

Lab: Streaming Files using the VFS Transport

# Training Objective

Use the Virtual File System (VFS) transport to stream files.

# High Level Steps

* Create the mediation sequences and other artifacts using WSO2 Integration Studio.
* Deploy the artifacts in the Micro Integrator.
* Test the scenario.

# Detailed Instructions

# File processing

## **What you'll build**

This sample demonstrates how to pick a file from a folder and process it within the Micro Integrator. In this sample scenario you pick a file from the local directory, insert the records in the file to a database, send an email with the file content, trace and write the log and finally move the file to another directory.

## **Let's get started!**

## 

### **Step 1: Set up the workspace**

Download the relevant [WSO2 Integration Studio](https://wso2.com/api-management/tooling/) based on your operating system.

Let's setup a MySQL database:

1. Manually set up the database.
2. Create a table named info in your schema. You can run the following commands to do this.

delimiter $$

**CREATE** **TABLE** `info` (

`name` varchar(45) **DEFAULT** '',

`surname` varchar(45) **DEFAULT** NULL,

`phone` varchar(45) **DEFAULT** NULL

) **ENGINE**=**InnoDB** **DEFAULT** **CHARSET**=utf8$$

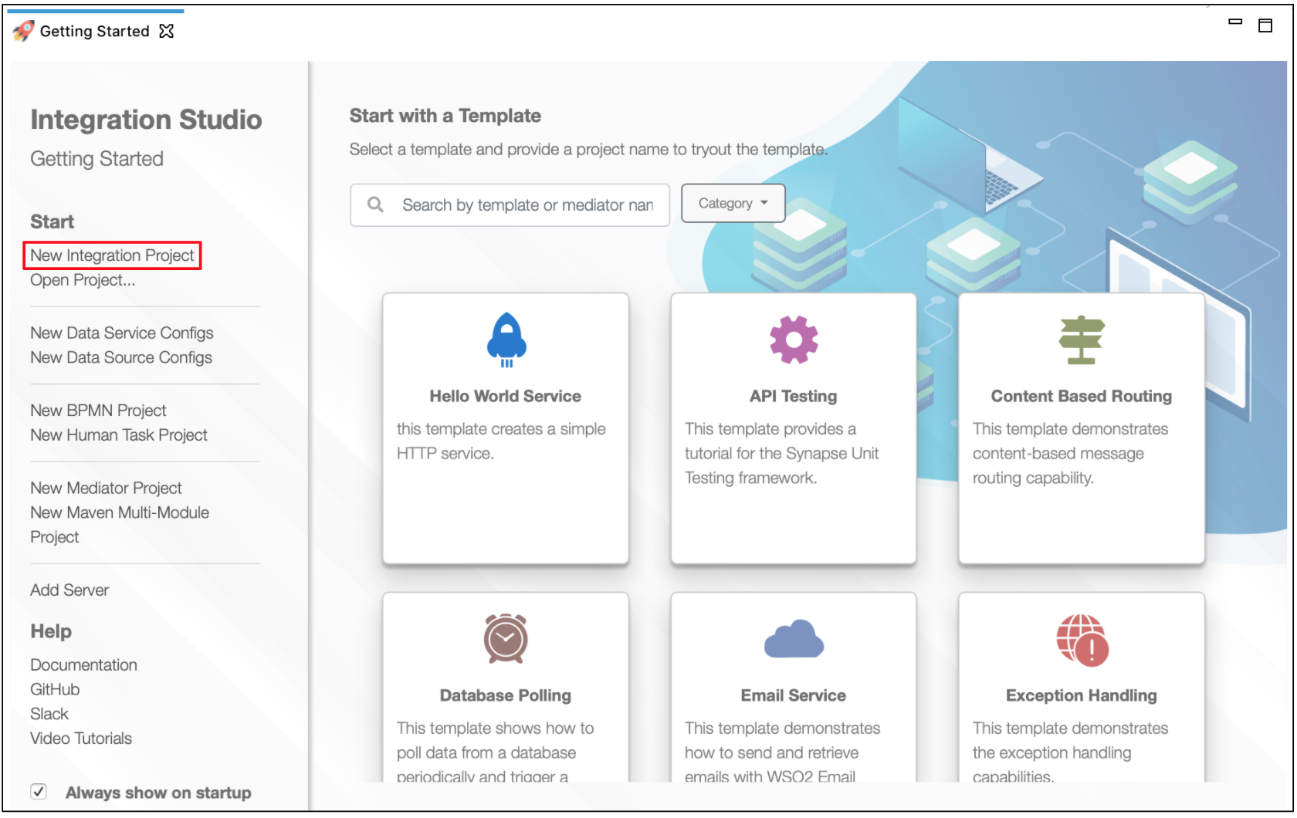
1. Make sure the info table is created and that it contains the following columns:
   * **name**
   * **surname**
   * **phone**

### **Step 2: Develop the integration artifacts**

Follow the instructions given in this section to create and configure the required artifacts.

#### Create an Integration Project

Create an integration project with the following modules: **ESB Configs** and **Composite Exporter** module:

1. Open **WSO2 Integration Studio**.
2. Click **New Integration Project** in the **Getting Started** tab as shown below. 
3. Enter FileProcessingService as the project name.
4. Click **Finish**. The created project is saved in the **Project Explorer**.

#### Create the Main and Fault sequences

1. Create the Main sequence with the following configuration. See the instructions on [creating a sequence](https://apim.docs.wso2.com/en/4.2.0/integrate/develop/creating-artifacts/creating-reusable-sequences).

<**sequence** name="main" xmlns="http://ws.apache.org/ns/synapse">

<**in**>

<**log** level="full"/>

<**filter** regex="http://localhost:9000.\*" source="get-property('To')">

<**then**>

<**send**/>

</**then**>

<**else**/>

</**filter**>

</**in**>

<**out**>

<**send**/>

</**out**>

</**sequence**>

1. Create the Fault sequence with the following configuration. See the instructions on [creating a sequence](https://apim.docs.wso2.com/en/4.2.0/integrate/develop/creating-artifacts/creating-reusable-sequences).

<**sequence** name="fault" trace="disable" xmlns="http://ws.apache.org/ns/synapse">

<**log** level="full">

<**property** name="MESSAGE" value="Executing default 'fault' sequence"/>

<**property** expression="get-property('ERROR\_CODE')" name="ERROR\_CODE"/>

<**property** expression="get-property('ERROR\_MESSAGE')" name="ERROR\_MESSAGE"/>

</**log**>

<**drop**/>

</**sequence**>

#### Create the FileProxy

1. Create a proxy service named FileProxy with the following configuration. See the instructions on [creating a proxy service](https://apim.docs.wso2.com/en/4.2.0/integrate/develop/creating-artifacts/creating-a-proxy-service).

<**proxy** xmlns="http://ws.apache.org/ns/synapse" name="FileProxy" transports="vfs" startOnLoad="true" trace="disable">

<**target**>

<**inSequence**>

<**log** level="full"/>

<**clone**>

<**target** sequence="fileWriteSequence"/>

<**target** sequence="sendMailSequence"/>

<**target** sequence="databaseSequence"/>

</**clone**>

</**inSequence**>

</**target**>

<**parameter** name="transport.vfs.ActionAfterProcess">MOVE</**parameter**>

<**parameter** name="transport.PollInterval">15</**parameter**>

<**parameter** name="transport.vfs.MoveAfterProcess">file:///home/username/test/original</**parameter**>

<**parameter** name="transport.vfs.FileURI">file:///home/username/test/in</**parameter**>

<**parameter** name="transport.vfs.MoveAfterFailure">file:///home/username/test/failure</**parameter**>

<**parameter** name="transport.vfs.FileNamePattern">.\*.txt</**parameter**>

<**parameter** name="transport.vfs.ContentType">text/plain</**parameter**>

<**parameter** name="transport.vfs.ActionAfterFailure">MOVE</**parameter**>

</**proxy**>

1. Edit the proxy service and define the directory to which the original file should be moved after processing.

<parameter name="transport.vfs.MoveAfterProcess">[file:**///**home/]**<username>**/test/original**</parameter>**

1. Edit the proxy service and define where the input file should be placed.

<parameter name="transport.vfs.FileURI">[file:**///**home/]**<username>**/test/in**</parameter>**

1. Edit the proxy service and define the directory to which the file should be moved if an error occurs.

<parameter name="transport.vfs.MoveAfterFailure">[file:**///**home/]**<username>**/test/failure**</parameter>**

#### Create the databaseSequence

Follow the instructions below to create a sequence that can be used to connect to the database to insert data.

1. Create a sequence named databaseSequence with the following configuration. See the instructions on [creating a sequence](https://apim.docs.wso2.com/en/4.2.0/integrate/develop/creating-artifacts/creating-reusable-sequences).

<**sequence** xmlns="http://ws.apache.org/ns/synapse" name="databaseSequence">

<**log** level="full">

<**property** name="sequence" value="before-smooks"/>

</**log**>

<**smooks** config-key="smooks">

<**input** type="text"/>

<**output** type="xml"/>

</**smooks**>

<**log** level="full">

<**property** name="sequence" value="after-smooks"/>

</**log**>

<**iterate** expression="//csv-set/csv-record">

<**target**>

<**sequence**>

<**dbreport**>

<**connection**>

<**pool**>

<**password**>db-password</**password**>

<**user**>db-username</**user**>

<**url**>jdbc:mysql://localhost:3306/test</**url**>

<**driver**>com.mysql.jdbc.Driver</**driver**>

</**pool**>

</**connection**>

<**statement**>

<**sql**><![CDATA[insert into INFO values (?, ?, ?)]]></**sql**>

<**parameter** expression="//csv-record/name/text()" type="VARCHAR"/>

<**parameter** expression="//csv-record/surname/text()" type="VARCHAR"/>

<**parameter** expression="//csv-record/phone/text()" type="VARCHAR"/>

</**statement**>

</**dbreport**>

</**sequence**>

</**target**>

</**iterate**>

</**sequence**>

1. Specify your database username, password, and URL in the <pool> section of the sequence.

#### Create the fileWriteSequence

1. Create a sequence named fileWriteSequence with the following configuration. See the instructions on [creating a sequence](https://apim.docs.wso2.com/en/4.2.0/integrate/develop/creating-artifacts/creating-reusable-sequences).

<**sequence** xmlns="http://ws.apache.org/ns/synapse" name="fileWriteSequence">

<**log** level="custom">

<**property** name="sequence" value="fileWriteSequence"/>

</**log**>

<**property** xmlns:ns2="http://org.apache.synapse/xsd" name="transport.vfs.ReplyFileName" expression="fn:concat(fn:substring-after(get-property('MessageID'), 'urn:uuid:'), '.txt')" scope="transport"/>

<**property** name="OUT\_ONLY" value="true"/>

<**send**>

<**endpoint** name="FileEpr">

<**address** uri="vfs:file:///home/username/test/out"/>

</**endpoint**>

</**send**>

</**sequence**>

1. Edit the sequence and define the directory to which the file should be moved.

#### Create the sendMailSequence

1. Create a sequence named sendMailSequence with the following configuration. See the instructions on [creating a sequence](https://apim.docs.wso2.com/en/4.2.0/integrate/develop/creating-artifacts/creating-reusable-sequences).

<**sequence** xmlns="http://ws.apache.org/ns/synapse" name="sendMailSequence">

<**log** level="custom">

<**property** name="sequence" value="sendMailSequence"/>

</**log**>

<**property** name="messageType" value="text/html" scope="axis2"/>

<**property** name="ContentType" value="text/html" scope="axis2"/>

<**property** name="Subject" value="File Received" scope="transport"/>

<**property** name="OUT\_ONLY" value="true"/>

<**send**>

<**endpoint** name="FileEpr">

<**address** uri="mailto:username@gmail.com"/>

</**endpoint**>

</**send**>

</**sequence**>

1. Edit the sequence and define the e-mail address to which the notification should be sent.

#### Create the Smooks configuration

Create a smooks configuration file (for example smooks-config.xml) as shown below and save it to a location on your computer.

<**smooks-resource-list** xmlns="http://www.milyn.org/xsd/smooks-1.0.xsd">

<!--Configure the CSVParser to parse the message into a stream of SAX events. -->

<**resource-config** selector="org.xml.sax.driver">

<**resource**>org.milyn.csv.CSVReader</**resource**>

<**param** name="fields" type="string-list">name,surname,phone</**param**>

</**resource-config**>

</**smooks-resource-list**>

#### Create a local registry entry

Configure a local entry as shown below. This local entry will be used to refer to the [smooks configuration](https://apim.docs.wso2.com/en/4.2.0/tutorials/integration-tutorials/file-processing/" \l "create-the-smooks-configuration). See the instructions on [creating a local registry configuration](https://apim.docs.wso2.com/en/4.2.0/integrate/develop/creating-artifacts/registry/creating-local-registry-entries).

<**localEntry** key="smooks" src="file:resources/smooks-config.xml"/>

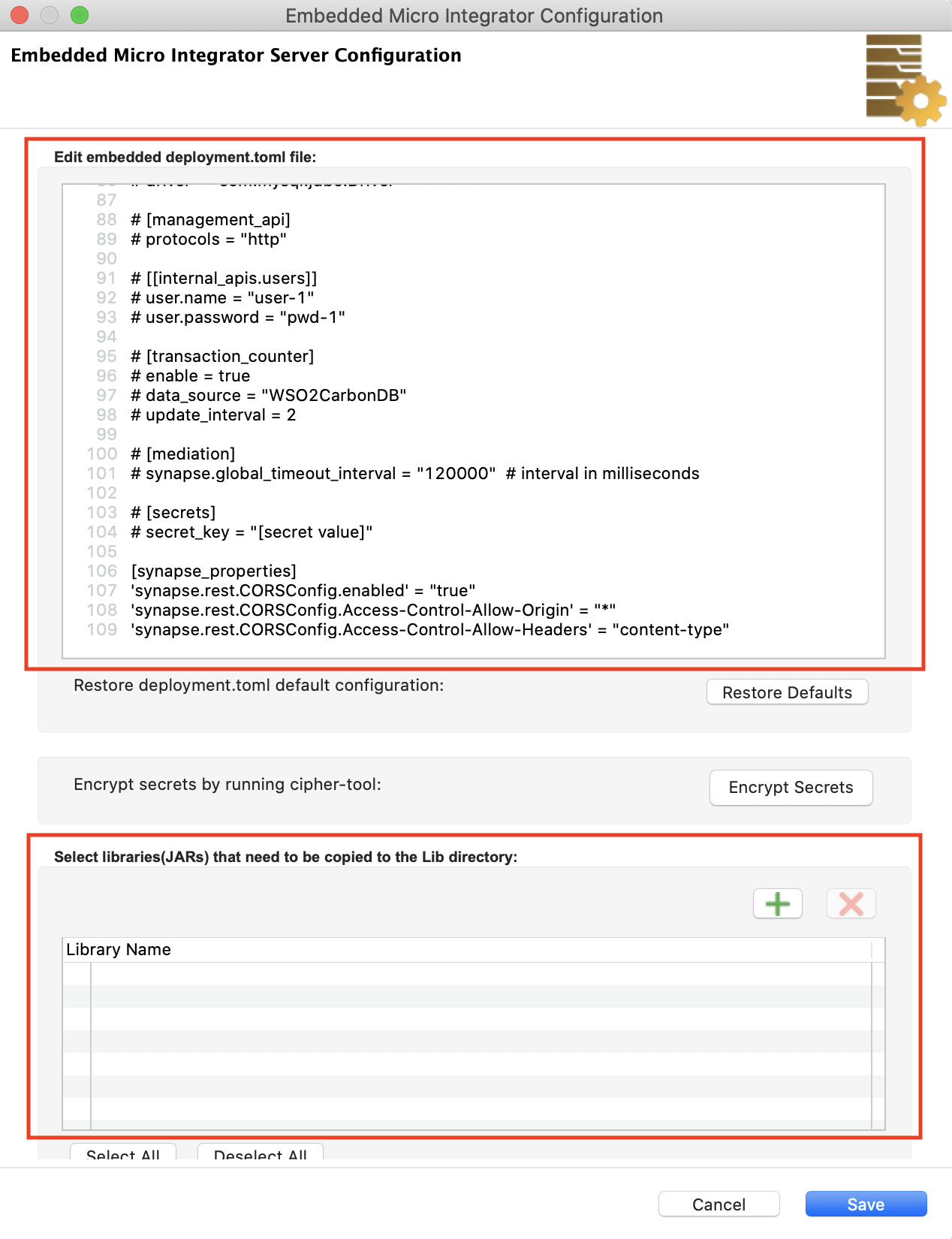
### **Step 3: Package the artifacts**

Package the artifacts in your composite application module to be able to deploy the artifacts in the server.

1. Open the pom.xml file in the composite exporter module.
2. Ensure that the relevant artifacts are selected in the POM file.
3. Save the changes.

### **Step 4: Configure the Micro Integrator server**

1. Click the **Embedded Micro Integrator Configuration** () icon on the upper menu to open the dialog box.



1. Add the following server configurations (to the deployment.toml file) using the upper section in the dialog box.
   * The **VFS** transport is enabled in the Micro Integrator by default. Enable the [MailTo transport](https://apim.docs.wso2.com/en/4.2.0/install-and-setup/setup/mi-setup/transport_configurations/configuring-transports/#configuring-the-mailto-transport) for sending the email message as shown below and update the values:

[[transport.mail.sender]]

name = "mailto"

parameter.hostname = "smtp.gmail.com"

parameter.port = "587"

parameter.enable\_tls = true

parameter.auth = true

parameter.username = "demo\_user"

parameter.password = "mailpassword"

parameter.from = "demo\_user@wso2.com"

**Note**

In this sample, you will not retrieve mails from a mailbox. Therefore, you do not need to enable the mailto transport receiver.

* + Add the following message formatter:

text\_xml = "org.apache.axis2.transport.http.ApplicationXMLFormatter"

1. Click the (https://apim.docs.wso2.com/en/4.2.0/assets/img/integrate/tutorials/common/plus-icon.png) icon in the lower section and add the following drivers and libraries.
   * [MySQL database driver](https://github.com/wso2-docs/WSO2_EI/blob/master/Integration-Tutorial-Artifacts/Integration-Tutorial-Artifacts-EI7.1.0/EI7.1.0-file-processing-tutorial-JARS/mysql-connector-java-5.1.10-bin.jar).
   * [CSV smooks library](https://github.com/wso2-docs/WSO2_EI/blob/master/Integration-Tutorial-Artifacts/Integration-Tutorial-Artifacts-EI7.1.0/EI7.1.0-file-processing-tutorial-JARS/milyn-smooks-csv-1.2.4.jar).

**Note**

These are copied to the /lib folder of the embedded Micro Integrator.

### **Step 5: Build and run the artifacts**

To test the artifacts, deploy the [packaged artifacts](https://apim.docs.wso2.com/en/4.2.0/tutorials/integration-tutorials/file-processing/#step-3-package-the-artifacts) in the embedded Micro Integrator:

1. Right-click the composite exporter module and click **Export Project Artifacts and Run**.
2. In the dialog box that opens, confirm that the required artifacts from the composite exporter module are selected.
3. Click **Finish**.

The artifacts will be deployed in the embedded Micro Integrator and the server will start.

* See the startup log in the **Console** tab.
* See the URLs of the deployed services and APIs in the **Runtime Services** tab.

### **Step 6: Test the use case**

#### Create the input file

Create a text file with the following format:

name\_1, surname\_1, phone\_1

name\_2, surname\_2, phone\_2

Save the file in the .txt format to the in directory that you specified.

#### Analyze the result

The Micro Integrator listens on a local file system directory. When a file is dropped into the in directory, the Micro Integrator picks this file.

1. Make sure the file appears in the out directory.
2. The Micro Integrator inserts the records from the text file to the database. Make sure the data is in the info table. The following screenshot displays the content of the test.info table with the data from the file.
3. Make sure the original file is moved to the /home/<username>/test/original directory.
4. Make sure the e-mail notification is sent to the email address that is specified. The message should contain the file data. The following screenshot displays a notification received. If you see the error message Username and Password not accepted in the logs, you might need to turn on Allow less secure apps in your Google account from [here](https://myaccount.google.com/lesssecureapps).

 [CC BY 4.0](http://creativecommons.org/licenses/by/4.0/)