Table of Contents

Introduction.........................................................................................................................2
About the author..................................................................................................................2
Download, install and run....................................................................................................2
The first project....................................................................................................................3
   Set up a new project.........................................................................................................3
   Create a new job...............................................................................................................4
   Execute the job..............................................................................................................7
Components, data flow and metadata..................................................................................8
Most common components..................................................................................................11
Repository........................................................................................................................12
Working with SQL databases..............................................................................................14
   Database installation.......................................................................................................14
   Database connection......................................................................................................15
   Example 1: Create a new table......................................................................................17
   Example 2: Subjobs.........................................................................................................22
   Example 3: Clone and synchronize a table....................................................................27
   Example 4: SQL queries and record deletion.................................................................29
   Example 5: Join and map...............................................................................................31
   Using other SQL and No-SQL database vendors............................................................34
Context variables................................................................................................................35
   Context groups...............................................................................................................38
   Database connection parameters....................................................................................40
Deployment........................................................................................................................42
Sub Jobs................................................................................................................................44
FAQ.....................................................................................................................................45
Introduction
This is a short guide about Talend Open Studio for Data Integration, one of the most popular and comprehensive tool for Data Integration. This guide can be freely downloaded and shared under the conditions of the Creative Common License [http://creativecommons.org/licenses/by-nc-sa/3.0/](http://creativecommons.org/licenses/by-nc-sa/3.0/).

Data Integration involves a collection of tools and techniques which aim to process information stored in different sources to create a coherent, clean, unified view of data.

Talend provides specific tools for data integration and data quality, avoiding the complicated development of custom procedures with traditional programming languages.

Talend is Open Source and freely downloadable, however the procedures developed by the final user do not need to be redistributed under an Open Source license.

About the Author
Roberto Marchetto is a certified independent consultant with years of expertise in Data Integration projects. He worked as a Java / Talend software engineer for important London based companies like Google, Imperial College, Publicis. He also provides coaching and tutoring. For more information see [www.robertomarchetto.com](http://www.robertomarchetto.com).

Download, install and run
Talend Open Studio for Data Integration is freely available and can be downloaded at [http://www.talend.com](http://www.talend.com). The version used in this guide is the Free Open Source version 5.5, however the same concepts apply to more recent versions and to the Enterprise edition as well.

Talend Open Studio is developed in the Java programming language and runs on Windows, Mac and Linux. The only prerequisite is the Oracle Java JRE or JDK, freely downloadable from the Oracle web site (both the Java SE and Java EE versions are fine, Java SE JRE is the minimum version required to run Talend).

Once Talend and Java JRE or JDK has been installed, Talend can be executed by one of the following files, depending on the platform:

- In Windows
  TOS_Di-win-x86_64.exe (64 bit) or TOS_Di-win32-x86.exe (32 bit)

- In Mac and Linux
  TOS_Di-linux-gtk-x86_64 (64 bit) or TOS_Di-linux-gtk-x86 (32 bit)
The first project

Set up a new project

During the first execution Talend Open Studio prompts a form as in Illustration 1 and afterwards a form as in Illustration 2. Basically Talend requires the user to open an existing project or to create a new one.

![Illustration 1: Talend first execution form](image1)

Press “Create...” and in the new form set Project Name to, for example, “TalendTutorial” and then click Finish. In the following form just select the “TalendTutorial” project and press Open.

![Illustration 2: Project opening](image2)

Talend Open Studio includes a useful collection of examples, which can be imported by clicking “Demo Project...”. In this guide we will use a new empty project however the user can import and explore the demo project as a practical reference.

While opening the project Talend shows a registration form. Even if this step is
optional the user registration gives access to the Talend support forum and Talend exchange for new components.

The loading of the application takes few seconds, if a Welcome page appears then click on the cross icon near the Welcome tab on the top, this will close the page and show the designer.

**Create a new Job**

Talend data integration projects are organised in Jobs. To create a new Job right click the Job Designs repository and press Create job, as in Illustration 3.

![Illustration 3: Creation of a new job](image)

On the form that will appear fill the Name field with “FirstTalendJob” and press Finish. All the remaining fields are optional. In a real world project however it is useful to give more information which will be used by Talend to improve the project documentation.

Now we are ready to develop our first job. For example we can generate some rows of data and display the result. Press the left mouse button on the tRowGenerator component which can be found in the Misc category of the Palette, as in Illustration 4.
Now drag and drop it in the workspace or just click an empty area of the workspace, the result should be similar to Illustration 5.

The new component needs to be configured, just double click on it and the editor in Illustration 6 will appear.
tRowGenerator generates a user defined collection of rows. Let’s suppose that we want to generate 100 rows of data with a sequence from 1 to 100. Just press the Add (+) button and fill the first row as follow:

- Column: id
- Type: Integer
- Function: Numeric.sequence
- Number of Rows for RowGenerator: 100

The result should be similar to Illustration 7 and to make sure the output is correct press the Preview button in the Preview tab.
Illustration 7: tRowGenerator with a numeric sequence

Press Ok and we can proceed with the next step. This component will generate 100 rows of data with a sequence from 1 to 100. We should do something meaningful with the generated data, for example display the result on console.

In the Palette click on the tLogRow component of the category Logs & Errors and drop it in the workspace. Then right click on the dropped tRowGenerator component and select Row → Main and click on tLogRow_1 as destination of the arrow.

Illustration 8: Link two components

A red arrow linking the two components should appear. Our first exercise is completed! In the next chapter we will run the job and display the result.

**Execute the job**

To run the job press the Run button on the Run tab, the result should be similar
to Illustration 9. The black text which appears is the output that this job produces in console, this is because we are using a tLogRow component, otherwise the job would not produce any output in console.

Components, data flow and metadata

To use Talend Open Studio effectively is it necessary to know three main concepts: Components, data flow and metadata.

Components are a simple yet powerful way to perform tasks. We can find components for any operation, for example row generator, data base connection, file system tools or even network utilities. To configure a component we just click on its icon and select the Component tab, as in...
In the designer components are linked each other with an arrow, this means that the Data flow runs from a component to the next linked component row by row. In our example the tRowGenerater component generates 100 rows and sends them to the tLogRow component, which function is to display the result.

The operation of the linked components is repeated for each row of data, in our case each record is logged on the console.

We can also use other types of links, for example conditional links which execute the linked components only after some conditions are true. This will be explained in the following chapters.

The last important concept is meta data, which means how the data flow is structured. It is important to remember that a component requires the definition of the schema while processing the data flow. The data structure in our first example is a single column named id of type integer. If we click on the Component tab of tRowGenerator and then click on Edit schema we can see its data structure.

Illustration 10.

Illustration 10: The Component tab
The meta data definition is important to make sure that the flow content is always coherent and to avoid errors which would be hard to recognise.

The linked components must have compatible meta data, otherwise Talend will produce an error. When we change the meta data of any component we can press the “Sync columns” button of the linked components and automatically import the new data structure.
### Most common components

Now that we have an overview of the Talend basics we can take a look at how many components are already available. To search for a component by name we can use the search box in the Palette and press the search button.

![Palette search](image)

*Illustration 13: Palette search*

Some of the most commonly used components are listed in the following table.

<table>
<thead>
<tr>
<th>Repository category</th>
<th>Component</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Databases</td>
<td>Several components</td>
<td>A comprehensive collection of specific relational databases components</td>
</tr>
<tr>
<td>Data Quality</td>
<td>tUniqRow</td>
<td>Removes duplicated rows</td>
</tr>
<tr>
<td>ESB → Rest</td>
<td>tRESTClient</td>
<td>REST web services client</td>
</tr>
<tr>
<td>ESB → Web Services</td>
<td>tESBConsumer</td>
<td>Web Services client</td>
</tr>
<tr>
<td>File</td>
<td>Several components</td>
<td>Collection of file and directory related operations</td>
</tr>
<tr>
<td>Internet</td>
<td>Several components</td>
<td>Several components which allow to connect to several network protocols</td>
</tr>
<tr>
<td>Custom Code</td>
<td>tJavaRow</td>
<td>Executes a custom Java code for each row</td>
</tr>
<tr>
<td></td>
<td>tLibraryLoad</td>
<td>Load an external Java library which can be used in other components, for example tJavaRow</td>
</tr>
<tr>
<td>Logs &amp; Errors</td>
<td>tLogRow</td>
<td>Prints the content of a flow to console</td>
</tr>
<tr>
<td></td>
<td>tWarn</td>
<td>Sends a warning, error or info message which is captured by tLogCatcher</td>
</tr>
<tr>
<td></td>
<td>tLogCatcher</td>
<td>Captures warning, error and info messages dispatched by tWarn</td>
</tr>
<tr>
<td></td>
<td>tDie</td>
<td>Interrupts the current job execution</td>
</tr>
<tr>
<td>Misc</td>
<td>Note</td>
<td>Shows a text note in the workspace</td>
</tr>
<tr>
<td></td>
<td>tBufferedInput / tBufferedOutput</td>
<td>Save / reads the flow in a temporary buffer, useful in some complex workflows</td>
</tr>
<tr>
<td>Component</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>----------------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>tFixedFlowInput</td>
<td>Generates a sequence of fixed rows, useful for example to provide predefined input for a linked component</td>
<td></td>
</tr>
<tr>
<td>tRowGenerator</td>
<td>Generates a dynamic sequence of rows</td>
<td></td>
</tr>
<tr>
<td>Orchestration</td>
<td></td>
<td></td>
</tr>
<tr>
<td>tReplicate</td>
<td>Replicates a flow which can be redirected to two or more components at the same time</td>
<td></td>
</tr>
<tr>
<td>tUnite</td>
<td>Merges two or more flows in a single flow</td>
<td></td>
</tr>
<tr>
<td>Processing</td>
<td></td>
<td></td>
</tr>
<tr>
<td>tAggregateRow</td>
<td>Aggregates a flow by one or more columns, similar to a GROUP BY SQL clause</td>
<td></td>
</tr>
<tr>
<td>tFilterRow</td>
<td>Filters a flow</td>
<td></td>
</tr>
<tr>
<td>tJoin</td>
<td>Merges two flows providing look up functionalities</td>
<td></td>
</tr>
<tr>
<td>tMap</td>
<td>One of the most used multi purpose component which allow to map, join and process flows</td>
<td></td>
</tr>
<tr>
<td>tReplace</td>
<td>Replaces values in a flow, for example Y/N with Yes/No values</td>
<td></td>
</tr>
<tr>
<td>tSortRow</td>
<td>Sorts a flow by one or more column values</td>
<td></td>
</tr>
<tr>
<td>tXMLMap</td>
<td>XML version of the tMap component which produces xml compatible output</td>
<td></td>
</tr>
<tr>
<td>System</td>
<td></td>
<td></td>
</tr>
<tr>
<td>tRunJob</td>
<td>Executes a subjob</td>
<td></td>
</tr>
<tr>
<td>tSystem</td>
<td>Executes a console system command</td>
<td></td>
</tr>
<tr>
<td>XML</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Several components</td>
<td>XML manipulation related components</td>
<td></td>
</tr>
</tbody>
</table>

The list of components is long, these are just some of the most used ones. The user can also download new components from Talend Exchange or, with the required Java expertise, create new ones.

**Repository**

Big projects can lead to maintainability issues, this is why Talend provides a set of repositories which allow to share reusable configurations among the components. The repository is shown in Illustration 14 and several straightforward directories are available.
• **Business Model**
  Provides a diagramming tool useful for design and documentation.

• **Job Designs**
  Contains the jobs. The user can also create sub folders by clicking on the Job Designs item and selecting “Create folder”

• **Contexts**
  User defined parameters can be declared in this repository and shared among the jobs. For example a user can create a set of development and production parameters and choose which set to use at a specific moment.

• **Code**
Contains custom Java routines. Talend is based on Java and the user can create custom Java code to perform advanced tasks.

- **SQL Templates**
  A deeper level of control over the SQL statements for a specific database can be achieved by editing the items in this directory.

- **Metadata**
  The metadata repository is an important directory in which the user defines the data connections and schema. A set of useful tools and wizards helps the user during the data mapping.

- **Documentation**
  External documentation files can be saved here.

- **Recycle bin**
  The items once cancelled from the repository are not erased from the disk but are moved in this recycle bin. This feature is similar to the recycle bin of the operating system.

**Working with SQL databases**

Enterprise data is usually stored in databases, either relational or non relational. A common Data Integration task involves the extraction of data from a database, then the data will be transformed and finally loaded in another relational database. This recurring Data Integration workflow is a typical example of ETL (Extraction Transformation and Loading).

In this chapter we will work with a MySQL database. Talend provides several connector for different SQL databases vendors, the choice of MySQL is just for its simplicity and popularity.

**Database installation**

The following examples require a working MySQL database server. MySQL is freely downloadable at [www.mysql.com](http://www.mysql.com). For further information on how to install and use MySQL the user can find several tutorials and documentation in Internet.

Once MySQL has been installed, we need to create a new database, for example “talend_examples”. The simplest way is to open the command prompt and digit the following command:

```
  mysql -u root -h localhost -p
```

And enter the MySQL password when required:

```
Enter password: <digit MySQL password>
```

Once the connection has been established run the following command:

```
  mysql> create database talend_example;
```

If everything is fine now we have an empty “talend_example” database in
MySQL.

**Database connection**

The Repository tab provides an easy method to establish a database connection. Right click on DB connections in the Metadata item of the Repository tab, select Create connection and in the new form digit “LocalMySQL” as connection name.

Press Next and fill the form with the following parameters:

- Db version: MySQL5 (or MySQL4 if using the version 4 of MySQL)
- Login: The MySQL user name, for example root
- password: The MySQL user password
- Server: The host of the server, for example localhost
- Port: 3306 is the default value
- DataBase: The previously created database talend_example
- Additional parameters: Leave as default
Illustration 15: Set up of a SQL connection

Check the connection with the button Check, if everything works a successful message should appear. Now press Finish and the new connection will be listed in the Repository as in Illustration 16.
Example 1: Create a new table

In this example we will use the Talend database components to create two MySQL tables and insert some data.

Create a new project named for example “SQLDataInsertion” (right click on Job Designs → Create Job). Now drop a tFixedFlowInput component and in the Component tab click the Edit schema button.

Fill the Schema with the values of the following table (also see Illustration 17):

<table>
<thead>
<tr>
<th>Column</th>
<th>Key</th>
<th>Nullable</th>
<th>Type</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>code</td>
<td>True</td>
<td>False</td>
<td>String</td>
<td>60</td>
</tr>
<tr>
<td>description</td>
<td>False</td>
<td>False</td>
<td>String</td>
<td>200</td>
</tr>
<tr>
<td>quantity</td>
<td>False</td>
<td>False</td>
<td>Integer</td>
<td></td>
</tr>
</tbody>
</table>
On the Component tab select “Use Inline Table” and fill the table with the following values (as in Illustration 18):

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
<th>Quantity</th>
</tr>
</thead>
<tbody>
<tr>
<td>“001”</td>
<td>“chocolate”</td>
<td>1</td>
</tr>
<tr>
<td>“002”</td>
<td>“cake”</td>
<td>10</td>
</tr>
<tr>
<td>“003”</td>
<td>“coffee”</td>
<td>3</td>
</tr>
</tbody>
</table>
Illustration 18: tFixedFlowInput data definition

This component generates a table with 3 rows. Now click the LocalMySQL connection from Metadata → Db connections of the repository (LocalMySQL is the MySQL connection previously created) and drop it on an empty Job area and select tMysqlOutput as component.
A new tMysqlOutput component should appear in the job workspace. Now right click on the tFixedFlowInput component in the workspace, select Row → Main and connect the flow to the tMysqlOutput component.

The connection parameters are already copied from the repository to the new tMysqlOutput. As you can see in Illustration 20 the Property Type is also set to Repository and it is using the LocalMySQL connection.
The user can set specific parameters for the component instead of the Repository one, this can be achieved by setting the Property Type to Built-in.

In addition the simple action of connecting the two components populates the tMysqlOutput schema, as you can see by clicking the Edit schema button.

Once again it is important to maintain a compatible schema between two linked components. The input schema of a component should always match the output schema of the following component, otherwise Talend will raise an error. The Sync columns button of the tMysqlOutput automatically updates the
schema in case of changes.

Finally we need to set the following values in the tMysqlOutput Component tab:

- Table: “inventory”
- Action on table: “Create if does not exist”
- Action on data: “Insert or update”

The “Create if does not exist” action is rarely used in real World projects, usually the database tables are already created, however in our example the database is empty and we are using Talend to create the new tables.

With the Action “Insert or update” Talend tries to insert the record and, if it is already present (by checking the key values), it will try to update the record.

To run the job just open the Run tab and press Run. In case of errors check the connection and make sure that the talend_example database is created and the user can create tables. In the Illustration 22 there is the output of a typical connection error:

![Illustration 22: MySQL connection error](image)

If this does not fix the error make sure that the Length field of the String columns of the tMysqlOutput schema (press Edit schema) is set, otherwise Talend will not be able to create a MySQL table. Also try to change the Action on table to “Drop table if exists and create”, this will delete any previous table.

If everything is fine the new table will be created and populated with the data of the tFixedFlowInput component.

**Example 2: Subjobs**

In this example we will extend the previous job with a sub job which creates an
“inventory” table. Keep the SQLDataInsertion project open and select both the tFixedFlowInput and tMysqlOutput components (click on them with the Ctrl button pressed). Then right click on one of the two components, select Copy and then paste in an empty area of the workspace.

Illustration 23: Copy of a workflow

The two “LocalMySQL” labels can be confusing, however we can rename these components in a more meaningful way. Click on the first LocalMySQL component, select the View section of the Component tab and update the field “Label format” to “create_inventory_table” as in Illustration 24. Also set the second LocalMySQL component label to “create_products_table”.

Page 23 - www.robertomarchetto.com
Now update the tFixedFlowInput_2 component with the following schema (just press on Edit schema of the component tab and remove the “quantity” column as in Illustration 25):

<table>
<thead>
<tr>
<th>Column</th>
<th>Key</th>
<th>Nullable</th>
<th>Type</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>code</td>
<td>True</td>
<td>False</td>
<td>String</td>
<td>60</td>
</tr>
<tr>
<td>description</td>
<td>False</td>
<td>False</td>
<td>String</td>
<td>200</td>
</tr>
</tbody>
</table>
After pressing Ok a dialog will appear to propagate the changes to the next component. Press Yes, this will synchronize the tMysqlOutput_2 component.

Now update the Inline Table of the tFixedFlowInput row with the following values:

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“001”</td>
<td>“belgian chocolate”</td>
</tr>
<tr>
<td>“002”</td>
<td>“cheesecake”</td>
</tr>
<tr>
<td>“003”</td>
<td>“black coffee”</td>
</tr>
</tbody>
</table>

Click on the tMysqlOutput_2 component and set the Table property to “products”. Also make sure that the Action on table is set to “Create table if does not exist”.
Once again the action “Create table if does not exist” is rarely used in real world projects but it is necessary in these examples.

With this configuration tFixedFlowInput_1 and tFixedFlowInput_2 will run in parallel and create two separated tables “inventory” and “products”.

If we want to execute the tFixedFlowInput_2 component only after the completion of tFixedFlowInput_1 we need to right click the “create inventory_table” component and select Trigger → On Component Ok and link the line to the tFixedFlowInput_2. The result should be similar to Illustration 28.

Illustration 28: OnComponentOk trigger

Basically we have a project with two processes linked in sequence, the first creates a inventory table and the second a products table. The second process starts only if the first process terminates successfully.

Now run the job and in case of errors check the connection, the schema and the table names. If the table schema was wrong try to change the Action on
table to “Drop table if exists and create”, this will delete any previous table.

If the error can not be fixed here is the full SQL script that can be used to create and populate both tables directly from command line.

```
DROP TABLE IF EXISTS inventory;
CREATE TABLE inventory (  
    code varchar(60) NOT NULL DEFAULT '',  
    description varchar(200) DEFAULT NULL,  
    quantity int(11) DEFAULT NULL,  
    PRIMARY KEY (code)  
);

INSERT INTO inventory VALUES ('001','chocolate',1),('002','cake',10),  
('003','coffee',3);

DROP TABLE IF EXISTS products;
CREATE TABLE products (  
    code varchar(60) NOT NULL DEFAULT '',  
    description varchar(200) DEFAULT NULL,  
    PRIMARY KEY (code)  
);

INSERT INTO products VALUES ('001','belgian chocolate'),('002','cheescake'),  
('003','black coffee');
```

**Example 3: Clone and synchronize a table**

In this example we will create and synchronize a copy of the products table. First create a new job called SQLDataUpdate. Now right click on the LocalMySQL connection in the Repository and and select Retrieve schema.

![Illustration 29: Database tables schema import utility](image)

Press Next and in the following form select both inventory and products table.

---

Page 27 - [www.robertomarchetto.com](http://www.robertomarchetto.com)
as in Illustration 30.

Illustration 30: Database tables selection

Press Next again and then Finish, the imported schema will appear in the Metadata repository.

Illustration 31: Table schema imported in the repository

Drag the products table in the empty workspace and select tMysqlInput. The tMysqlInput component will load the content of the whole table and send it to the flow.

From the Databases → MySQL palette drag a tMysqlOutput component in the workspace, then link the tMysqlInput component previously created with the tMysqlOutput component (right click on tMysqlInput, select Row → Main and drag the link to tMysqlOutput).

Now select the tMysqlOutput and fill the Component tab as follows:

- Property Type: Repository, click on the “..” and select LocalMySQL as value
To check if everything works drag a final tLogRow component linked to the tMysqlOutput component (right on tMysqlOutput click, select Row → Main). The result should be similar to Illustration 32.

Now click Run in the Run tab and a new table named products_copy will be updated with the content of the products table.

It is important to remember that new and updated data is always propagated and synchronized however deleted records from the products table will not be deleted in the products_copy table. To remove deleted records we need to implement a separated flow, as in the following example.

**Example 4: SQL queries and record deletion**

Basically to delete records we use a tMysqlOutput component with Action on data set to “Delete” and an input flow which contains the keys to be deleted.

We can implement several approaches to extract the records deleted from the master table. Unfortunately there is not just a component or flag to do that, we need to create a separated flow.
In the previous SQLDataSynchronization job right click on the MySQL connection from the Db connections Repository and select Edit Queries. Fill the blank query area with the following statement:

```
select c.code
from products_copy as c
where c.code not in (select p.code from products as p)
```

In a real world project it is better to use a separated table which logs the deleted records rather than using a query like this, which scans the entire two tables.

The running man icon on the top of the query tab executes the query and it is useful to check if the query works. The output should be empty as we do not have records in the products category table which are not present in the products_copy table. However if we delete some records in the products table the result will appear on the Result tab.

![Illustration 33: SQL query for deleted records](image)

Press Ok and digit “products_deleted_records” as query name when required. The new SQL query appears in the Metadata repository.

Now drag the products_deleted_records in the workspace and select tMysqlInput as component. Also drag the products_copy table from the repository to the workspace and select tMysqlOutput as component.

Right click on the products_deleted_records component, select Row → Main and link the products_copy component. Also right click on the tLogRow component, select Trigger → On Component Ok and link the products_deleted_records
Finally set the Action on data value of the products_copy component to Delete, the result should be similar to Illustration 34.

Illustration 34: Job for records synchronization and deletion

Press Run on the Run tab to execute the job. If we delete some records from the products table the same records will be deleted from the products_copy table as well. Notice that to perform a record deletion just the key column “code” is required.

Example 5: Join and map

In this example we will update the description column of the inventory table with the values of products table.

In a new job “SQLDataJoinAndMap” drag the following components in the workspace (see Illustration 35):

- the inventory table from the Metadata repository and select tMysqlInput as component
- the products table from the repository and select tMysqlInput as component
- another inventory table from the repository but this time selecting a tMysqlOutput component
- A tMap component from the Processing Palette

Right click on the inventory tMysqlInput component, select Row → Main and link it to the tMap, then right click on the products tMysqlInput component, select
Row → Main and link it to the tMap. It is important to link the inventory component first and the products component afterwards.

Now right click on the tMap component and select Row → New output (Main), link it to the inventory tMysqlOutput component and when required digit “inventory_updates” as output name. The result should be similar to Illustration 35.

Illustration 35: Join flows with tMap

Double click on the tMap component and in the new window (see Illustration 36):

- Drag the column code from row1 to the first row of inventory_updates table
- Drag the column code from row1 to the first row of row2
- Drag the column description from row2 to the second row of inventory updates

The result should look like the Illustration 36:
By doing so we have linked the code column of the inventory table with the code column of the products table, an operation similar to a SQL join, and we are using code and description as output.

Press Ok and set the property Action on data of the tMysqlOutput inventory component to Update (see Illustration 37).

Illustration 36: Join and map values in the tMap editor
By running this job the description values of the inventory table will be updated with the description values of the products table. Notice that we do not need to send all the columns of the inventory table as output, we just need the code and description columns.

**Using other SQL and No-SQL database vendors**

In addition to MySQL Talend offers several specific connectors for various SQL databases like Oracle, Microsoft SQL Server, IBM DB2 etc. If the connector is not listed in the Palette → Databases section we can use the DB generic category which uses any compliant JDBC driver.

Each specific JDBC connector requires the vendor JDBC driver to be imported. Talend shows a prompt before using a component which driver has not been installed and guides the user through the installation process.

If the specific JDBC driver is still missing the user needs to download the vendor driver from Internet, copy the .jar file in `<Talend installation directory>/lib/java` and restart Talend.

Talend Opend Source for Data Integration also provides some Big Data component for Google storage and Hadoop Hive. Several specific components for Big Data (MongoDB, Hadoop, Cassandra, etc) are also available in the

**Context variables**

A Job execution can be parametrized, for example by using a set of variables for the production release and another set for testing. Either at run time or design time the user can choose the set of variables to be used.

To create context variables right click on Context in the Repository and select Create context group and in the new form digit as name for example “Parameters”. Press Finish, the context “Parameters” will appear in the Repository.

Now double click on the “Parameters” context and click Next. In this form we can declare the custom parameters, for example a “message_title” and “message_content” as in Illustration 38.

![Illustration 38: Context variables](image)

To assign the parameter values select Values as tree tab as in Illustration 39.
Once declared the context variables can be used in almost any of the Talend components by using the expression “context.<variable name>”. For example Illustration 40 shows a new job with a tMsgBox component with the values set as follow:

- Title: context.message_title
- Message: context.message_context
Before executing the job we need make sure that the context is imported by clicking the Context button in the Contexts tab of the project (see red circle in Illustration 41). We need to select all of the context variables we are going to use.
When we execute the job a message box with the assigned variable values should appear.

**Context groups**

The same job can be executed in different environments, for example with a configuration for development and another for production.

To manage different execution environments we can define a context group. To create a new context group open the context in the Contexts repository and in the Values as tree tab press the Configure Contexts button (top right of the form as in Illustration 42).
Select “New...” and give a name to the new context group, for example “Development”.

The selected context will be used by default while running the jobs. It is always possible to select another context at run time, as we will discuss in the chapter Deployment. The result should be similar to Illustration 44.
Database connection parameters

Context parameters are very useful while working with database connections. We can create a context variable for each of the database connection parameter and Talend provides a specific tool for that. While setting a connection in the Repository we can click on the “Export as context” button of the database connection form (see Illustration 45). Talend will generate a context variable for each connection parameter (see Illustration 46).
Illustration 45: Export as context button
Deployment

One of the big advantages of Talend is that procedures are compiled in Java, which makes them fast, reliable and portable. To deploy a procedure right click on the job and select “Build job” (in different versions of Talend the name can change).
The available parameters are:

- **To archive file**: Select the folder where to save the executable job.
- **Select the Job version**: Select 0.1. Talend allows to create new versions of the same job and archive the previous versions. The version number can be set during the job creation and should be selected in this dialog.
- **Select the build type**: Select “Standalone job”. A talend job can be executed in several environments however a common choice is the standalone executable (Standalone job). A job can also be run in an OSGi container or in an Axis web service.
- **Shell launcher**: Select true, this will provide the console scripts required to execute the job.
- **Context scripts**: Select true, this will export the selected context variable values in a file. In a production environment sensitive informations as password should not be saved in context but should be assigned at runtime instead (see at the end of this chapter).
- **Apply context to children jobs**: Select true, this will apply the selected
context to any project sub job.

- Java sources: Not required. This options deploys the Java source code along with the Java compiled code.

- Items: Not required. Exports the project workflow with all the components and settings. It can be useful for documentation propose or to import the project in another Talend environment.

By pressing Finish the executable program will be created in a zip archive.

To execute the project unzip the archive and execute the bash script (<job name>_run.bat in Windows or <job name>_run.sh in Linux and Mac), this script is saved in a folder called as the exported job name, for example FirstETLJob.

We can overwrite the context parameters and assign variables values at runtime with the following command line:

<job name>_run.<bat or sh> --context_param ParameterX=ValueX --context_param ParameterY=ValueY

The jobs can be programmatically executed via the system scheduler or via the more comprehensive Talend Enterprise version of the Data Integration tool.

Sub Jobs

An interesting feature, especially when a project starts growing, is the ability to split big jobs in smaller sub jobs.

The procedure is easy:

- Create a normal job
- Create a new empty job
- On the new empty job drag the job previously created from the Job Designs Repository

Talend will create a new component tRunJob which executes to the job dragged from the repository. Several tRunJobs can be execute sequentially by linking them with a On Component Ok trigger (right click on the dragged tRunJob, select Triger → On Component Ok).
Illustration 48: Sub jobs sequential execution

It is important to check “ Transmit the whole context” in the Component tab when a job requires context variables.

FAQ

How can I rename a link between two components? Once created, a link can not be renamed.

How can I rename the label of a component? Click on the component, select the Component tab and in the View section set “Label format” to the desired label.

How reliable is Talend? Talend is a leader in Data Integration tools. Talend offers enterprise services and support and the products are used in several mission critical environments.

What are the advantages of a Data Integration tool over custom developed procedures? The first advantage is the development time. A data integration procedure can be developed within days in Talend, while it can take several days or weeks with a traditional programming language. In addition all the features commonly required (logging, reliability, etc) are already available.

The second advantage is that Talend is easy to learn and several professionals are available in the market, this is very important for the project maintainability over the time.

How scalable is Talend Open Studio for Data Integration? Talend for Data Integration is a highly scalable solution and provides enterprise grade support and tools. However what makes a difference in big project is the design and approach rather than the specific tool. For example parallel execution, data partition and clustering are commonly used strategies while dealing with big amount of data.