




Smooks components

This chapter details the major components that you can find in **Smooks** group of the **Palette** of **Talend Open Studio**.

The Smooks component family groups components that covert specific data like EDI-message to the Talend data flow using Smooks mapping capabilities.



tSmooksInput Properties

Component family	Smooks	
Function	tSmooksInput reads and outputs multiple schema within a structured file (possibly non-XML).	
Purpose	tSmooksInput uses Smooks mapping capabilities to transform a complex multi-structured file to SAX event flow, reads its data and then uses Row links to send fields as defined in the different schemas to the next job components.	
Basic settings	Type of the input data	The type of the data to process.
	Input file	Name of the file to be processed. For example, EDI-message file name.
	Mapping file	Name of the Smooks mapping file. In case of EDI-message the mapping must comply xml scheme http://www.milyn.org/schema/edi-message-mapping-1.0.xsd
	Outputs	Schema: Define as many schemas as needed. The columns of schemas must have the same names as the xmltags defined in the mapping file. Loop Path: Node of the smooks output tree which the loop is based on.
Advanced settings	Encoding Type	Select the encoding type from the list or select Custom and define it manually. This field is compulsory.
Usage	Use this component to read a multi-schema file and separate fields using a Smooks mapping capabilities.	
Limitation/Prerequisites	n/a	

Scenario: Reading a multi structure EDI-message file

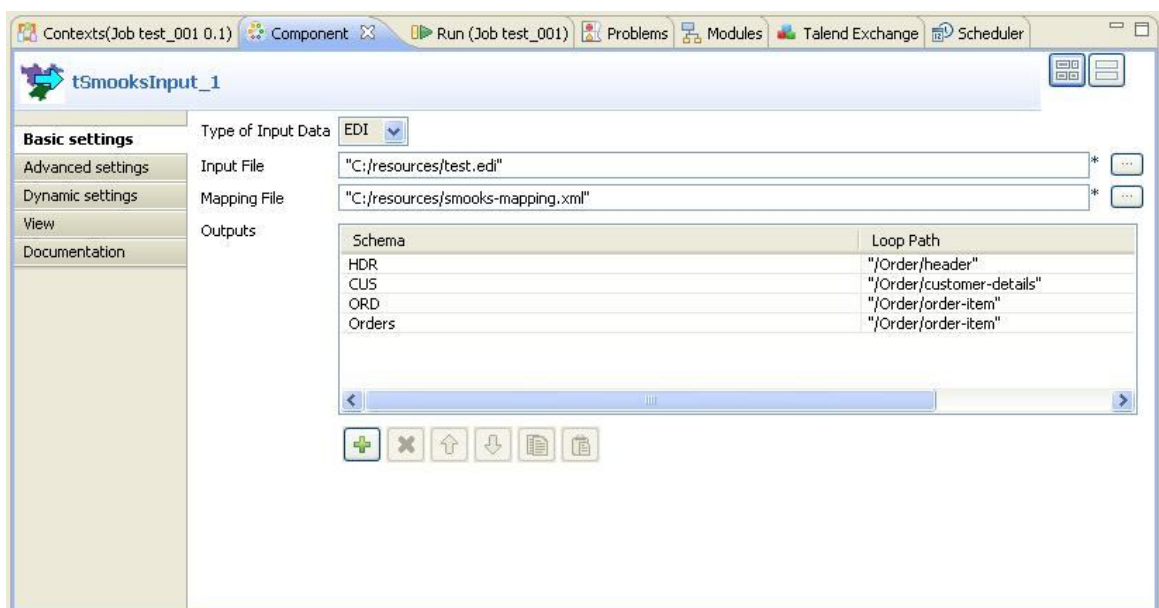
The following scenario creates a Java job which aims at reading a multi schema EDI-message file and displaying data structures on the **Run Job** console. The EDI file processed in this example looks like the following:

```
HDR*1*0*59.97*1,625.72*4.95*Wed Nov 15 13:45:28 EST 2006
CUS*user1*Harry^Fletcher*SD
ORD*1*1*364*The 40-Year-Old Virgin*29.98
ORD*2*1*299*Pulp Fiction*29.99
ORD*3*20*111*Robinson Crusoe*480
ORD*4*10*364*From Russia with Love*252.50
ORD*5*25*299*You Only Live Twice*631.25
ORD*6*1*115*The Man with the Golden Gun*25.25
ORD*7*7*116*A View to a Kill*176.75
```

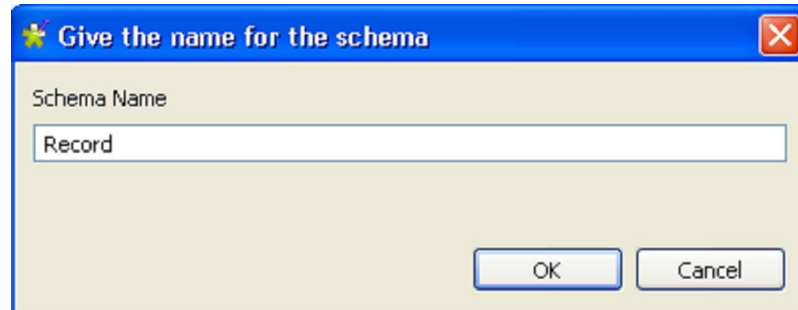
The Smooks mapping file for this EDI-message is:

```
- <medi:edimap xmlns:medi="http://www.milyn.org/schema/edi-message-mapping-1.0.xsd">
  <medi:description name="DVD Order" version="1.0" />
  <medi:delimiters segment="" field="*" component="^" sub-component="~" />
  - <medi:segments xmltag="Order">
    - <medi:segment segcode="HDR" xmltag="header">
      <medi:field xmltag="order_id" />
      <medi:field xmltag="status_code" />
      <medi:field xmltag="net_amount" />
      <medi:field xmltag="total_amount" />
      <medi:field xmltag="tax" />
      <medi:field xmltag="date" />
    </medi:segment>
    - <medi:segment segcode="CUS" xmltag="customer_details">
      <medi:field xmltag="username" />
      - <medi:field xmltag="name">
        <medi:component xmltag="firstname" />
        <medi:component xmltag="lastname" />
      </medi:field>
      <medi:field xmltag="state" />
    </medi:segment>
    - <medi:segment segcode="ORD" xmltag="order_item" maxOccurs="-1">
      <medi:field xmltag="position" />
      <medi:field xmltag="quantity" />
      <medi:field xmltag="product_id" />
      <medi:field xmltag="title" />
      <medi:field xmltag="price" />
    </medi:segment>
  </medi:segments>
</medi:edimap>
```

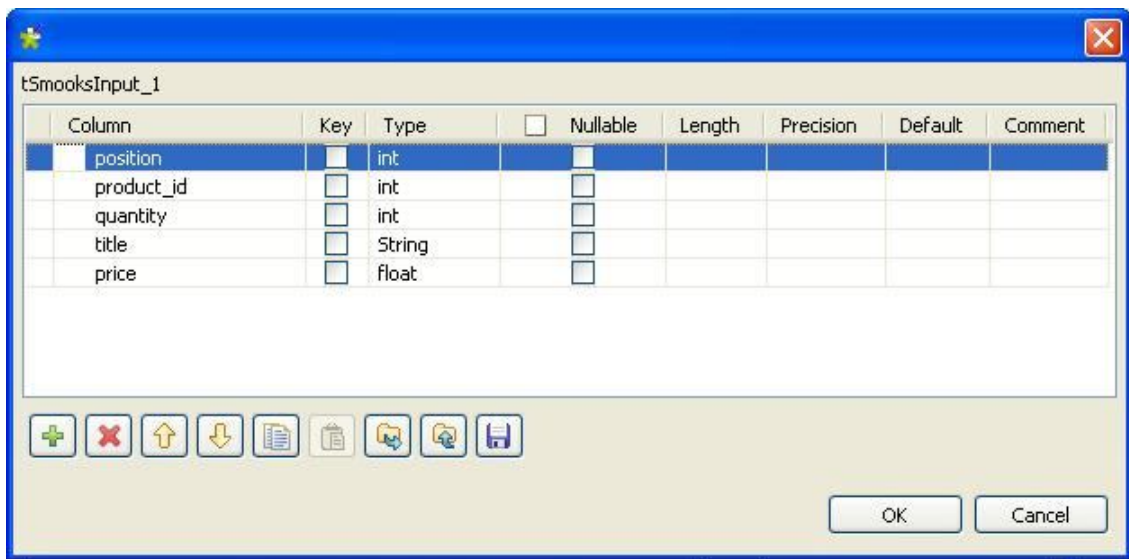
- Click and drop a **tSmooksInput** and four **tLogRow** components from the **Palette** onto the design workspace.
- Double click **tSmooksInput** to open the component **Basic settings** view.



- Identify the type of source data. For example, "EDI" means that the input data is EDI-message.
- Browse to the file you want to process. Specify the **Input File** field value.
- Browse to the mapping file and specify the **Mapping File** field value.
- Click the plus button to add lines in the **Outputs** table where you can define the output schema.
- In the **Outputs** table, click in the **Schema** cell and then click a three-dot button to display a dialog box where you can define the schema name.

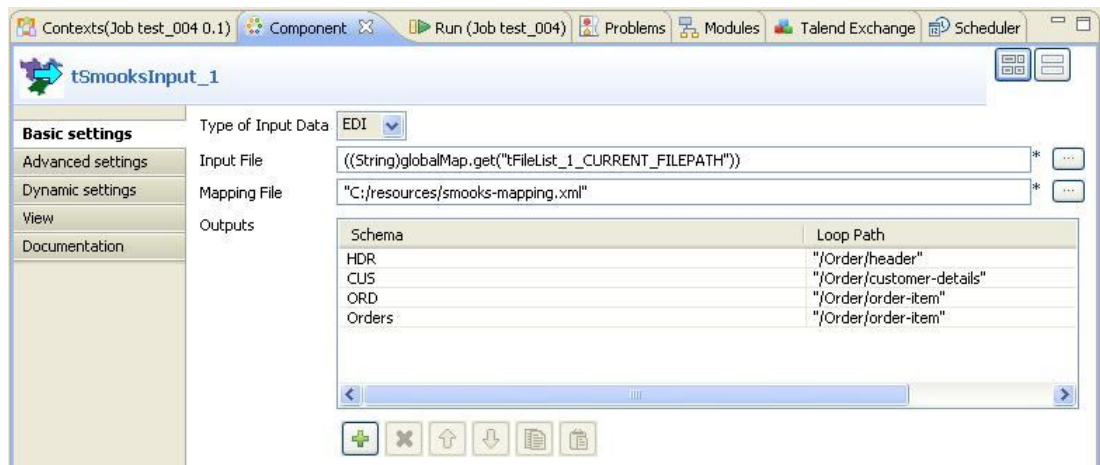


- Enter a name for the output schema and click **OK** to close the dialog box. The **tSmooksInput** schema editor displays.
- Define the schema you previously defined in the **Outputs** table. The fields that you specify must have the same names as the xmltag in mapping file.

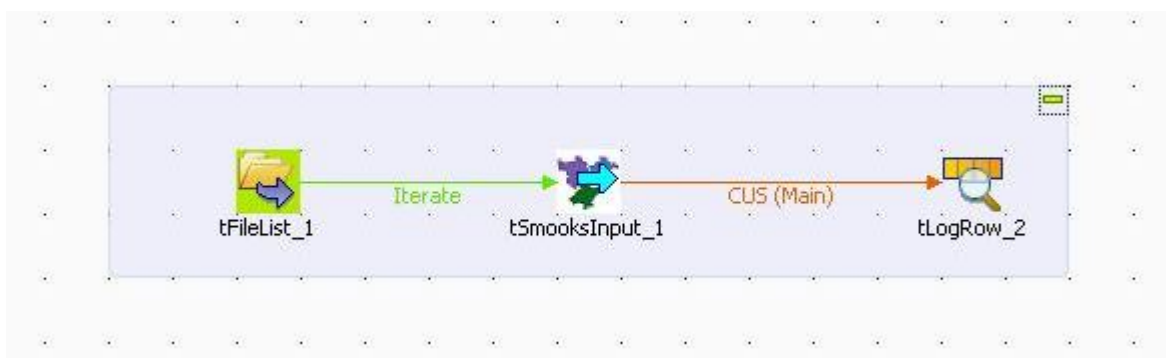


- Do the same for all the output schemas you want to define. In this scenario, we want to define four output schemas: *HDR*, *CUS*, *ORD* and *Orders*.

- Double click **tSmooksInput** to open the component **Basic settings** view.



- Identify the type of source data. For example, "EDI" means that the input data is EDI-message.
- Enter the **Input File** field using a variable containing the current filename path, as you filled in the Basic settings of **tFileList**. Press **Ctrl+Space bar** to access the autocomplete list of variables.
- Browse to the mapping file and specify the **Mapping File** field value.
- Click the plus button to add lines in the **Outputs** table where you can define the output schema. Fill in the table as described in the previous scenario.
- Select the last component, **tLogRow** and fill in the separator to be used to distinguish field content displayed on the Log. Related topic: tLogRow in Reference Guide of Talend Open Studio Component
- Right-click on the **tFileList** component, and pull an Iterate connection to the **tSmooksInput** component. Then pull a CUS row from the **tSmooksInput** to the **tLogRow** component.




The job iterates on the directory defined, and reads each file contained. Then delimited data is passed on to the last component which displays it on the Log.

```
Starting job test_004 at 13:01 02/10/2009.
```

```
user1|Harry|Fletcher  
user2|Taras|Shevchenko  
user3|Alexander|Pushkin  
Job test_004 ended at 13:01 02/10/2009. [exit code=0]
```



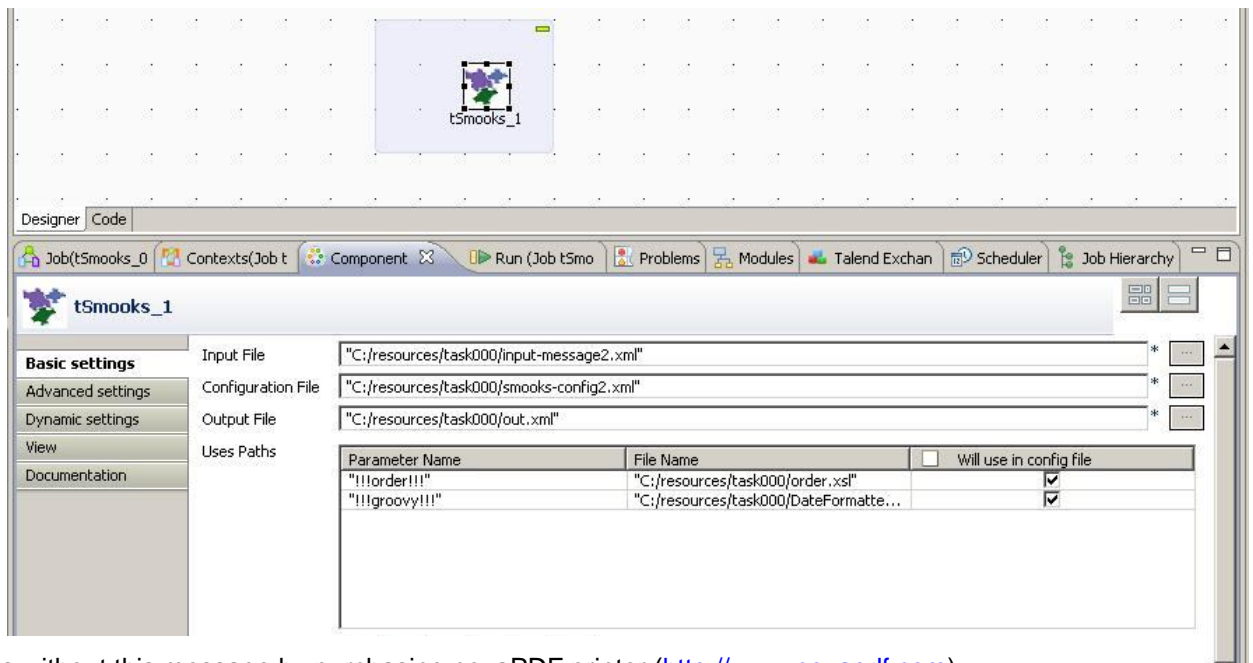
tSmooks Properties

Component family	Smooks	
Function	tSmooks implements the Smooks data transformation in TOS (EDI-to-XML, CSV-to-XML, XML-to-XML, XSLT-basic, XSLT-namespaces, XSLT-groovy)	
Purpose	tSmooks uses Smooks mapping capabilities to data transform, save the output data to the file and send the output file name to the Talend global Map. Next job components can use file name as parameter.	
Basic settings	Input file	Name of the file to be processed. For example, EDI-message or CSV file name.
	Configuration file	Name of the smooks configuration file.
	Output file	The name of the file that we receive after the data transformation.
	Uses	<p>Parameter name: Define as many parameters as needed. The name of the parameter you can use in a configuration file instead of specifying the name of external file.</p> <p>File name: The name of external file. (For example, the template or mapping file.)</p> <p>Will use in mapping file: If checked, the parameter will be processed in configuration handing.</p>
Usage	Use this component to implement the data transform by smooks capabilities in Talend Open Studio.	
Limitation/Prerequisites	n/a	

Scenario: Self-performance job (data transformation)

The following scenario creates a Java job using only tSmooks component. The component implements the Smooks data transformation XSLT-groovy in Talend.

- Click and drop a **tSmooks** component from the **Palette** onto the design workspace.
- Now define the Basic settings of **tSmooks**.



- Browse to the file you want to process. Specify the **Input File** field value. The input XML file processed in this example looks like the following:

```
<Order>
  - <header>
    <order-id>1</order-id>
    <status-code>0</status-code>
    <net-amount>59.97</net-amount>
    <total-amount>64.92</total-amount>
    <tax>4.95</tax>
    <date>15.11.2006 8:14 PM</date>
  </header>
  - <customer-details>
    <username>user1</username>
    - <name>
      <firstname>Harry</firstname>
      <lastname>Fletcher</lastname>
    </name>
    <state>South Dakota</state>
  </customer-details>
  - <order-item>
    <position>1</position>
    <quantity>1</quantity>
    <product-id>364</product-id>
    <title>The 40-Year-Old Virgin</title>
    <price>29.98</price>
  </order-item>
  - <order-item>
    <position>2</position>
    <quantity>1</quantity>
    <product-id>299</product-id>
    <title>Pulp Fiction</title>
    <price>29.99</price>
  </order-item>
</Order>
```

- Browse to the Smooks configuration file and specify the **Mapping File** field value. This example illustrates how Smooks can be used to combine Groovy scripting with XSLT to perform an Order message transform. In addition, we used the name of the parameter in a Smooks configuration file instead of specifying the name of Groovy script file (“**!!!groovy!!!**”) and XSL template file (“**!!!order!!!**”).

Thus, we have the following configuration file:

```
<?xml version="1.0" ?>
  - <smooks-resource-list xmlns="http://www.milyn.org/xsd/smooks-1.1.xsd"
    xmlns:xsl="http://www.milyn.org/xsd/smooks/xsl-1.1.xsd">
<!--
  Break out the <date> field into separate "time", "day", "month" and "year" fields
  using Groovy.
  This makes the complex date field value consumable by something like XSLT.
-->
  - <resource-config selector="header date">
    <resource>!!!groovy!!! </resource>
    <param name="input-format">dd.MM.yyyy hh:mm a </param>
    <param name="output-format">time=HH:mm
      day=dd
      month=MM
      year=yy</param>
  </resource-config>
```

```
<!--
```

Transform the document (as a whole) using XSLT. The complex date field has already been preprocessed into separate fields using Groovy (see above config) - XSLT can handle it easily now.

```
-->
```

```
- <xsl:xsl applyOnElement="$document">
  <xsl:template>!!!order!!!</xsl:template>
</xsl:xsl>
</smooks-resource-list>
```

- Specify the **Output File** field value.
- Define two parameters needed for the configuration file in our example. Parameter **!!!groovy!!!** specifies a file **DateFormatter.groovy** with a groovy script. Parameter **!!!order!!!** specifies a file **order.xsl** with a XSLT template. The file containing the template is as follows:

```
- <xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
  <xsl:output method="xml" encoding="UTF-8" />
  - <xsl:template match="Order">
    - <Order orderId="{header/order-id}"
      statusCode="{header/status-code}"
      netAmount="{header/net-amount}"
      totalAmount="{header/total-amount}"
      tax="{header/tax}"
      date="{header/date/month}-{header/date/day}-
{header/date/year}">
      <xsl:apply-templates select="customer-details" />
      - <OrderLines>
        <xsl:apply-templates select="order-item" />
      </OrderLines>
    </Order>
  </xsl:template>
  - <xsl:template match="customer-details">
    <Customer userName="{username}"
      firstName="{name/firstname}"
      lastName="{name/lastname}"
      state="{state}" />
  </xsl:template>
  - <xsl:template match="order-item">
    - <order-item quantity="{quantity}"
      product-id="{product-id}"
      price="{price}">
      <xsl:value-of select="title" />
    </order-item>
  </xsl:template>
</xsl:stylesheet>
```

- Save your job and press **F6** execute it.

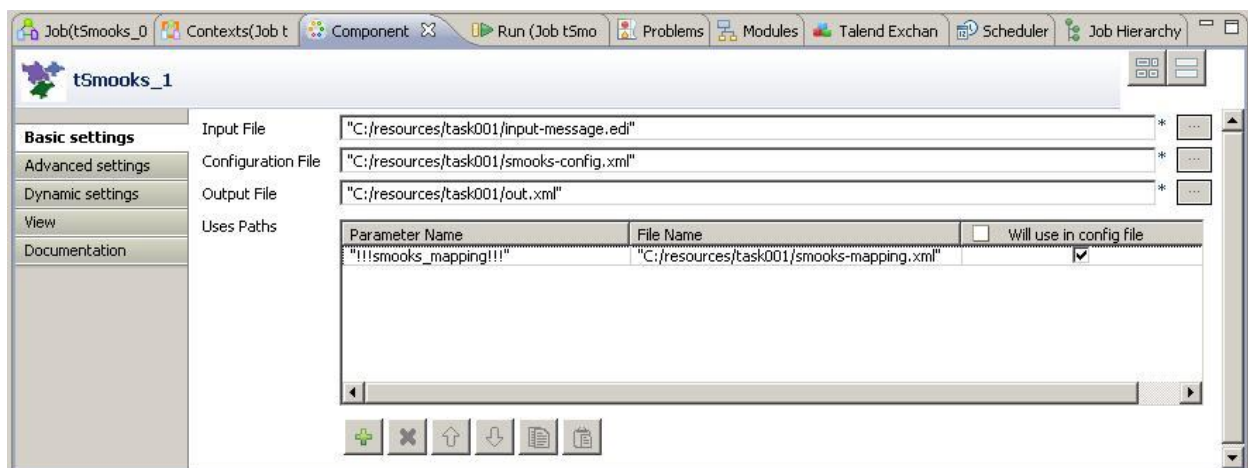
In this example, Smooks applies a piece of Groovy script to a message "date" fragment. This script parses the date, re-adding it to the message as a set of nodes containing the date's constituent parts. The main transformation is then carried out on the whole document ("#document") using XSLT. What this illustrates is how easy it is to combine the powers of different technologies under a single Smooks transform. We use Groovy to pre-process a message fragment that is very difficult to process using XSLT (i.e. a date string). We then use XSLT to do what it's good at i.e. templating.

- The result of data transformation we see in the file, which is defined earlier in the **Output File** field value:
- `<Order date="11-15-2006" netAmount="59.97"
 orderId="1" statusCode="0" tax="4.95" totalAmount="64.92">
<Customer firstName="Harry" lastName="Fletcher"
 state="South Dakota" userName="user1" />
- <OrderLines>
 <order-item price="29.98" product-id="364" quantity="1">
 The 40-Year-Old Virgin</order-item>
 <order-item price="29.99" product-id="299" quantity="1">Pulp Fiction</order-item>
</OrderLines>
</Order>`

Scenario: EDI-to-XML transformation

The following scenario creates a Java job which transform EDI-message to XML file and read a multi schema XML file using the standard Talend components.

- Click and drop a **tSmooks**, **tFileInputMSXML** and two **tLogRow** components from the Palette onto the design workspace.
- Now define the Basic settings of all components.
- Double click **tSmooks** to open the component Basic settings view.



- Browse to the files you want to process. Specify the **Input File**, **Configuration File** and **Output File** field's values.

- Specify the Smooks mapping file as parameter named **!!!smooks_mapping!!!**

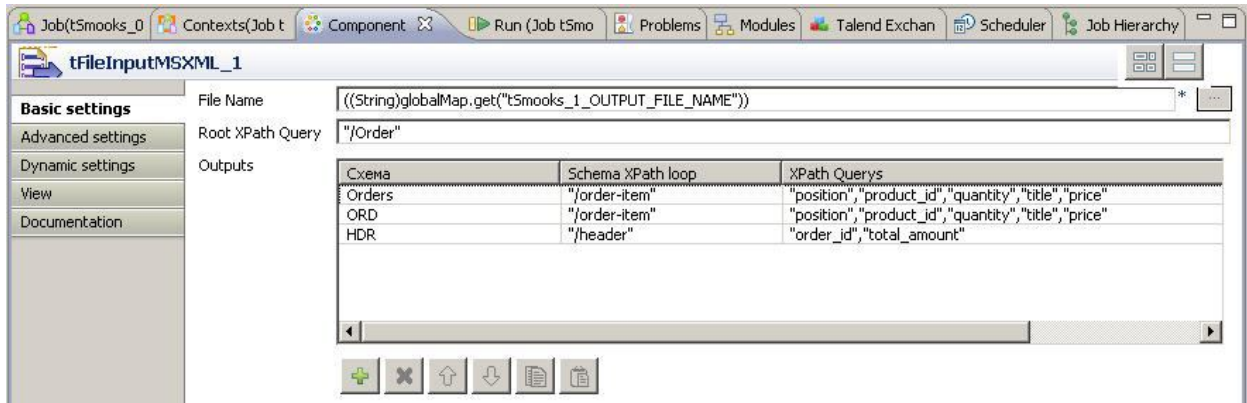
The input EDI-message file and Smooks mapping file are the same as in the tSmooksInput scenario (*Reading a multi structure EDI-message file*).

We used the name of the parameter in a Smooks configuration file instead of specifying the name of Smooks mapping file (“!!!smooks_mapping!!!”).

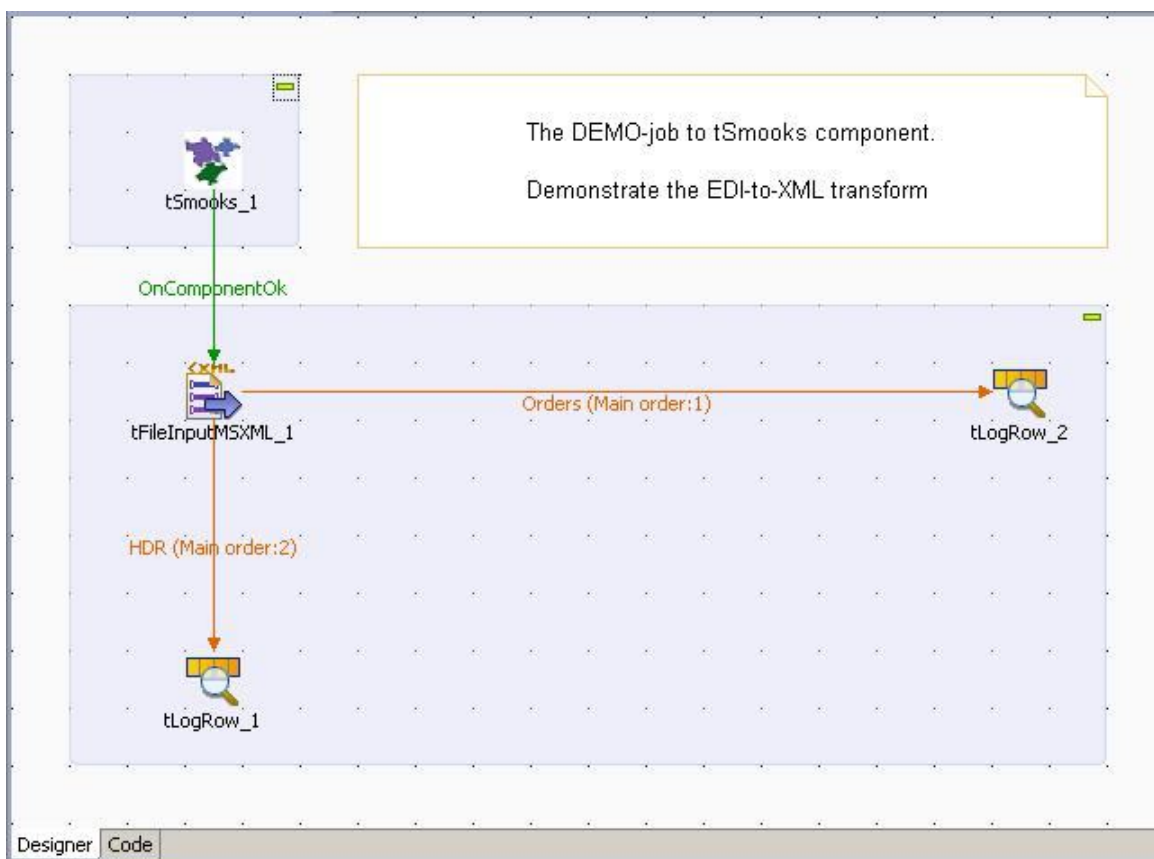
Thus, we have the following Smooks configuration file:

```
<?xml version="1.0" ?>  
- <smooks-resource-list xmlns="http://www.milyn.org/xsd/smooks-1.1.xsd"  
    xmlns:edi="http://www.milyn.org/xsd/smooks/edi-1.1.xsd">  
<!--  
    Configure the EDI Reader to process the message stream into a stream of SAX  
    events.  
-->  
    <edi:reader mappingModel="!!!smooks_mapping!!!" />  
</smooks-resource-list>
```

- Double click **tFileInputMSXML** to open the component Basic settings view.



- Enter the **File Name** field using a variable containing the Smooks output file, as you filled in the Basic settings of **tSmooks**. Press **Ctrl+Space** bar to access the autocomplete list of variables.
- Specify the output schemas in **tFileInputMSXML**. Related topic: tFileInputMSXML in Reference Guide of Talend Open Studio Components.
- Select the last components, **tLogRow** and fill in the separator to be used to distinguish field content displayed on the Log. Related topic: tLogRow in Reference Guide of Talend Open Studio Components.
- Right-click on the **tSmooks** component, and pull an **OnComponentOk** connection to the **tFileInputMSXML** component. Then pull a **HDR** and **Orders** rows from the **tFileInputMSXML** to the **tLogRows** components.



- Save your job and press **F6** execute it.. The EDI-message is transformed into an XML file that is read into the Talend flow using standard Talend components:

```
Starting job tSmooks_001 at 17:13 30/10/2009.
30.10.2009 17:13:20 org.milyn.smooks.edi.EDIReader getMappingModel
INFO: Parsed, validated and cached EDI mapping model [DVD Order, Version 1.0]. Target Profile(s)
[org.milyn.profile.profile#default_profile].
End in process!
```

```

+-----+-----+-----+-----+-----+
|                                     |tLogRow_2|                                     |
+-----+-----+-----+-----+-----+
|position|product_id|quantity|title|price|
+-----+-----+-----+-----+-----+
|1|364|1|The 40-Year-Old Virgin|29.98|
|2|299|1|Pulp Fiction|29.99|
|3|111|20|Robinson Crusoe|480.0|
|4|364|10|From Russia with Love|252.5|
|5|299|25|You Only Live Twice|631.25|
|6|115|1|The Man with the Golden Gun|25.25|
|7|116|7|A View to a Kill|176.75|
+-----+-----+-----+-----+-----+

+-----+-----+
|tLogRow_1|
+-----+-----+
|order_id|total_amount|
+-----+-----+
|1|1625.72|
+-----+-----+

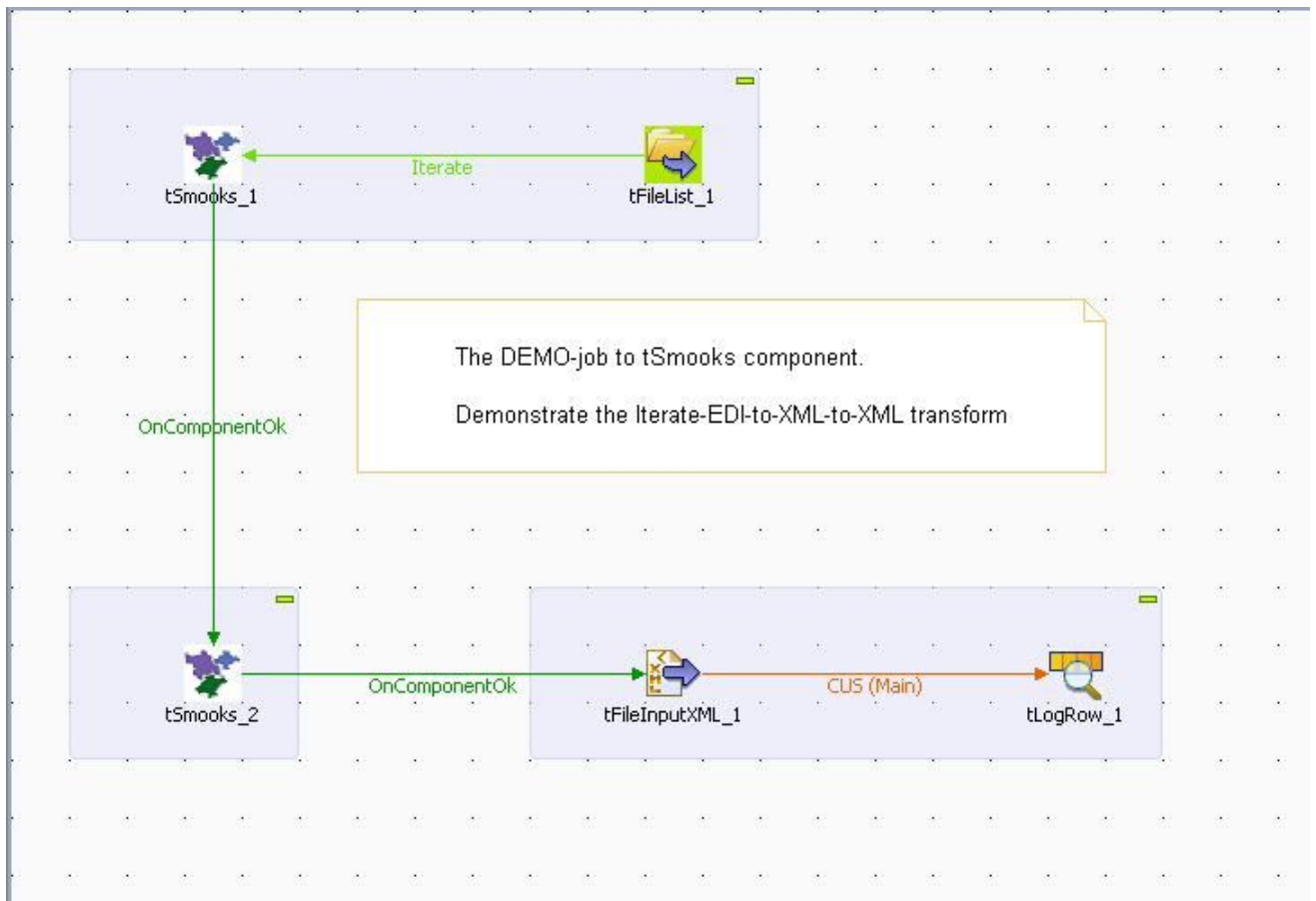
```

Job tSmooks_001 ended at 17:13 30/10/2009. [exit code=0]

Screenshots: Other scenarios

The **tSmooks** component has a broad scope for the application. Below are screenshots of various scenarios of implementation in Talend Jobs.

- Iterate-EDI-to-XML-to-XML data transformation



- CSV-to-XML data transformation (XML-to-XML data transformation)

