



Talend User Components tElasticSearch*

Purpose

These components are dedicated to work with ElasticSearch in the Data Integration suite of Talend.
These components are working with the http API from ElasticSearch. This is the preferred way to write data into an index according to the recommendation of ElasticSearch.

Component	Purpose
tElasticSearchIndexOutput	Write JSON documents into an ElasticSearch index.
tElasticSearchIndexErrors	Provides the errors from the last write request performed by tElasticSearchIndexOutput
tElasticSearchRequest	Send http requests e.g. read or search documents in ElasticSearch

Talend-Integration

You find this component in the studio palette under: ElasticSearch

Component tElasticSearchIndexOutput

This component writes JSON documents with batch post requests into an ElasticSearch index.

Basic settings

Property	Content	
Schema	This component uses a fixed schema:	
	Column	Purpose
	key	This value is the actual document id. With this id the document can be addressed within the index.
	json	The json document to be indexed. This can be a String or a JsonNode (Jackson API)
	delete	If this boolean value is true, the document with the given key will be deleted from the index. The column json can be null in this case.
Server Nodes	ElasticSearch allows client-side load balancing. The client can determine which server should be used. To allow this setup here a list of ElasticSearch hosts comma separated. The hosts are setup with hostname:port	
Use encrypted conection	The connection will be established encrypted.	
Use Authentication	Activate the authentication for the connection	
User / Password	The user credentials if authentication is required	
Index	The index the component has to write in	
Object Type	The type of objects written into the index	

The JSON documents can easily be created with the tJSONDoc* components.

Advanced settings

Property	Content
Batch Size	The number of documents send as batch to the index

Return values

Return value	Content
ERROR_MESSAGE	Error messages (without details about the records failed inserting)
NB_LINE	The amount of inserted, updated or deleted documents.
NB_LINE_INSERTED	The amount documents inserted
NB_LINE_DELETED	The amount documents deleted

Component tElasticSearchIndexErrors

This returns the errors occurred while indexing documents with tElasticSearchIndexOutput.

Basic settings

Property	Content								
ElasticSearch component	Choose the tElasticSearchIndexOutput component which errors should be returned here								
Schema	<div>This component uses a fixed schema:<table><tr><th>Column</th><th>Purpose</th></tr><tr><td>key</td><td>The key of the affected document.</td></tr><tr><td>operation</td><td>Which operation was performed: index or delete</td></tr><tr><td>failure_message</td><td>The actual error</td></tr></table></div>	Column	Purpose	key	The key of the affected document.	operation	Which operation was performed: index or delete	failure_message	The actual error
Column	Purpose								
key	The key of the affected document.								
operation	Which operation was performed: index or delete								
failure_message	The actual error								

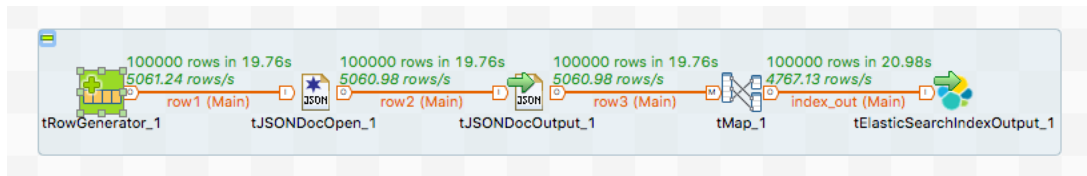
Return values

Return value	Content
ERROR_MESSAGE	Error messages of the operation of this component (not the index errors)
NB_LINE	Number index errors returned
COUNT_ERRORS	The number detected errors

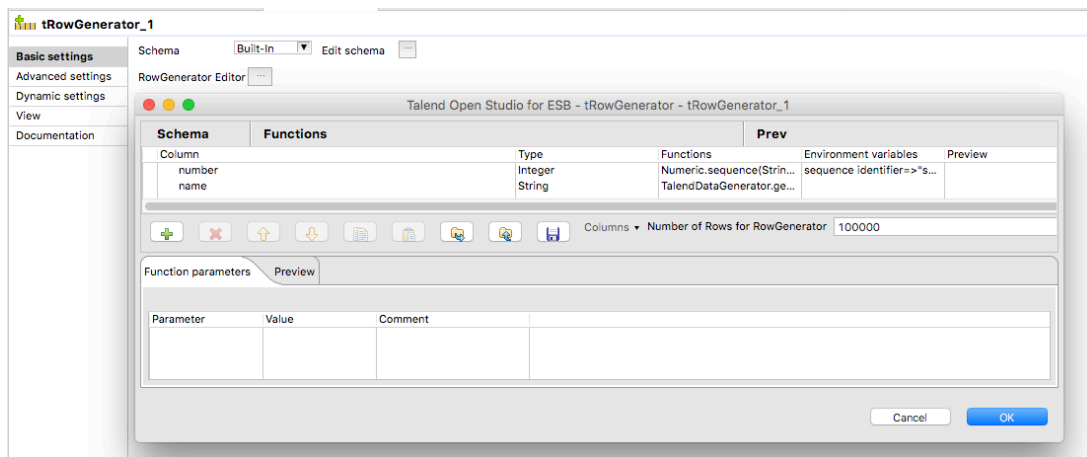
Example Use Cases

This is a load testing job. The source of the data is the row generator from Talend. Its values will be converted with the tJSONDocOpen + tJSONDocOutput component into a json document and after that send to the ElasticSearch index.

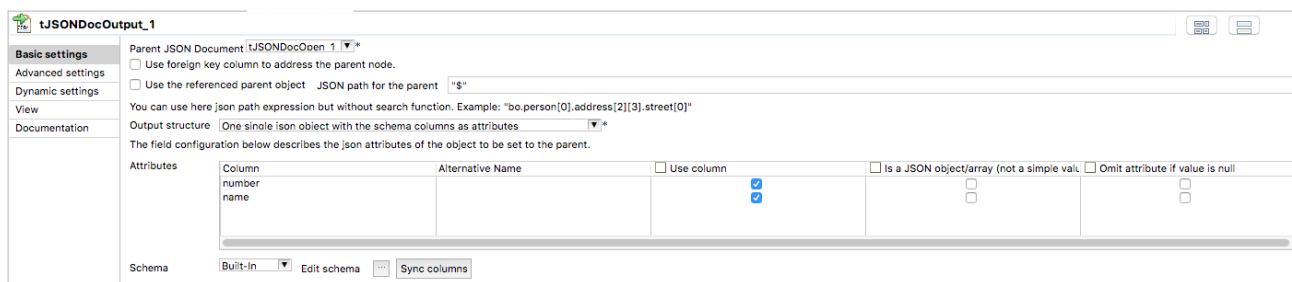
This is the example job design



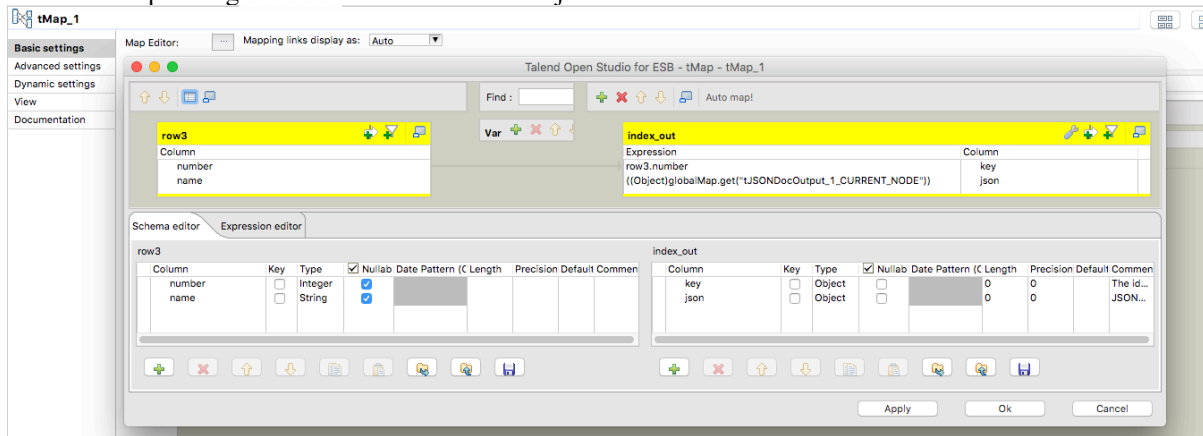
Here the settings of the row generator



The json component settings. The tJSONDocOpen component simply creates the json document and does not do anything to the flow running through it.



Here the tMap settings. You see here how the final json document will be inserted into the flow.



And finally, the setting of the tElasticSearchIndexOutput component:

tElasticSearchIndexOutput_2

Schema: Built-In Edit schema Sync columns

Client Setup

Server Nodes: "searchdev01.gvl.local:9200"

☒ Use encrypted connection

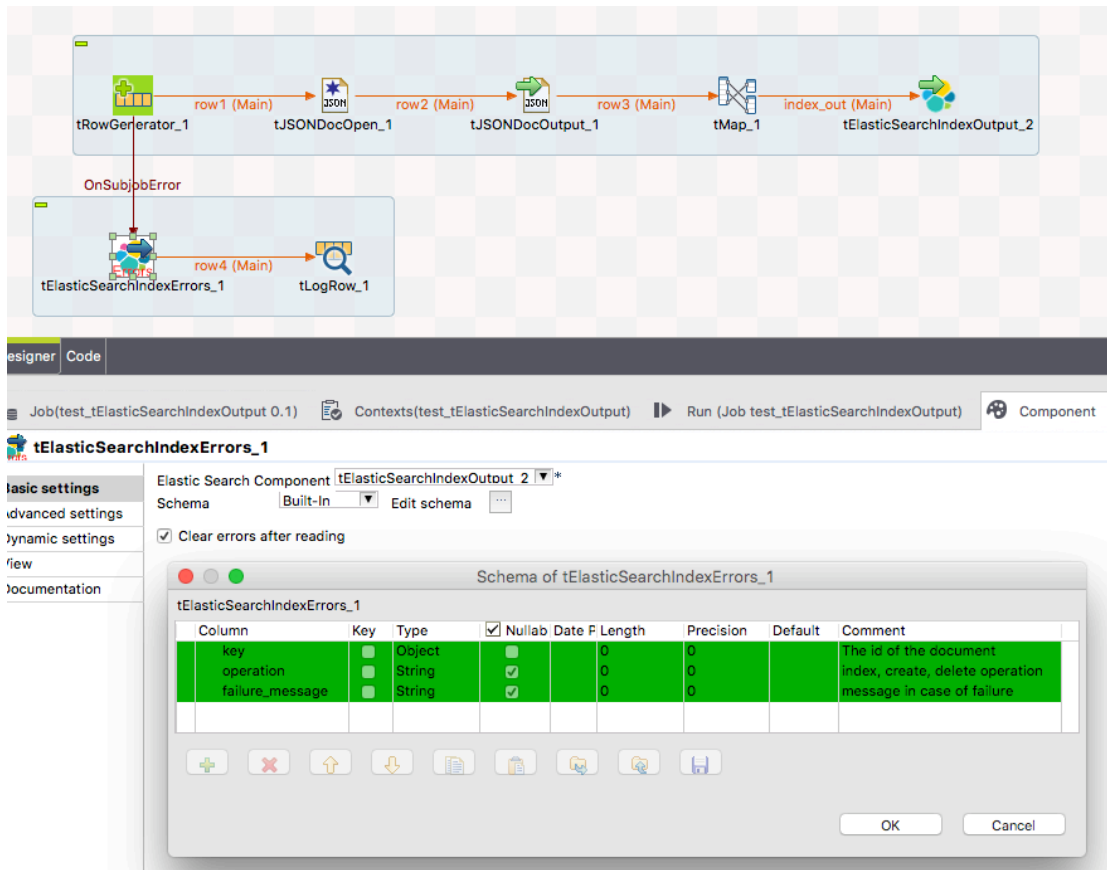
☒ Use authentication

User: "user" Password: *****

Index: "products"

Object Type: "Product"

Here another job design with the output of the errors occurred while indexing:



Component tElasticSearchRequest

This component can send arbitrary requests to ElasticSearch.

Basic settings

Property	Content						
Server Nodes	ElasticSearch allows client-side load balancing. The client can determine which server should be used. To allow this setup here a list of ElasticSearch hosts comma separated. The hosts are setup with hostname:port						
Use encrypted conection	The connection will be established encrypted.						
Use Authentication	Activate the authentication for the connection						
User / Password	The user credentials if authentication is required						
Http Method	GET, POST, PUT, DELETE						
Endpoint path	The path to the desired service endpoint						
Query parameters	A list of query parameters and their values						
Setup request payload from	Read from input field as Java code: Write your payload for the request (e.g. a search request) as json as Java String (with e.g. code snippets like String operations etc) Read from input flow column: Get the payload from an input flow column. In this mode the component needs an incoming flow.						
Request Payload	The payload for the request in case of the first setup option above.						
Request Payload input column	The payload will be read from an input column. This option appears for the second setup option above. The input flow will be directed through the component without changes. Choose here the column containing the payload for the input flow.						
Schema	<div>This component uses a fixed schema:<table><tr><th>Column</th><th>Purpose</th></tr><tr><td>statusCode</td><td>The response http status code</td></tr><tr><td>body</td><td>The response payload</td></tr></table></div>	Column	Purpose	statusCode	The response http status code	body	The response payload
Column	Purpose						
statusCode	The response http status code						
body	The response payload						

Example job with 2 different kind of preparing the payload for the request

Get the request payload from an input field.

The component will be triggered with iteration or via OnSubjobOk or onComponentOk. The component send the request and the response appears in the output flow (see the schema)

The screenshot shows the configuration of the **tElasticSearchRequest_1** component in a workflow designer. The workflow includes a **tLoop_1** component, an **iterate** connector, the **tElasticSearchRequest_1** component, and a **tLogRow_1** component. The configuration panel for **tElasticSearchRequest_1** is open, showing the following settings:

- Client Setup**
 - Server Nodes: `"dksearch01:9200;dksearch:9200"`
 - ☐ Use encrypted connection
 - ☐ Use authentication
- Http Method**: **POST**
- Endpoint Path**: `"/product_matching/product/_mget"`
- Query parameters**: A table with columns **Name** and **Value**.
- Setup the request payload from**: **Read from input field below as Java Code**
- Request payload**: `"{"ids":["1,2,3"]}"`

A modal window titled **Schema of tElasticSearchRequest_1** is displayed, showing the following schema:

Column	Key	Type	Nullab	Date Pattern	Length	Precision	Default	Comment
statusCode		int	<input checked="" type="checkbox"/>		0	0		HTTP status code
body		String	<input checked="" type="checkbox"/>		0	0		JSON response payload

The modal window also includes a toolbar with icons for adding, deleting, and moving columns, and buttons for **OK** and **Cancel**.

Part of the job for send a request which payload comes from an incoming flow.

The flow goes through the component. Incoming columns will be directed to output columns if they exist in the outgoing flow.

The screenshot displays the configuration for the **tElasticSearchRequest_2** component in a flow designer. The top section shows a flow diagram with components **tFixedFlowInput_1**, **tElasticSearchRequest_2**, and **tLogRow_2**. The middle section shows the configuration for **tElasticSearchRequest_2**, including Client Setup, Server Nodes, and Query parameters. The bottom section shows a schema comparison dialog between the input and output schemas.

Client Setup

Server Nodes: "dksearch01:9200;dksearch:9200"

☐ Use encrypted connection

☐ Use authentication

Http Method: **POST** Endpoint Path: **/product_matching/product/_mget**

Query parameters:

Name	Value
------	-------

Setup the request payload from: **Read from input flow column**

Request payload input column: **payload**

Schema: **Built-in** Edit schema Sync columns

Schema of tElasticSearchRequest_2

Column	Key	Type	Nullab	Date	Patte	Length	Precisi	Defar	Cor
payload	<input type="checkbox"/>	String	<input checked="" type="checkbox"/>						
xxx	<input type="checkbox"/>	String	<input checked="" type="checkbox"/>						
yyy	<input type="checkbox"/>	String	<input checked="" type="checkbox"/>						

tElasticSearchRequest_2 (Output)

Column	Key	Type	Nullab	Date	Patte	Length	Precisi	Defar	Cor
payload	<input type="checkbox"/>	String	<input checked="" type="checkbox"/>						
yyy	<input type="checkbox"/>	String	<input checked="" type="checkbox"/>						
statusCode	<input type="checkbox"/>	int	<input checked="" type="checkbox"/>			0	0		HT
body	<input type="checkbox"/>	String	<input checked="" type="checkbox"/>			0	0		JSC

This way you can use information from the input (request) also for processing the response of the request.