

Talend Open Studio for ESB Getting Started Guide

7.0.1

Contents

Copyleft.....	3
Introduction to Talend Open Studio for ESB.....	5
Functional architecture of Talend Open Studio for ESB.....	5
Prerequisites to using Talend Open Studio for ESB.....	6
Memory requirements.....	6
Software requirements.....	6
Installing Java.....	7
Setting up the Java environment variable on Windows.....	7
Setting up the Java environment variable on Linux.....	8
Installing 7-Zip (Windows).....	8
Downloading and installing Talend Open Studio for ESB.....	9
Downloading Talend Open Studio for ESB.....	9
Installing Talend Open Studio for ESB.....	9
Configuring and setting up your Talend product.....	11
Launching Talend Runtime and its Infrastructure Services.....	11
Launching the Studio for the first time.....	12
Logging in to the Studio.....	12
Installing additional packages.....	12
Working with Data Services.....	14
Looking up customer information from a database.....	14
What's next?.....	31

Copyleft

Adapted for 7.0.1. Supersedes previous releases.

Publication date: April 13, 2018

This documentation is provided under the terms of the Creative Commons Public License (CCPL).

For more information about what you can and cannot do with this documentation in accordance with the CCPL, please read: <http://creativecommons.org/licenses/by-nc-sa/2.0/>.

Notices

Talend and Talend ESB are trademarks of Talend, Inc.

Talend, Talend Integration Factory, Talend Service Factory, and Talend ESB are trademarks of Talend, Inc.

Apache CXF, CXF, Apache Karaf, Karaf, Apache Camel, Camel, Apache Maven, Maven, Apache Syncope, Syncope, Apache ActiveMQ, ActiveMQ, Apache Log4j, Log4j, Apache Felix, Felix, Apache ServiceMix, ServiceMix, Apache Ant, Ant, Apache Derby, Derby, Apache Tomcat, Tomcat, Apache ZooKeeper, ZooKeeper, Apache Jackrabbit, Jackrabbit, Apache Santuario, Santuario, Apache DS, DS, Apache Avro, Avro, Apache Abdera, Abdera, Apache Chemistry, Chemistry, Apache CouchDB, CouchDB, Apache Kafka, Kafka, Apache Lucene, Lucene, Apache MINA, MINA, Apache Velocity, Velocity, Apache FOP, FOP, Apache HBase, HBase, Apache Hadoop, Hadoop, Apache Shiro, Shiro, Apache Axiom, Axiom, Apache Neethi, Neethi, Apache WSS4J, WSS4J are trademarks of The Apache Foundation. Eclipse Equinox is a trademark of the Eclipse Foundation, Inc. Hyperic is a trademark of VMware, Inc. Nagios is a trademark of Nagios Enterprises, LLC.

All brands, product names, company names, trademarks and service marks are the properties of their respective owners.

License Agreement

The software described in this documentation is licensed under the Apache License, Version 2.0 (the "License"); you may not use this software except in compliance with the License. You may obtain a copy of the License at <http://www.apache.org/licenses/LICENSE-2.0.html>. Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

This product includes software developed at AOP Alliance (Java/J2EE AOP standards), ASM, AntLR, Apache ActiveMQ, Apache Ant, Apache Avro, Apache Axiom, Apache Axis, Apache Axis 2, Apache Batik, Apache CXF, Apache Camel, Apache Chemistry, Apache Common Http Client, Apache Common Http Core, Apache Commons, Apache Commons Bcel, Apache Commons JXPath, Apache Commons Lang, Apache Derby Database Engine and Embedded JDBC Driver, Apache Geronimo, Apache Hadoop, Apache Hive, Apache HttpClient, Apache HttpComponents Client, Apache JAMES, Apache Log4j, Apache Lucene Core, Apache Neethi, Apache POI, Apache Pig, Apache Qpid-Jms, Apache Tomcat, Apache Velocity, Apache WSS4J, Apache WebServices Common Utilities, Apache Xml-RPC, Apache Zookeeper, Box Java SDK (V2), CSV Tools, DataStax Java Driver for Apache Cassandra, Ehcache, Ezmorph, Ganymed SSH-2 for Java, Google APIs Client Library for Java, Google Gson, Groovy, Guava: Google Core Libraries for Java, H2 Embedded Database and JDBC Driver, Hsqldb, Ini4j, JClouds, JLine, JSON, JSR 305: Annotations for Software Defect Detection in Java, JUnit, Jackson Java JSON-processor, Java API for RESTful Services, Jaxb, Jaxen, Jettison, Jetty, Joda-Time, Json Simple, MetaStuff, Mondrian, OpenSAML, Paracel JDBC Driver, PostgreSQL JDBC Driver, Resty: A simple HTTP REST client for Java,

Rocoto, SL4J: Simple Logging Facade for Java, SQLite JDBC Driver, Simple API for CSS, SshJ, StAX API, StAXON - JSON via StAX, Talend Camel Dependencies (Talend), The Castor Project, The Legion of the Bouncy Castle, W3C, Woden, Woodstox : High-performance XML processor, XML Pull Parser (XPP), Xalan-J, Xerces2, XmlBeans, XmlSchema Core, Xmlsec - Apache Santuario, Zip4J, atinject, dropbox-sdk-java: Java library for the Dropbox Core API, google-guice. Licensed under their respective license.

Introduction to Talend Open Studio for ESB

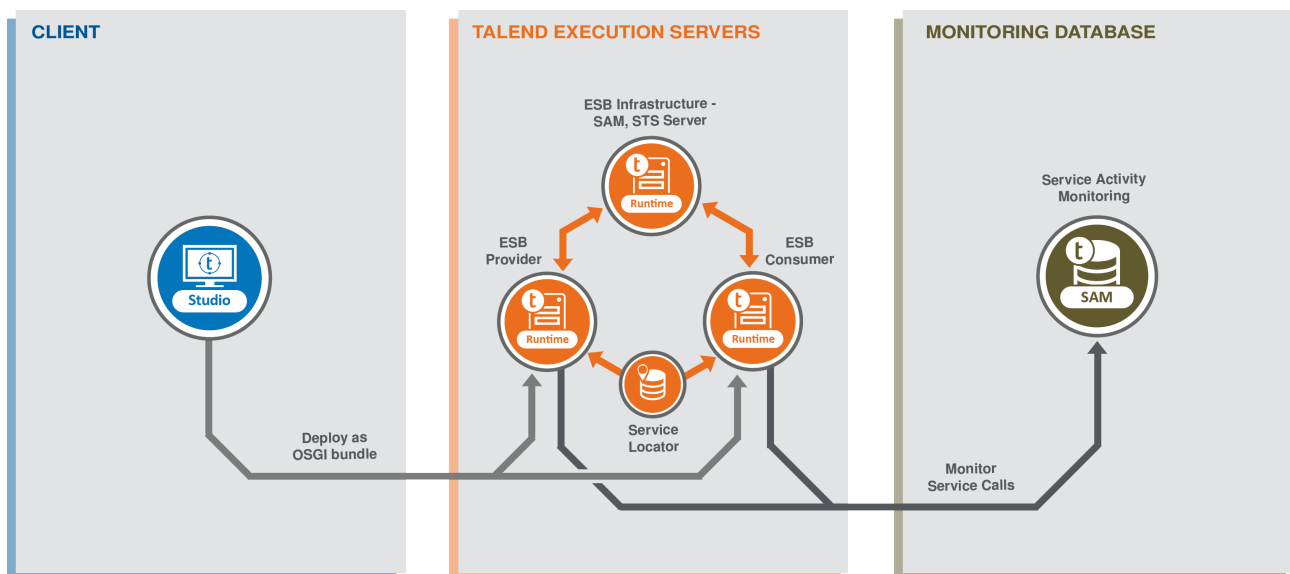
Talend provides unified development and management tools to integrate and process all of your data with an easy to use, visual designer.

Talend's ESB solution provides a versatile and flexible, enterprise service bus (ESB) that allows organizations to address any integration challenge - from simple departmental projects to complex, heterogeneous IT environments.

Functional architecture of Talend Open Studio for ESB

The Talend Open Studio for ESB functional architecture is an architectural model that identifies Talend Open Studio for ESB functions, interactions and corresponding IT needs. The overall architecture has been described by isolating specific functionalities in functional blocks.

The following chart illustrates the main architectural functional blocks.



The different types of functional blocks are:

- The Client block includes a Talend Studio where you can carry out data integration or data service processes, mediation routes and services.
- The Talend Execution Servers block represents one or more Talend Runtime servers (execution container) deployed inside your information system. Talend Runtime enables you to deploy and execute the Jobs, Routes and Services created in the Studio.
- The Monitoring Databases block represents the Service Activity Monitoring database to monitor service calls.

Prerequisites to using Talend Open Studio for ESB

This chapter provides basic software and hardware information required and recommended to get started with your Talend Open Studio for ESB.

- [Memory requirements](#) on page 6
- [Software requirements](#) on page 6

It also guides you to install and configure required and recommended third-party tools:

- [Installing Java](#) on page 7
- [Setting up the Java environment variable on Windows](#) on page 7 or [Setting up the Java environment variable on Linux](#) on page 8
- [Installing 7-Zip \(Windows\)](#) on page 8

Memory requirements

To make the most out of your Talend product, please consider the following memory and disk space usage:

Memory usage	3GB minimum, 4 GB recommended
Disk space	3GB

Software requirements

To make the most out of your Talend product, please consider the following system and software requirements:

Required software

- Operating System for Talend Studio:

Support type	Operating system (64 bits only)
Recommended	Ubuntu 16.04 LTS
Recommended	Microsoft Windows 10
Supported	Apple macOS 10.13/High Sierra
	Apple macOS 10.12/Sierra
	Apple OS X 10.11/El Capitan

- Operating System for Talend Server modules:

Support type	Operating system (64 bits only)
Recommended	Microsoft Windows Server 2016
Recommended	Red Hat Enterprise Linux Server/CentOS 7.4
Supported	Ubuntu 17.04
	Ubuntu 16.04 LTS
	Ubuntu 14.04 LTS

- Java 8 JRE Oracle. See [Installing Java](#) on page 7.
- A properly installed and configured MySQL database, with a database named `gettingstarted`.

Optional software

- 7-Zip. See [Installing 7-Zip \(Windows\)](#) on page 8.

Installing Java

To use your Talend product, you need Oracle Java Runtime Environment installed on your computer.

Procedure

1. From the [Java SE Downloads](#) page, under **Java Platform, Standard Edition**, click the **JRE Download**.
2. From the **Java SE Runtime Environment 8 Downloads** page, click the radio button to **Accept License Agreement**.
3. Select the appropriate download for your Operating System.
4. Follow the Oracle installation steps to install Java.

Results

When Java is installed on your computer, you need to set up the `JAVA_HOME` environment variable. For more information, see:

- [Setting up the Java environment variable on Windows](#) on page 7.
- [Setting up the Java environment variable on Linux](#) on page 8.

Setting up the Java environment variable on Windows

Prior to installing your Talend product, you need to set the `JAVA_HOME` and Path environment variables.

Procedure

1. Go to the **Start Menu** of your computer, right-click on **Computer** and select **Properties**.
2. In the **Control Panel Home** window, click **Advanced system settings**.
3. In the **System Properties** window, click **Environment Variables....**

4. Under **System Variables**, click **New...** to create a variable. Name the variable `JAVA_HOME`, enter the path to the Java 8 JRE, and click **OK**.

Example of default JRE path: `C:\Program Files\Java\jre1.8.0_77`.

5. Under **System Variables**, select the **Path** variable and click **Edit...** to add the previously defined `JAVA_HOME` variable at the end of the `Path` environment variable, separated with semi colon.

Example: `<PathVariable>;%JAVA_HOME%\bin`.

Setting up the Java environment variable on Linux

Prior to installing your Talend product, you have to set the `JAVA_HOME` and `Path` environment variables.

Procedure

1. Find the JRE installation home directory.

Example: `/usr/lib/jvm/jre1.8.0_65`

2. Export it in the `JAVA_HOME` environment variable.

Example:

```
export JAVA_HOME=/usr/lib/jvm/jre1.8.0_65
export PATH=$JAVA_HOME/bin:$PATH
```

3. Add these lines at the end of the user profiles in the `~/.profile` file or, as a superuser, at the end of the global profiles in the `/etc/profile` file.
4. Log on again.

Installing 7-Zip (Windows)

Talend recommends to install 7-Zip and to use it to extract the installation files: <http://www.7-zip.org/download.html>.

Procedure

1. Download the 7-Zip installer corresponding to your Operating System.
2. Navigate to your local folder, locate and double-click the 7z exe file to install it.

Results

The download will start automatically.

Downloading and installing Talend Open Studio for ESB

Talend Open Studio for ESB is easy to install. After downloading it from Talend's Website, a simple unzipping will install it on your computer.

This chapter provides basic information useful to download and install it.

Downloading Talend Open Studio for ESB

Talend Open Studio for ESB is a free open source product that you can download directly from Talend's Website.

Procedure

1. Go to the Talend Open Studio for ESB [download page](#).
2. Click **DOWNLOAD FREE TOOL**.

Results

The download will start automatically.

Installing Talend Open Studio for ESB

Installation is done by unzipping the zip file previously downloaded.

This can be done either by using:

- 7Zip (Windows recommended): [Extracting via 7-Zip \(Windows recommended\)](#) on page 9.
- Windows default unzipper: [Extracting via Windows default unzipping tool](#) on page 10.
- Linux default unzipper (for a Linux based Operating System): [Extracting via Windows default unzipping tool](#) on page 10.

Once extracted, you get ready to use Talend Studio and Talend Runtime.

Extracting via 7-Zip (Windows recommended)

For Windows, Talend recommends you to install 7-Zip and use it to extract files. For more information, see [Installing 7-Zip \(Windows\)](#) on page 8.

To install the studio, follow the steps below:

Procedure

1. Navigate to your local folder, locate the **TOS** zip file and move it to another location with a path as short as possible and without any space character.

Example: `C:/Talend/`

2. Unzip it by right-clicking on the compressed file and selecting **7-Zip > Extract Here**.

Extracting via Windows default unzipping tool

If you do not want to use 7-Zip, you can use Windows default unzipping tool.

Procedure

1. Unzip it by right-click the compressed file and select **Extract All**.
2. Click **Browse** and navigate to the C: drive.
3. Select **Make new folder** and name the folder `Talend`. Click **OK**.
4. Click **Extract** to begin the installation.

Extracting via the Linux GUI unzipper

To install the studio, follow the steps below:

Procedure

1. Navigate to your local folder, locate the zip file and move it to another location with a path as short as possible and without any space character.

Example: `home/user/talend/`

2. Unzip it by right-clicking on the compressed file and selecting **Extract Here**.

Configuring and setting up your Talend product

This chapter provides basic information required to configure and set up your Talend Open Studio for ESB.

Launching Talend Runtime and its Infrastructure Services

Talend Runtime includes the execution server, Talend Runtime Container and its infrastructure services, including: Service Locator, Service Activity Monitoring and Security Token Service.

To fully benefit from all these functionalities, do the following:

Procedure

1. Go to subdirectory `<TalendRuntimePath>\container\bin` of the Talend Runtime installation directory.
2. Run the `trun.bat` (Windows) or `trun.sh` (Linux) file.

When the container starts up, you will see a short introduction (similar to the one below) followed by the Talend Runtime Container console command prompt:

```
Hit '<tab>' for a list of available commands
and '[cmd] --help' for help on a specific command.
Hit '<ctrl-d>' or type 'system:shutdown' to shutdown TRUN.
karaf@trun(>
```

3. After starting the Talend Runtime Container, wait a few seconds for initialization to complete before going to the next steps. Karaf, on which the Talend Runtime Container is built, starts the noncore bundles in the background. So even if the console is already available, the commands may not be.
4. At the Console prompt: **karaf@trun>**, execute the `list` command.
This will list all the OSGi Bundles installed in the Talend Runtime Container and let you know if they are active, or not yet.
5. Once the Talend Runtime Container and its OSGi Bundles start, execute the `tesb:start-all` command to start all the Infrastructure Services at once.

Now, the Service Locator, Service Activity Monitoring and Security Token Service services are running as features in the Talend Runtime Container.

Results

For more information on how to launch the Talend Runtime Container and each of the above services individually or as standalone, see the Talend Open Studio for ESB Installation and Upgrade Guide.

You can find further information about the Talend Runtime Container and how to get started with it in the Talend ESB Container Administration Guide.

For more information on Talend ESB Infrastructure Services, see the Talend ESB Infrastructure Services Configuration Guide.

Launching the Studio for the first time

The Studio installation directory contains binaries for several platforms including Mac OS X and Linux/Unix.

To open the Talend Studio for the first time, do the following:

Procedure

1. Double-click the executable file corresponding to your operating system, for example:
 - TOS_*-win-x86_64.exe, for Windows.
 - TOS_*-linux-gtk-x86_64, for Linux.
 - TOS_*-macosx-cocoa.app, for Mac.
2. In the **User License Agreement** dialog box that opens, read and accept the terms of the end user license agreement to proceed.

Logging in to the Studio

To log in to the Talend Studio for the first time, do the following:

Procedure

1. In the Talend Studio login window, select **Create a new project**, specify the project name: `getting_started` and click **Finish** to create a new local project.
2. Depending on the product you are using, either of the following opens:
 - the **Quick Tour**. Play it to get more information on the User Interface of the Studio, and click **Stop** to end it.
 - the **Welcome page**. Follow the links to get more information about the Studio, and click **Start Now!** to close the page and continue opening the Studio.

Tip:

After your Studio successfully launches, you can also click the **Videos** link on the top of the Studio main window to watch a couple of short videos that help you get started with your Talend Studio. For some operating systems, you may need to install an MP4 decoder/player to play the videos.

Results

Now you have successfully logged in to the Talend Studio. Next you need to install additional packages required for the Talend Studio to work properly.

Installing additional packages

Talend recommends that you install additional packages, including third-party libraries and database drivers, as soon as you log in to your Talend Studio to allow you to fully benefit from the functionalities of the Studio.

Procedure

1. When the **Additional Talend Packages** wizard opens, install additional packages by selecting the **Required** and **Optional third-party libraries** check boxes and clicking **Finish**.

This wizard opens each time you launch the studio if any additional package is available for installation unless you select the **Do not show this again** check box. You can also display this wizard by selecting **Help > Install Additional Packages** from the menu bar.

For more information, see the section about installing additional packages in the Talend Open Studio for ESB Installation and Upgrade Guide

2. In the **Download external modules** window, click the **Accept all** button at the bottom of the wizard to accept all the licenses of the external modules used in the studio.

Depending on the libraries you selected, you may need to accept their license more than once.

Wait until all the libraries are installed before starting to use the studio.

3. If required, restart your Talend Studio for certain additional packages to take effect.

Working with Data Services

This chapter takes the example of a company that provides movie rental and streaming video services, and shows how such a company could make use of Talend Open Studio for ESB.

You will work with data about your customers as you learn how to look up the customer email address and phone number from the Customer Support System by customer ID.

Looking up customer information from a database

In this scenario, you will learn:

- How to set up the input data. See [Setting up input data](#) on page 14 for details.
- How to build a web service provider. See [Building a Customer service provider](#) on page 15 for details.
- How to build a web service consumer. See [Building a Customer consumer](#) on page 25 for details.
- How to run the web service in a Talend ESB Container. See [Running the service](#) on page 29 for details.

Setting up input data

The example in this document assumes that the customer data you want to look up is stored in a MySQL database.

If you want to replicate the example and use the exact input data, you can download the `gettingstarted.sql` file of the customer data and then import it in a MySQL database.

Before you begin

- You have an access to a MySQL database.
- You have downloaded `tos_esb_gettingstarted_source_files.zip` from the **Downloads** tab of the online version of this page at <https://help.talend.com>, and stored the source file `gettingstarted.sql` locally.

Procedure

1. Open the MySQL Workbench to launch an instance of the database.
2. From the menu bar, select **Server > Data Import** to open the import wizard wizard.
3. Select the **Import from Self-Contained File** option and browse to where you have stored the `gettingstarted.sql` file.
4. Select the schema to which you want to import the data, or click **New...** to define a new schema.
5. Click **Start Import** in the lower right corner.

Results

The `gettingstarted` database is imported in the MySQL database.

Building a Customer service provider

This section provides you with step-by-step instructions on building the Customer service provider that will give access to the Web service via a WSDL, send a request and retrieve the response.

Creating a service

In this section, you will create a WSDL to define the Customer service.

Procedure

1. In the **Integration** perspective of Talend Studio, right-click the **Services** node in the **Repository** tree view, and select **Create Service** from the contextual menu.

Services

New Service

Add a service in the repository

Name: CustomerService

Purpose: demo

Description: A service that accepts a customer id and then returns the customer phone number and email address in turn.

Author: user@talend.com

Locker:

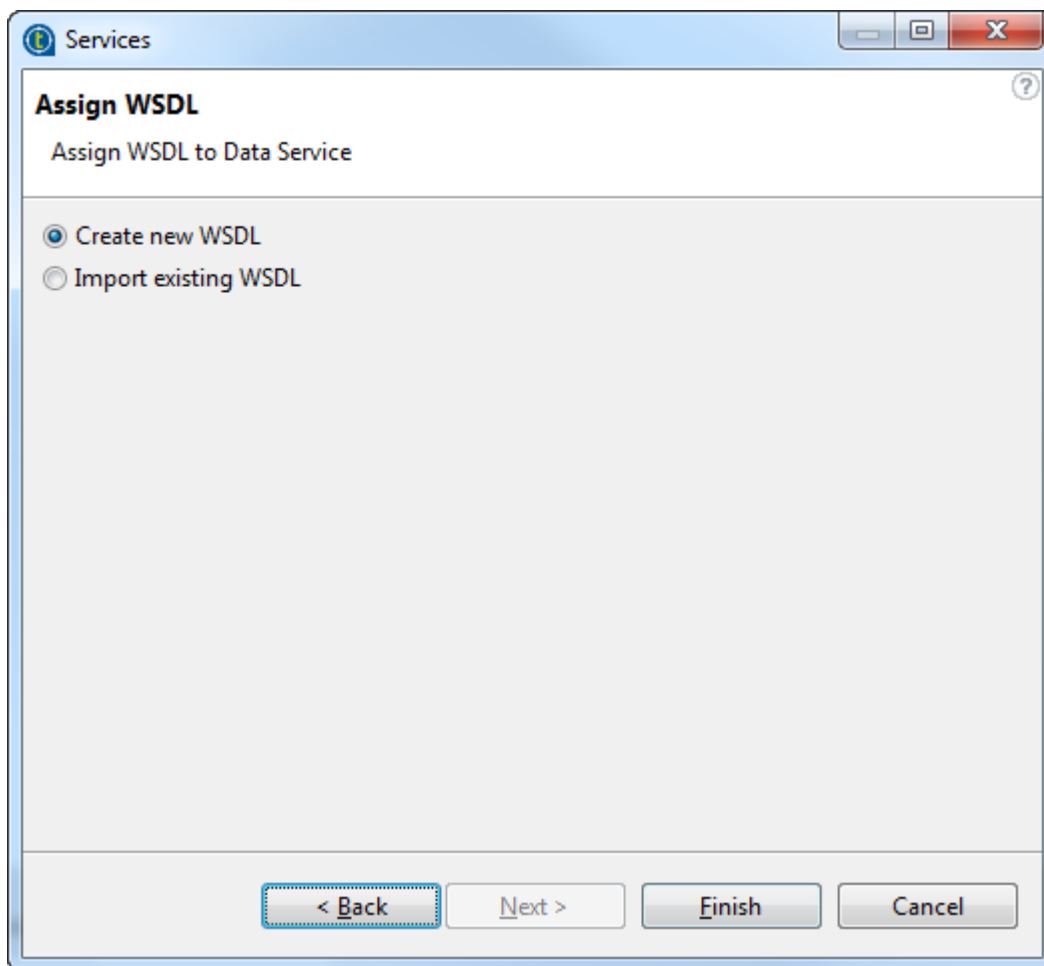
Version: 0.1 M m

Status:

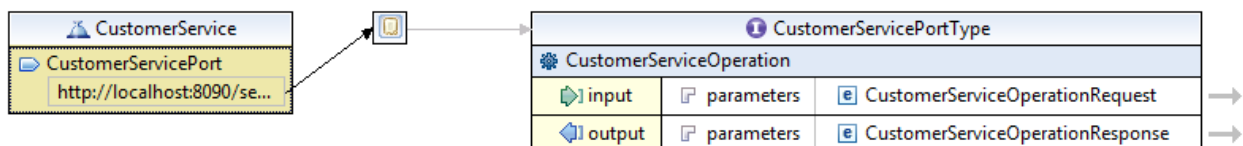
Path: Select

< Back Next > Finish Cancel

2. In the pop-up wizard, enter the name `CustomerService`, the purpose `demo` and a description of the service, and then click **Next**.
3. Select the **Create new WSDL** option, and then click **Finish**.




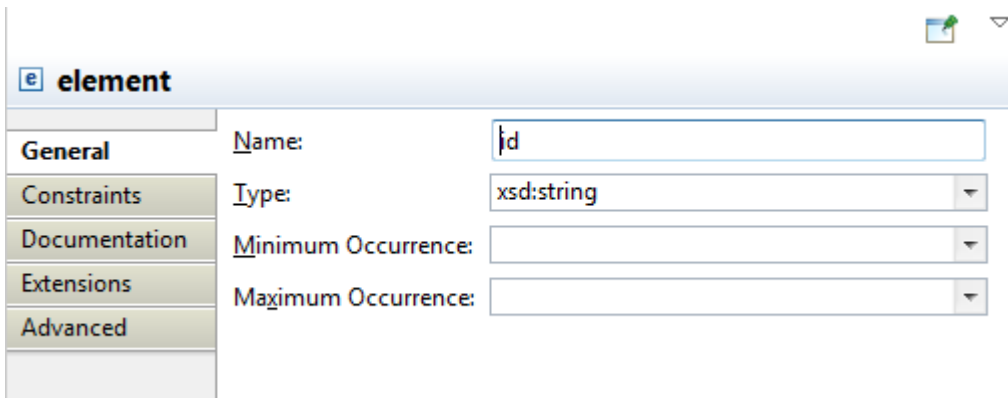
The service opens in the design workspace with a basic WSDL skeleton, which contains one service, one binding and one port type of one operation.



- Click the arrow icon to the right of the input element, `CustomerServiceOperationRequest` in the WSDL skeleton. The schema editor opens, allowing you to define the schema of the request message.



5. Right-click the `in` element and select **Show properties** in the context menu. In the **Properties** view, change its name to `id` in the **Name** field, as in this use case, the request message will be the customer id. Click the  icon in the menu bar of the Talend Studio to save the schema and close it.

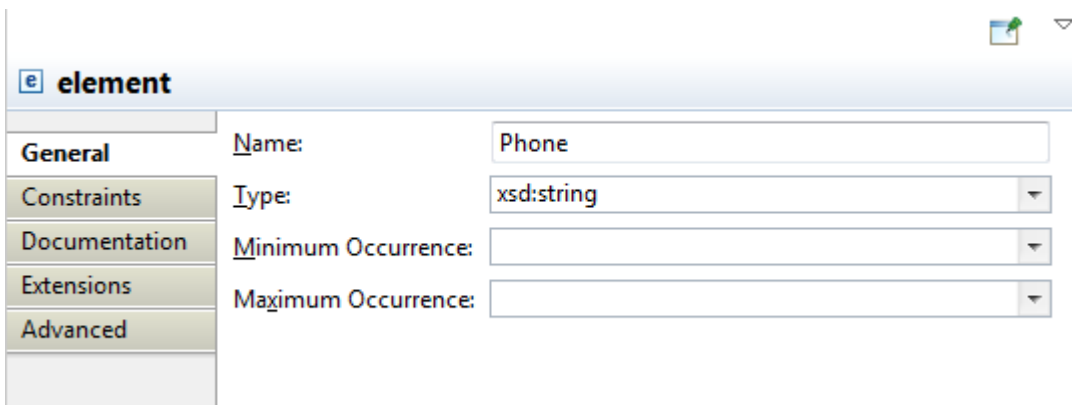


element	
General	Name: <input type="text" value="id"/>
Constraints	Type: <input type="text" value="xsd:string"/>
Documentation	Minimum Occurrence: <input type="text" value="1"/>
Extensions	Maximum Occurrence: <input type="text" value="1"/>
Advanced	


6. In the WSDL skeleton, click the arrow icon to the right of the output element, `CustomerServiceOperationResponse` in the WSDL skeleton to edit the schema of the response message in the schema editor.

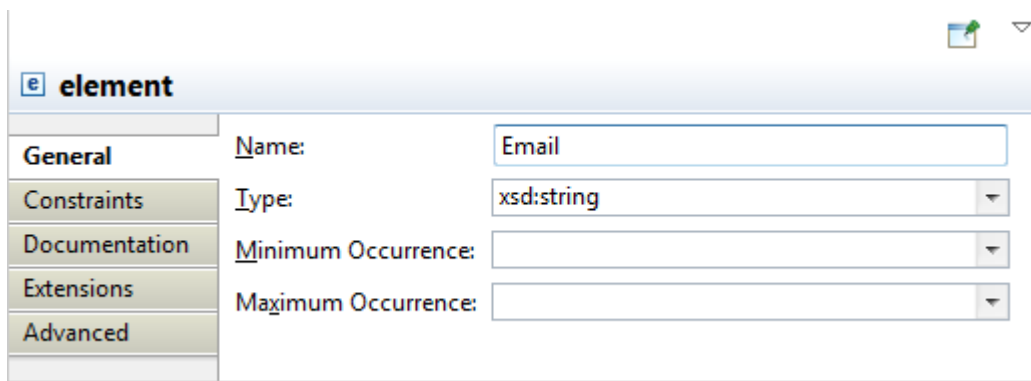


7. In this use case, there will be two rows, email and phone of the customer in the response message. Click the `out` element and change its name to `Phone` in the **Properties** view.



element	
General	Name: <input type="text" value="Phone"/>
Constraints	Type: <input type="text" value="xsd:string"/>
Documentation	Minimum Occurrence: <input type="text" value="1"/>
Extensions	Maximum Occurrence: <input type="text" value="1"/>
Advanced	

8. In the design workspace, right-click the `Phone` element and select the **Insert Element > After** in the context menu. Give the name `Email` to it in the **Properties** view. Click the  icon in the menu bar of the Talend Studio to save the schema and close it.

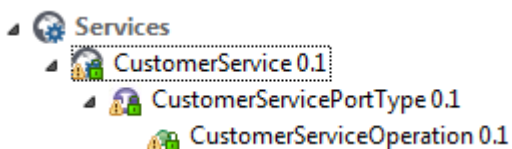


element	
General	Name: Email
Constraints	Type: xsd:string
Documentation	Minimum Occurrence:
Extensions	Maximum Occurrence:
Advanced	

9. Save the WSDL file. It will be used to build the Web service.

Results

The newly defined Web service with exclamation icon is then shown under the **Services** node of the **Repository** view. The exclamation icon means that this defined Web service is not yet used.

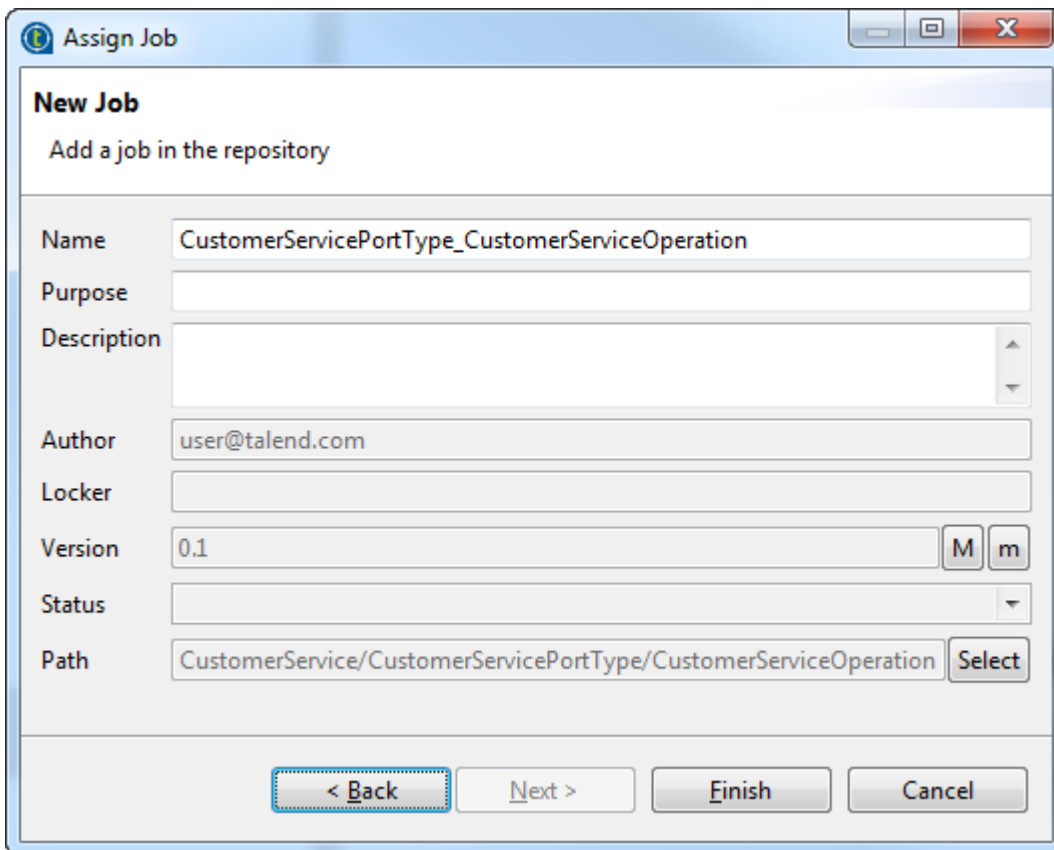


Configuring and exposing the service

In this scenario, the data service provider uses the **tESBProviderRequest** and the **tESBProviderResponse** components to create the access to the Customer Web service and uses the **tXMLMap** component to join the Customer data provided by a given MySQL database into the request-response main flow for publication. The database data is loaded by the **tMysqlInput** component.

Procedure

1. Save the service details and WSDL Request / Response data types to the Metadata so that they can be accessible to other components. In **Services**, right-click **CustomerService 0.1** and select **Import WSDL Schemas**.
This option imports the WSDL metadata from the service into the **Repository**, under the **Metadata > File xml**, which allows you to share the operations details across services and other components.
2. Expand the elements displayed in **CustomerService 0.1**, right-click **CustomerServiceOperation 0.1** and select **Assign Job**.
3. The **Assign Job** wizard opens. Select the **Create a new Job and Assign it to this Service Operation** and click **Next**.
4. In the **New Job** view of the wizard, the Job to be created is already named automatically, so simply click **Finish**.



Assign Job

New Job
Add a job in the repository

Name: CustomerServicePortType_CustomerServiceOperation

Purpose:

Description:

Author: user@talend.com

Locker:

Version: 0.1 [M] [m]

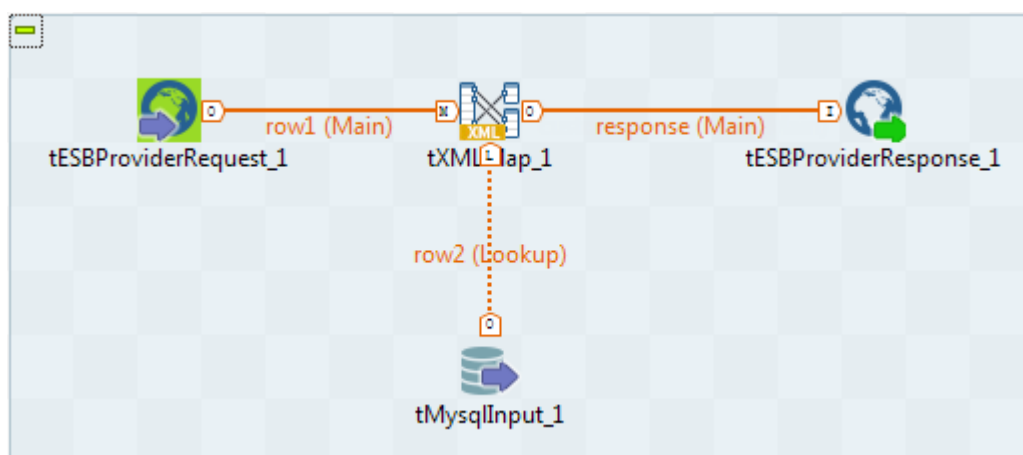
Status:

Path: CustomerService/CustomerServicePortType/CustomerServiceOperation [Select]

< Back Next > Finish Cancel

A default template of the Job is opened on the workspace. In the template, a **tESBProviderRequest** and a **tESBProviderResponse** are already selected and configured. **tESBProviderRequest** will send a request to the specified Web service and **tESBProviderResponse** will send back the response corresponding to the request. These two components can be found in the **ESB** group of the **Palette**.

- Now add a **tXMLMap** between the two ESB components by typing its name on the design workspace.



- Right-click the **tESBProviderRequest_1** and select **Row**, then **Main** and drop the end of the line on **tXMLMap_1**.
- Right-click **tXMLMap_1**, select **Row > Main** and drop the end of the line on **tESBProviderResponse_1**. Give it the name **response**, and click **OK**. Click the default **Yes** when asked if you wish to import the schemas.
- Add a **tMysqlInput** below the **tXMLMap** by typing its name on the design workspace to load the customer data in a MySQL database.
- Right-click **tMysqlInput**, select **Row > Main** and drop the end of the line on **tXMLMap**.

Configuring the service operation

In this section, the service operation is customized to match the scenario.

Procedure

1. On the workspace, double-click **tMysqlInput** to open its **Basic settings** view in the **Component** tab.

tMysqlInput_1

Basic settings

Property Type: Built-In

DB Version: Mysql 5

☐ Use an existing connection

Host: localhost Port: 3306 Database: gettingstarted

Username: root Password: ****

Schema: Built-In Edit schema

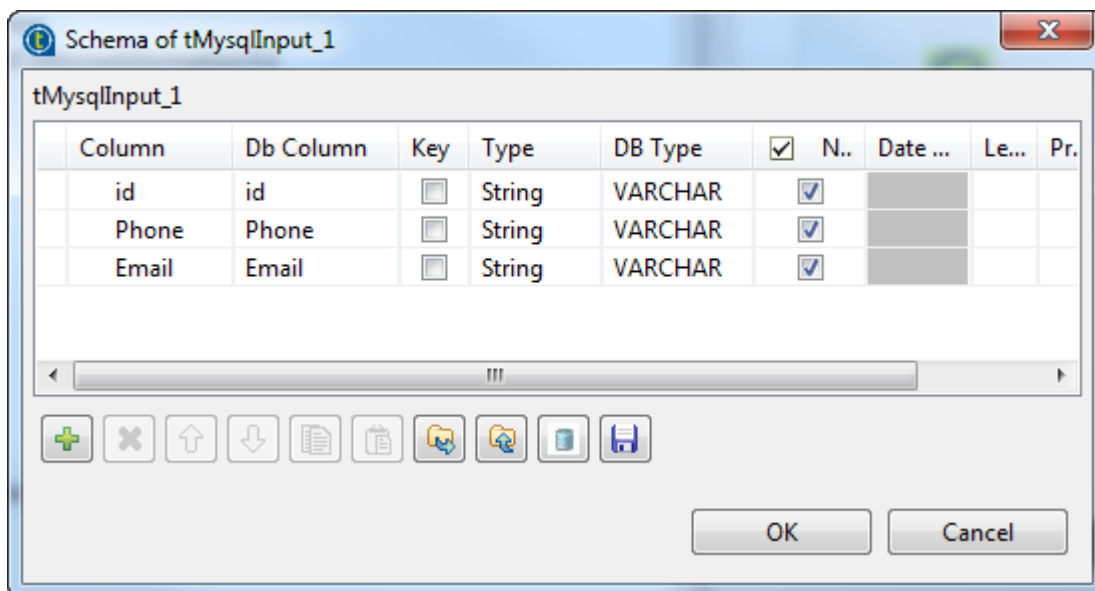
Table Name: customers

Query Type: Built-In Guess Query Guess schema

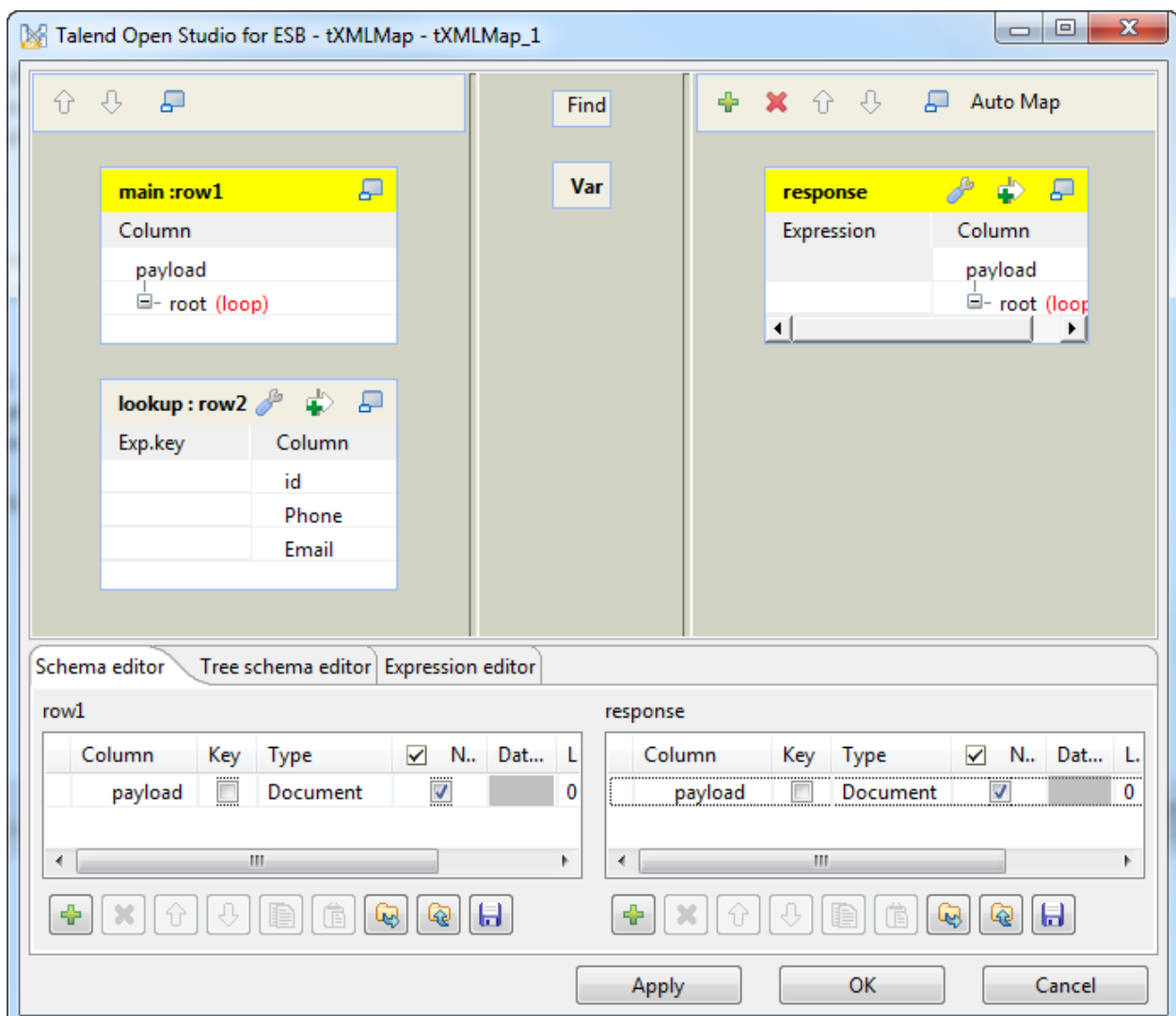
Query: `SELECT
customers.id,
customers.Phone,
customers.Email
FROM customers`

Data source
This option only applies when deploying and running in the Talend Runtime
☐ Specify a data source alias

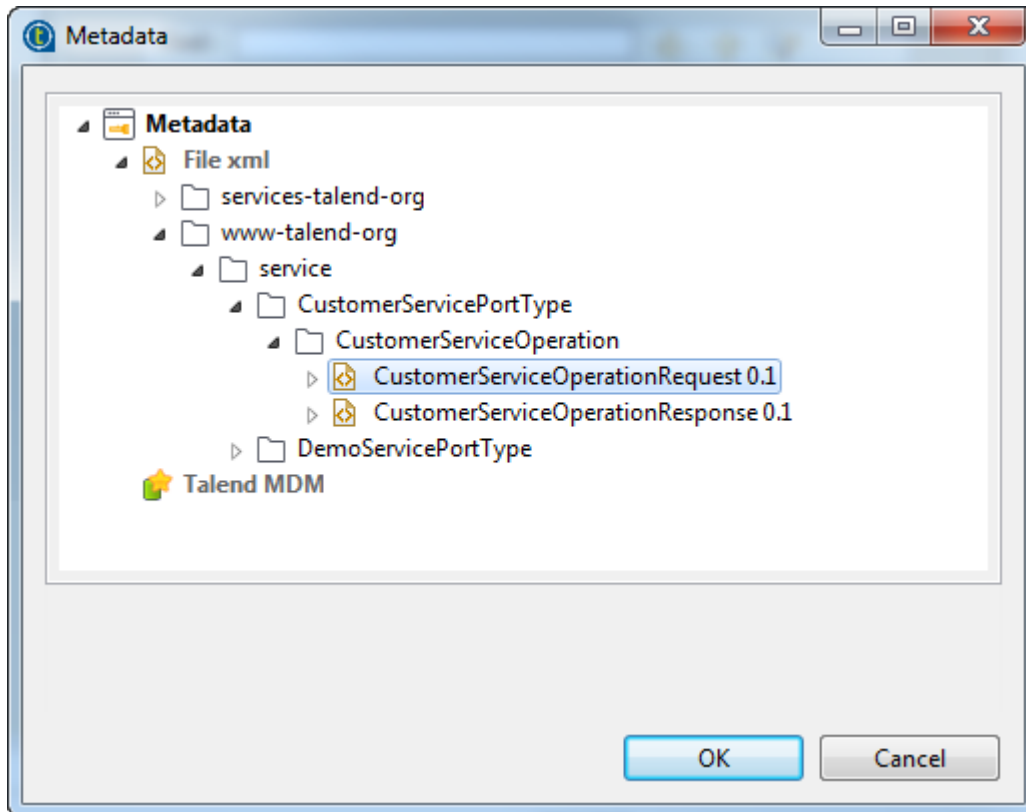
2. Configure the **tMysqlInput** to load the customer data in a MySQL database. In the **DB version** field, select the version of your MySQL database. It is **Mysql 5** in this example. Specify the connection details in the relevant fields, including:
 - the host name or IP address of your database server
 - the listening port number
 - the database name
 - the user name and password for your database authentication
3. Set the **Schema** as **Built-In** and click **Edit schema** to define the desired schema. The schema editor opens. Click the **[+]** button to add three rows of **String** type and give the name **id**, **Phone**, **Email** to the columns. Click **OK** to close the schema editor.



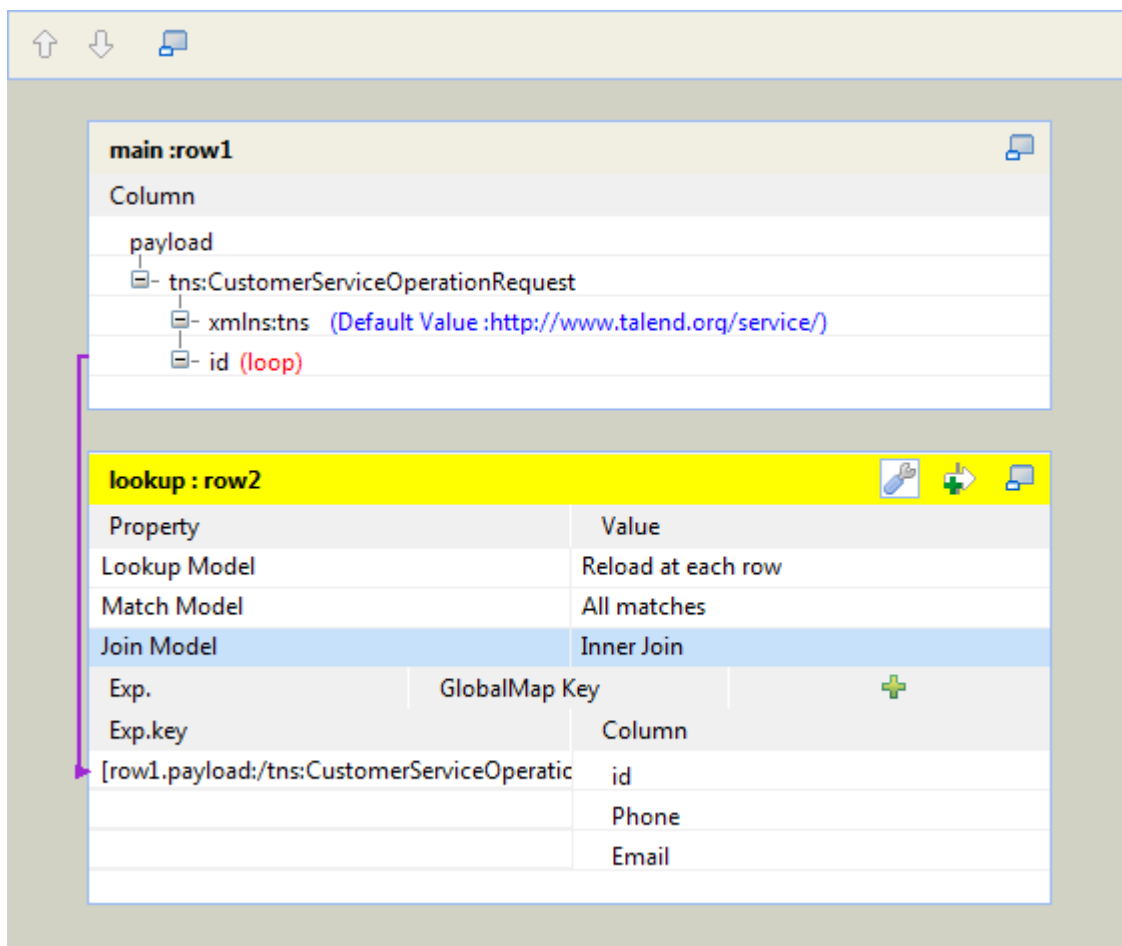
4. In the **Table Name** field, type in the name of the database table, `customers` in this case.
5. In the **Query** box, enter the query required to retrieve the desired columns from the table, `id, Phone, Email` in this example.
6. On the workspace, double-click **tXMLMap** to open its editor. At this moment, the editor should look like:



7. In the main row table of the input flow side (left), right-click the column name `payload` and from the contextual menu, select **Import from Repository**. Then the **Metadata** wizard is opened.
8. Expand the **File XML** node in this wizard, select the schema of the request side and click **OK** to validate this selection. In this example, the schema is `CustomerServiceOperationRequest`.



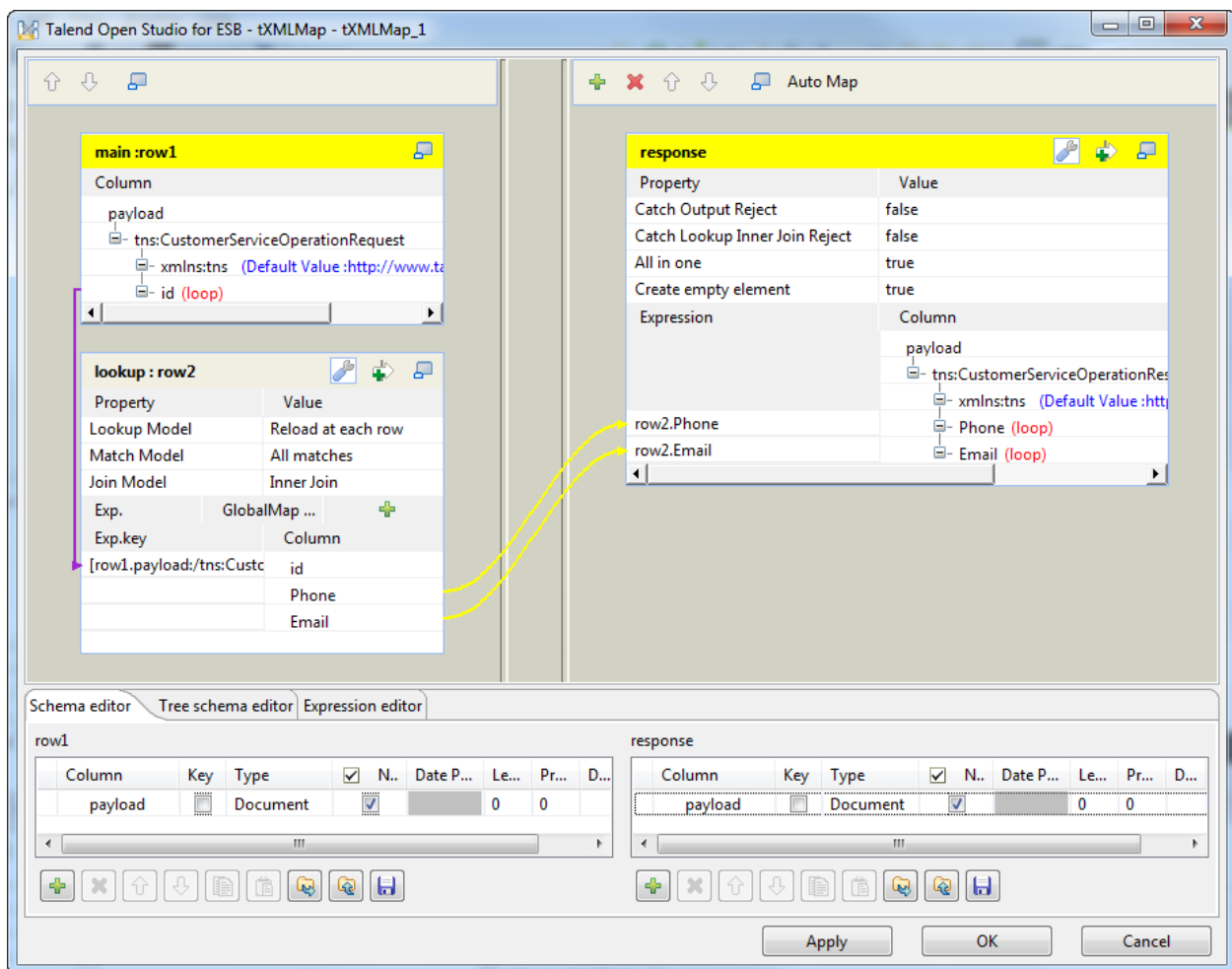
9. In the main row table of the output flow side (right), right-click the column name `payload` and from the contextual menu, select **Import from Repository**. In the **Metadata** wizard, select the schema `CustomerServiceOperationResponse` to import the hierarchical schema for the response.
10. To create the join to the lookup data, click the `id` node in the main row of the input side (left), hold and drop it onto the **Exp.key** column of the lookup flow, corresponding to the `id` row.
11. On the table representing the lookup flow, click the wrench icon on the up-right corner to open the settings panel. Set **Lookup Model** as **Reload at each row**, **Match Model** as **All matches** and **Join Model** as **Inner join**.



12. On the output table (right), click the wrench icon on the up-right corner to open the settings panel and set the **All in one** option as **true**. Right-click the `Email` node and select **As loop element** in the context menu.

13. Click the `Phone` row in the lookup flow (left), hold and drop it onto the **Expression** column corresponding to the `Phone` node in the XML tree view of the output flow. Do the same to map `Email` from the left side to the right side.

The **tXMLMap** editor should look like this:



14. Click **OK** to close the editor and validate this configuration.

Results

Now, the implementation of the **CustomerServiceOperation** is complete.

Executing the service operation in the Talend Studio

In this section, the Customer service is published to listen to all requests.

Press **F6** to run this Job. Once it is launched, the **Run** view is opened for you to read the execution result.


```

[statistics] connecting to socket on port 4043
[statistics] connected
Apr 22, 2016 5:27:07 PM
org.apache.cxf.wsdl.service.factory.ReflectionServiceFactoryBean
buildServiceFromWSDL
INFO: Creating Service
{http://www.talend.org/service/}CustomerService from WSDL:
D:/TOS_ESB-20160401_1937-V6.2.0SNAPSHOT/workspace/ESB_DEMOS/services
CustomerService_0.1.wsdl
Apr 22, 2016 5:27:08 PM org.apache.cxf.endpoint.ServerImpl
initDestination
INFO: Setting the server's publish address to be
http://localhost:8090/services/CustomerService
2016-04-22 17:27:08.231:INFO:oejs.Server:jetty-8.1.14.v20131031
2016-04-22 17:27:08.311:INFO:oejs.AbstractConnector:Started
SelectChannelConnector@localhost:8090
web service [endpoint:
http://localhost:8090/services/CustomerService] published

```

The service is now published, and will listen to all requests sent to the Web service until you click the **Kill** button to stop it since, by default, the **Keep listening** option in the **Basic settings** view of **tESBProviderRequest** is selected automatically.

The service will be called by a consumer that is built later.

Building a Customer consumer

In this section, you will see how to build a Customer consumer that will send requests to the Customer service that have been published.

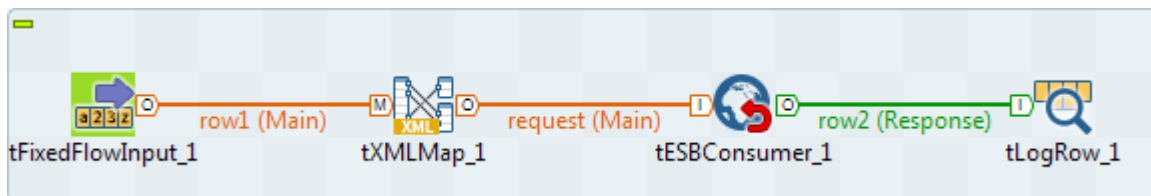
Creating a Customer consumer

To create the consumer Job, you need at least these components: an input component to read a data flow; a **tXMLMap** component that will map this flat data to a hierarchical document, the format expected by **ESB** components; the **tESBConsumer** components that will request the corresponding Web service and read its result; and the **tLogRow** component that displays the Job execution result. For this specific scenario, you will use a **tFixedFlowInput** as input component to send an id request to the **tESBConsumer** component.

Procedure

1. Right-click **Job Designs** in the **Repository** tree view and select **Create Job**.
2. In the dialog box that opens, only the first field (**Name**) is required. Type in **CustomerConsumer** and click **Finish**. An empty Job then opens on the main window and you can continue to create the Job.
3. Add a **tFixedFlowInput**, a **tXMLMap**, a **tESBConsumer** and a **tLogRow** component by typing its name on the design workspace.
4. To link the input components to the mapper, simply right-click **tFixedFlowInput**, hold and drop it to **tXMLMap**.
5. To link **tXMLMap** to **tESBConsumer**, right-click **tXMLMap**, hold and drag to **tESBConsumer**. In the pop-up window that opens, type in the name you want to give to the output row link (**request**, for example) and then accept the propagation that prompts you to get the schema from **tESBConsumer**.
6. Link the **tESBConsumer** component to the **tLogRow** with a **Response** row link.

The data service consumer Job should look like this:



Configuring the Customer consumer

In this section, the components in the Customer consumer Job is configured.

Procedure

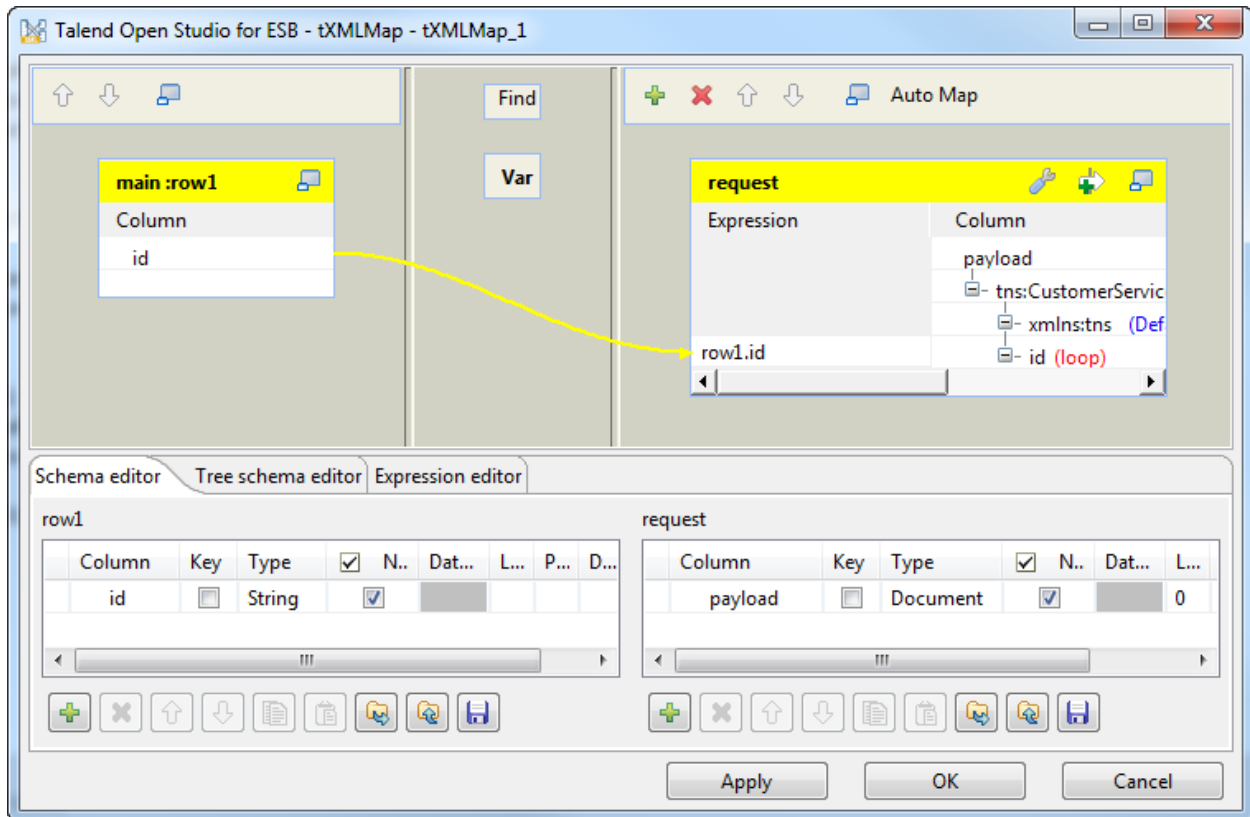
1. Double-click **tFixedFlowInput** to open its **Component** view.

Column	Value
id	"100"

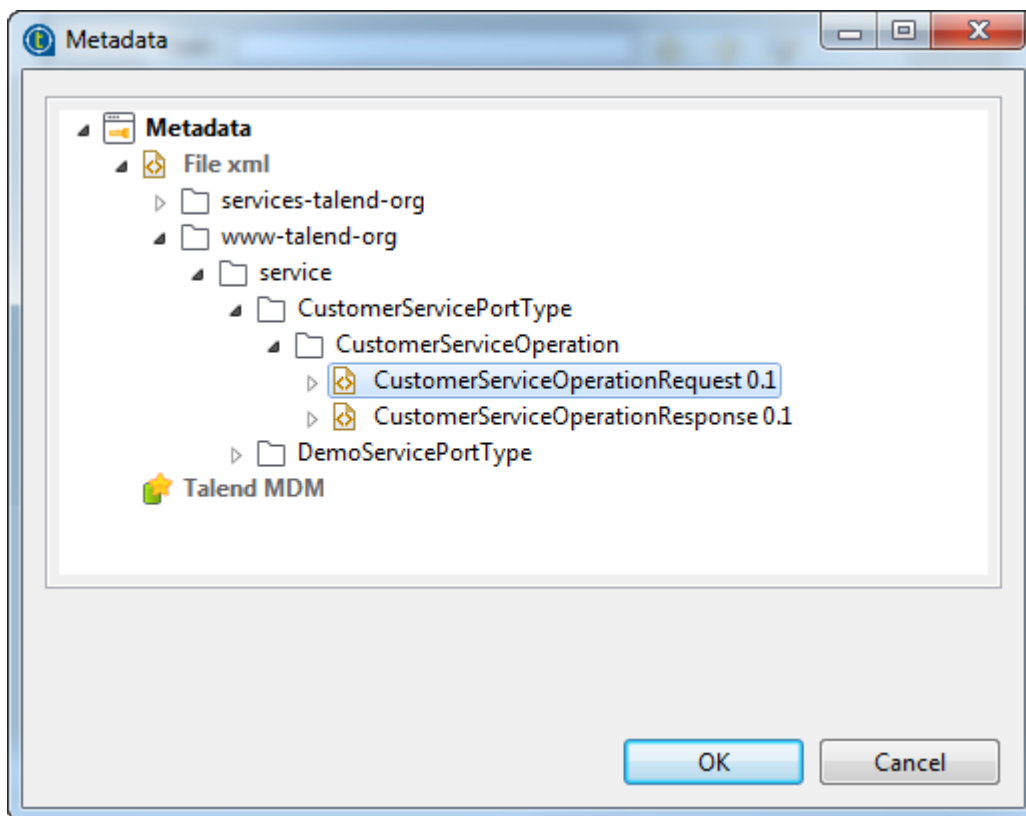
2. Click the three-dot button next to the **Edit schema** field to open the **Schema** window. Click the plus button once to add one **Column** to the schema and name it **id**. Keep the **Type** field as **string**. Click **OK** to validate this schema.

Column	Key	Type	N.	Date Patte...	Len...	Prec...	De...	Co...
id	<input type="checkbox"/>	String	<input checked="" type="checkbox"/>					

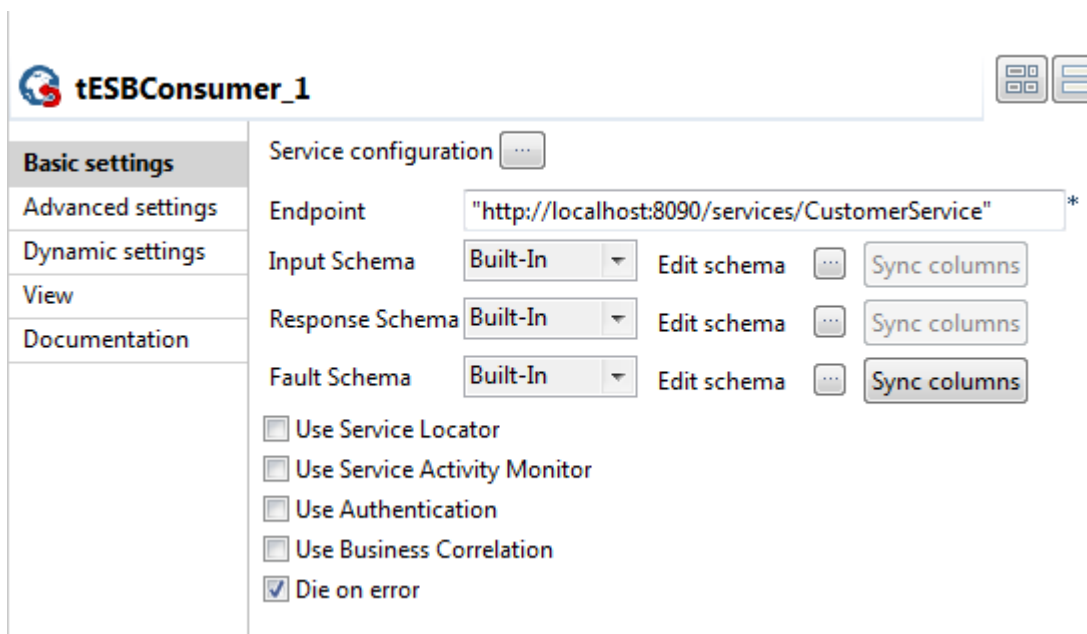
3. In the **Mode** area of the **tFixedFlowInput** basic settings, the active option should be **Use Single Table** and the `id` row is already added automatically to the **Values** table. In the **Value** column of the **Values** table, type in `100` within quotation marks.
4. Double-click **tXMLMap** to open its **Map editor**.



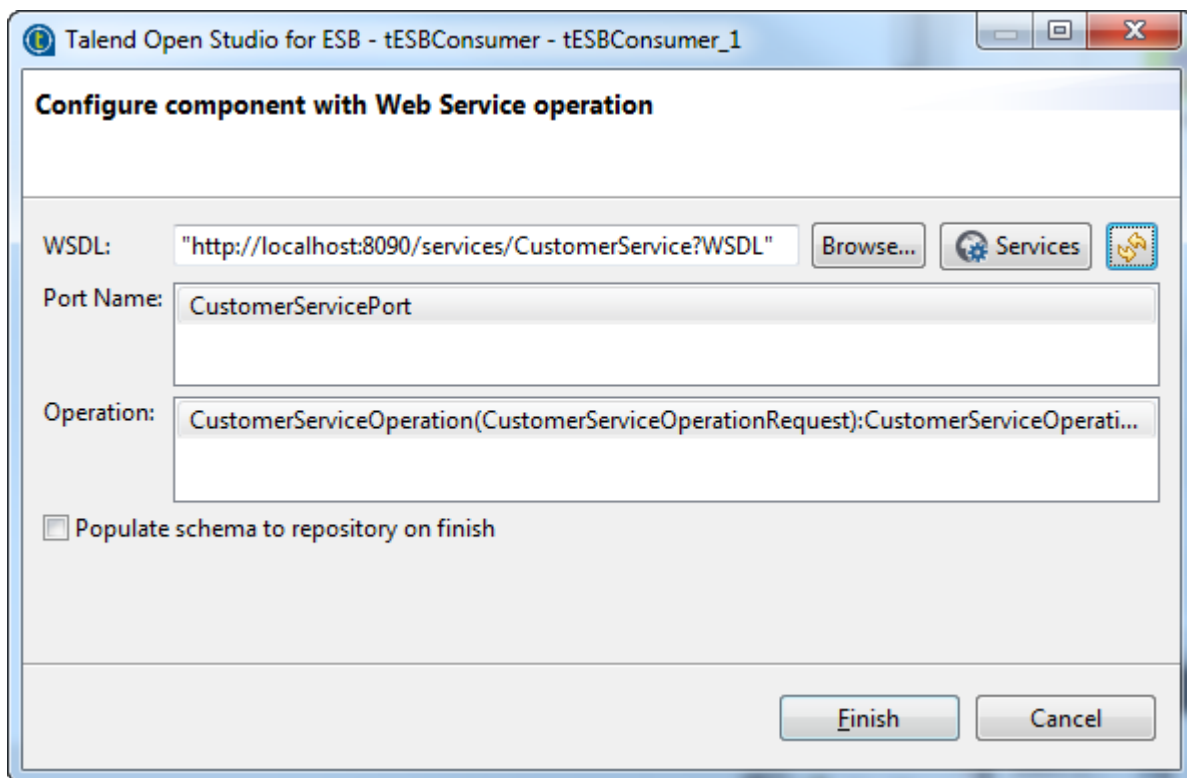
5. In the main row table of the output flow side (right), right-click the column name `payload` and from the contextual menu, select **Import from Repository**. Then the **Metadatas** wizard opens. Expand the **File XML** node in this wizard, select the schema of the request side and click **OK** to validate this selection. In this example, the request schema is `CustomerServiceOperationRequest`.



6. Click **id** in the main flow table of the input side (left), hold **and** drop it to the **Expression** column corresponding to the **id** node in the XML tree of the request table on the output side (right). Click **OK** to validate this configuration.
7. Double-click the **tESBConsumer** component to open its **Component** view.



8. Click the [...] button next to the **Service configuration** field to open the WSDL editor, paste the service `"http://localhost:8090/services/CustomerService?WSDL"` in the **WSDL** field and click the refresh button to the right to load the information, and then click **Finish**.



Results

The **tLogRow** component will automatically retrieve the schema from the previous component. If not, double-click it and click the **Sync columns** button in its **Component** view.

Executing the Customer consumer

In this section, the Customer consumer Job is executed.

To execute this Job, press **F6**.

Once done, the **Run** view is opened automatically, where you can check the execution result.

```
[statistics] connecting to socket on port 3701
[statistics] connected
<?xml version="1.0" encoding="UTF-8"?>
<tns:CustomerServiceOperationResponse
xmlns:tns="http://www.talend.org/service/"><Phone>504-710-5840</Phone>
<Email>arlene_klusman@gmail</Email></tns:CustomerServiceOperationRespo
nse>
[statistics] disconnected
```

This Job sends one id request to the consumer that requests the Web service through the provider Job. The user's phone number and email address are retrieved by the **tESBProviderResponse** and **tESBConsumer** components at the same time.

Note that although the provider Job received some requests, it did not stop and is still listening to new requests.

Running the service

Once the service is built, you can run it in an OSGi container, the Talend ESB Container. In this scenario, you will learn:

- How to export the service and run it in a Talend ESB Container for development purposes. See [Exporting the service and running it in a Talend Runtime Container](#) on page 30 for details.

Exporting the service and running it in a Talend Runtime Container

In this section, you will see how to export the Customer service to run it in an OSGi container, the Talend Runtime Container, for development purposes.

Before exporting the service, first start a Talend Runtime Container, and make sure all its Infrastructure Services have been started. For more information, see [Launching Talend Runtime and its Infrastructure Services](#) on page 11.

Procedure

1. Under **Services**, right-click `CustomerService 0.1` and select **Export Service**. In the **Save As** window, specify a folder. Click **Finish**.

This process builds and exports the service to the specified directory, as a `CustomerService-0.1.kar` file.

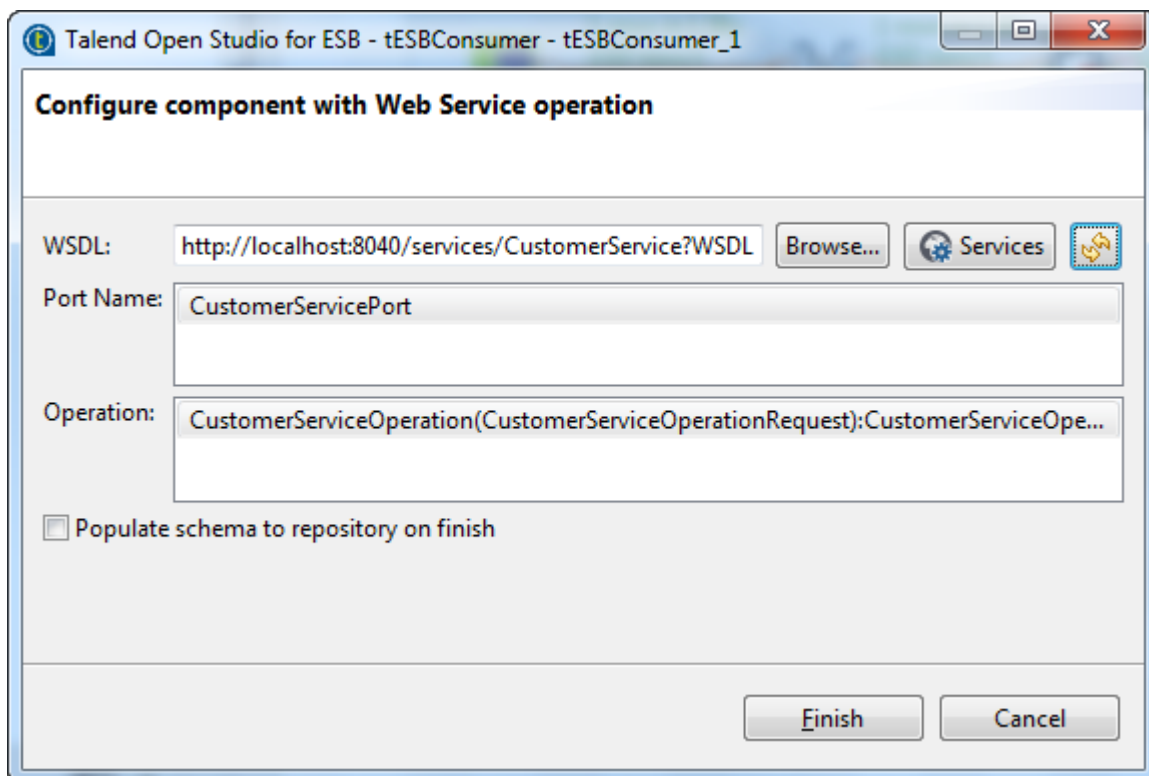
2. Copy the `CustomerService-0.1.kar` file, and paste it into the deploy folder of the Talend Runtime Container. The service starts directly.
3. Since this is a dynamic loading environment, the service starts running automatically. To see it in the Talend Runtime Container window, type in the `list` command at the console prompt.

```

247 | Active | 80 | 2.4.0.5 | Apache ServiceMix :: Bundles :: oscache
249 | Active | 80 | 6.2.0.SNAPSHOT | Talend ESB :: SAM :: Server
250 | Active | 80 | 6.2.0.SNAPSHOT | Talend ESB :: SAM :: Service Common
251 | Active | 80 | 6.2.0.SNAPSHOT | Talend ESB :: SAM :: REST Service
252 | Active | 80 | 6.2.0.SNAPSHOT | Talend ESB :: SAM :: SOAP Service
253 | Active | 80 | 6.2.0.SNAPSHOT | Talend ESB :: Locator :: Server
254 | Active | 80 | 6.2.0.SNAPSHOT | Talend ESB :: Auxiliary Storage :: REST Service
255 | Active | 80 | 2.0 | Content Repository for JavaTM Technology API
256 | Active | 80 | 2.2.0 | Commons IO
257 | Active | 80 | 2.10.1 | Apache Jackrabbit API
258 | Active | 80 | 2.10.1 | Jackrabbit JCR Commons
259 | Active | 80 | 2.10.1 | Jackrabbit SPI
260 | Active | 80 | 2.10.1 | Jackrabbit SPI Commons
261 | Active | 80 | 1.3.0 | Apache Tika core
262 | Active | 80 | 6.2.0.SNAPSHOT | Talend ESB :: Auxiliary Storage :: Persistence
263 | Active | 80 | 6.2.0.SNAPSHOT | Talend ESB :: Auxiliary Storage :: Persistence File
264 | Active | 80 | 6.2.0.SNAPSHOT | Talend ESB :: Auxiliary Storage :: Persistence JCR
265 | Active | 80 | 6.2.0.SNAPSHOT | Talend ESB :: Auxiliary Storage :: Server
266 | Active | 80 | 0 | wrap_file_D_TESB_SE-U6.2.0-SNAPSHOT_container_syste
m_concurrent_concurrent_1.3.3_concurrent-1.3.3.jar
267 | Active | 80 | 0.1 | CustomerService-control-bundle
268 | Active | 80 | 0.1 | CustomerServicePortType_CustomerServiceOperation
karaf@trun(>>)

```

4. To check if the service has correctly been deployed, go to `http://localhost:8040/services`, and the `CustomerService` service will be listed.
5. Now check it is working, by starting the consumer. The port that the service is running at has changed, and now uses the Talend Runtime Container port, which is by default 8040. To update the port, under **Job Designs**, open the **CustomerConsumer 0.1** Job. Click the middle of **tESBConsumer_1**. Then go to the **Component** tab.
6. Click the **[...]** button next to **Service configuration**, which opens a WSDL settings window.
7. Update the port number to use the Talend Runtime Container port by changing 8090 to 8040, and click the refresh button.



8. Now run the consumer job as before from the **Run** tab, and you see the same output as before.

What's next?

You have seen how Talend Studio helps you to build Services, implement the Service using a Job that listens to all requests, and to consume the Service using a Job. You have learned how to deploy it in the Talend ESB Container. Along the way, you have learned how to centralize frequently used connections in the **Repository** and easily reuse these connections in your Jobs.

To learn more about Talend Studio, see:

- Talend Studio User Guide
- Talend components documentation

To ensure that your data is clean, you can try Talend Open Studio for Data Quality and Talend Data Preparation Free Desktop.

To learn more about Talend products and solutions, visit www.talend.com.