-19T16:00:00.000+01:00
	e10	order10	TransportOrder	dp02	2008-01-19T16:00:00.000+01:00
	e11	order11	TransportOrder	dp01	2008-01-19T16:00:00.000+01:00
	e12	order12	TransportOrder	dp02	2008-01-19T16:00:00.000+01:00
	e13	order13	TransportOrder	dp01	2008-01-19T16:00:00.000+01:00
	e14	order14	TransportOrder	dp02	2008-01-19T16:00:00.000+01:00
	e15	order15	TransportOrder	dp01	2008-01-19T16:00:00.000+01:00
	e16	order16	TransportOrder	dp02	2008-01-19T16:00:00.000+01:00
	e17	order17	TransportOrder	dp01	2008-01-19T16:00:00.000+01:00
	e18	order18	TransportOrder	dp02	2008-01-19T16:00:00.000+01:00
	e19	order19	TransportOrder	dp01	2008-01-19T16:00:00.000+01:00
	e20	order20	TransportOrder	dp02	2008-01-19T16:00:00.000+01:00
	Filmauftrag	Filmauftrag	Template	-	2019-12-31T23:00:00.000+01:00

Figure 4: Use Case import Orders

List of Figures

1	Inbound message	1
2	Outbound message	1
3	Use Case export Orders	4
4	Use Case import Orders	6