JOB AID

Working with files

Overview

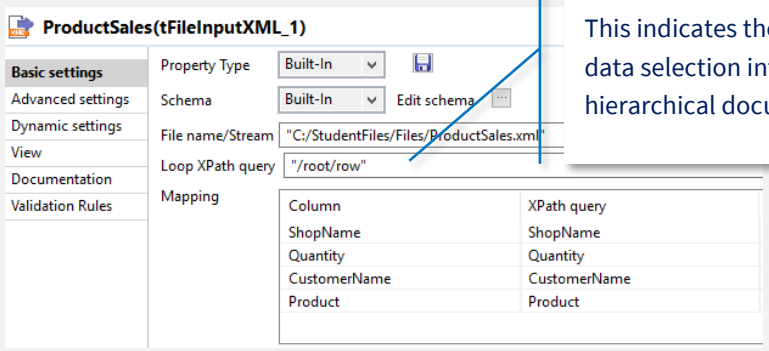
In this module, you learned how to handle different types of structured files. It focused on using Talend Studio components dedicated to reading data from structured files (tFileInputDelimited) and hierarchical files (tFileInputJSON and tFileInputXML).

Key steps

Regardless of the file type, data is interpreted and processed in the same way.

1 Identify file properties so data can be correctly extracted from the file and separated into fields.

- File path
- For delimited files:
  - Row and field separators
  - Column headers
- For hierarchical files:
  - **Loop XPath query** (XML files)
  - **Loop Json query** (JSON files)
  - Column mapping



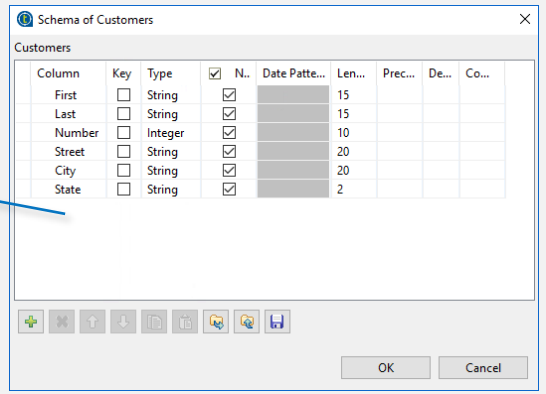
Loop queries

This indicates the path for data selection into hierarchical documents.

2 Define the output schema and map columns.

- The schema is used to define data properties: names of data columns, data type, length.
- Each data input field is then mapped to a column in the schema.
- Schemas are propagated to the next component in the data flow.
  You can propagate a schema again, for instance, after modifying the input schema, by clicking the **Sync columns** button.

You can manually configure the output schema or import it from a file or the repository.

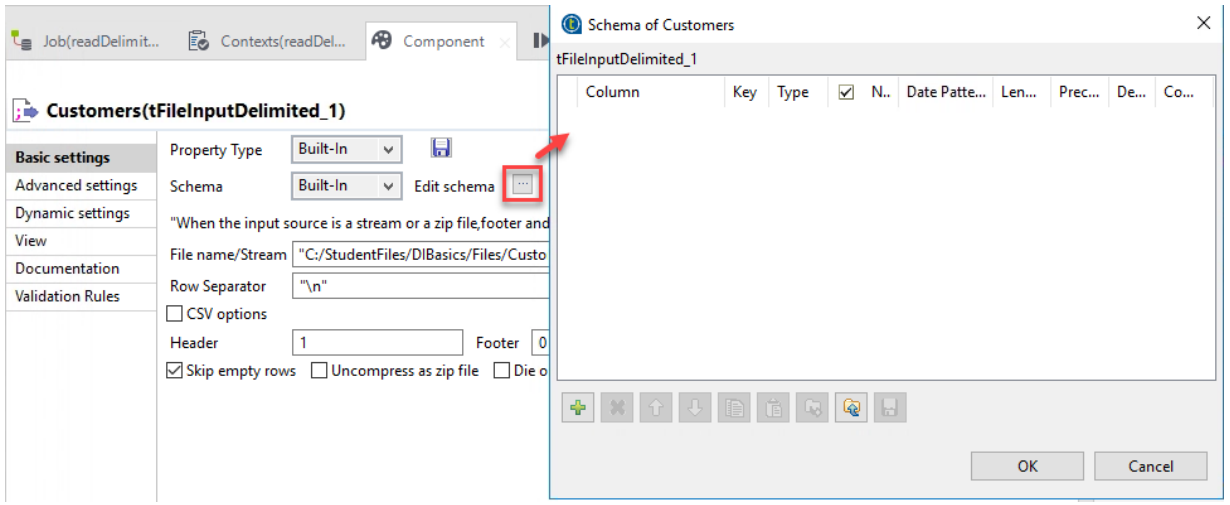


JOB AID

Tips

Settings for working with input components

Schema definition

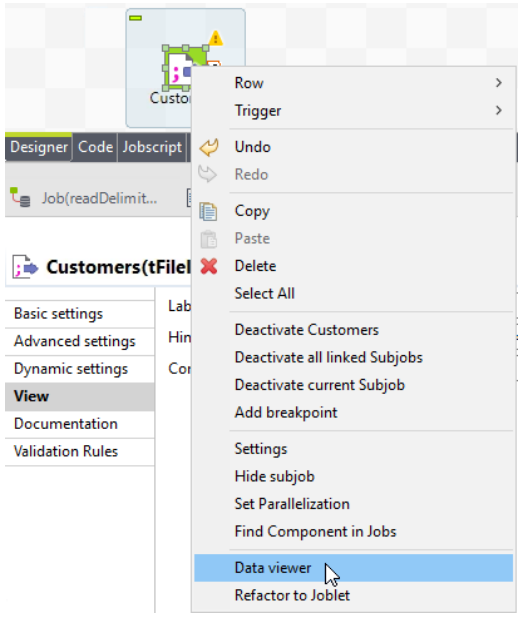


To set up a schema, in the **Basic settings** section on the **Component** tab, click the **Edit schema** button.

Enter column names manually, import them from an XML file, or copy and paste them among components.

You can also create schemas in advance and store them as metadata in the repository. Using this method, you can automatically import column names from header rows of input files and reuse schemas in several components.

Data viewer

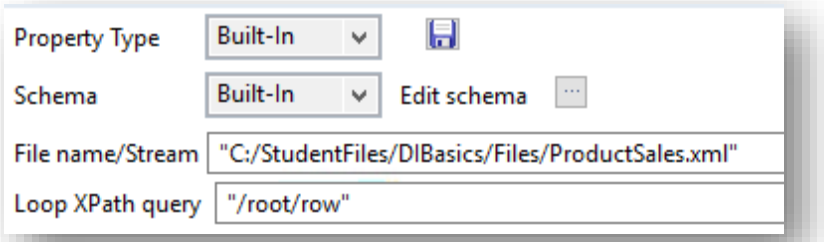


Use **Data viewer** to preview data extracted from a file. This is a good way to confirm that the file reader configuration is set up correctly.

JOB AID

Loop XPath query

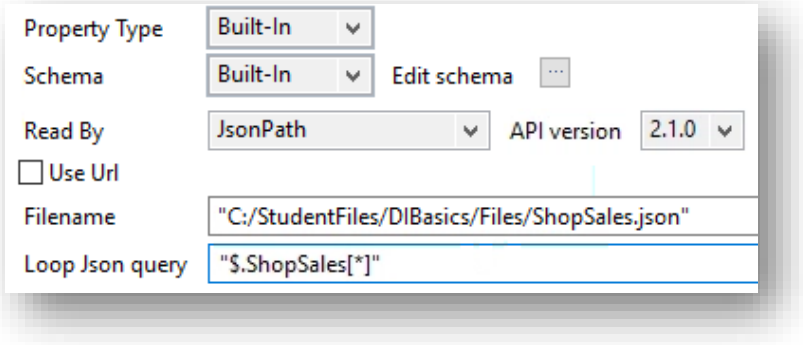
```
<root>
    <row>
        <ShopName>Shop1</ShopName>
        <Quantity>790</Quantity>
        <CustomerName>Acme Chemical</CustomerName>
        <Product>Product 34</Product>
    </row>
    <row>
        <ShopName>Shop1</ShopName>
        <Quantity>886</Quantity>
        <CustomerName>Truck-X Repair</CustomerName>
        <Product>Product 36</Product>
    </row>
    <row>
        <ShopName>Shop1</ShopName>
        <Quantity>972</Quantity>
        <CustomerName>Lee Stairworks</CustomerName>
        <Product>Product 47</Product>
    </row>
</root>
```



Example of a Loop XPath query for an XML file

Loop Json query

```
{
    "ShopSales":
    [
        {
            "ShopName": "Shop1",
            "Quantity": 415,
            "Product": "Product 29",
            "CustomerName": "Johnson, Erico & Co CPA's"
        },
        {
            "ShopName": "Shop1",
            "Quantity": 231,
            "Product": "Product 47",
            "CustomerName": "Gordon & Son Brockerage Ltd."
        },
        {
            "ShopName": "Shop1",
            "Quantity": 790,
            "Product": "Product 34",
            "CustomerName": "Acme Chemical Supply"
        }
    ]
}
```



Example of a Loop Json query for a JSON file