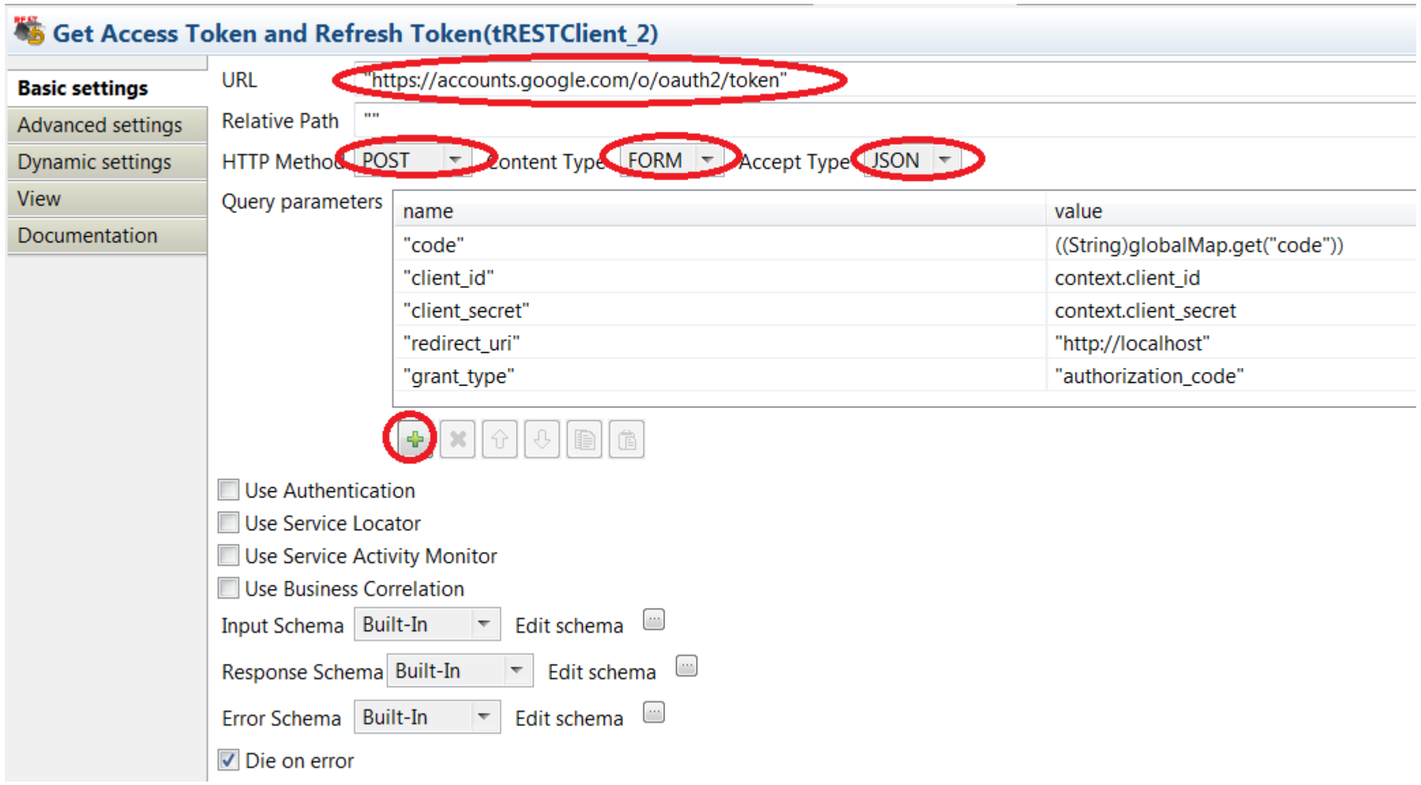
...Continued

15) "Get Access Token and Refresh Token" (tRESTClient)

This component is used to retrieve an access token and refresh token using the authorisation code retrieved from the component before.

To configure this component copy the configuration shown below. Ensure that the sections circled in red are set correctly. To add the "Query parameters" use the green plus symbol circled in red.



The values required can be seen above, but you can find them below so that you can copy and paste them.....

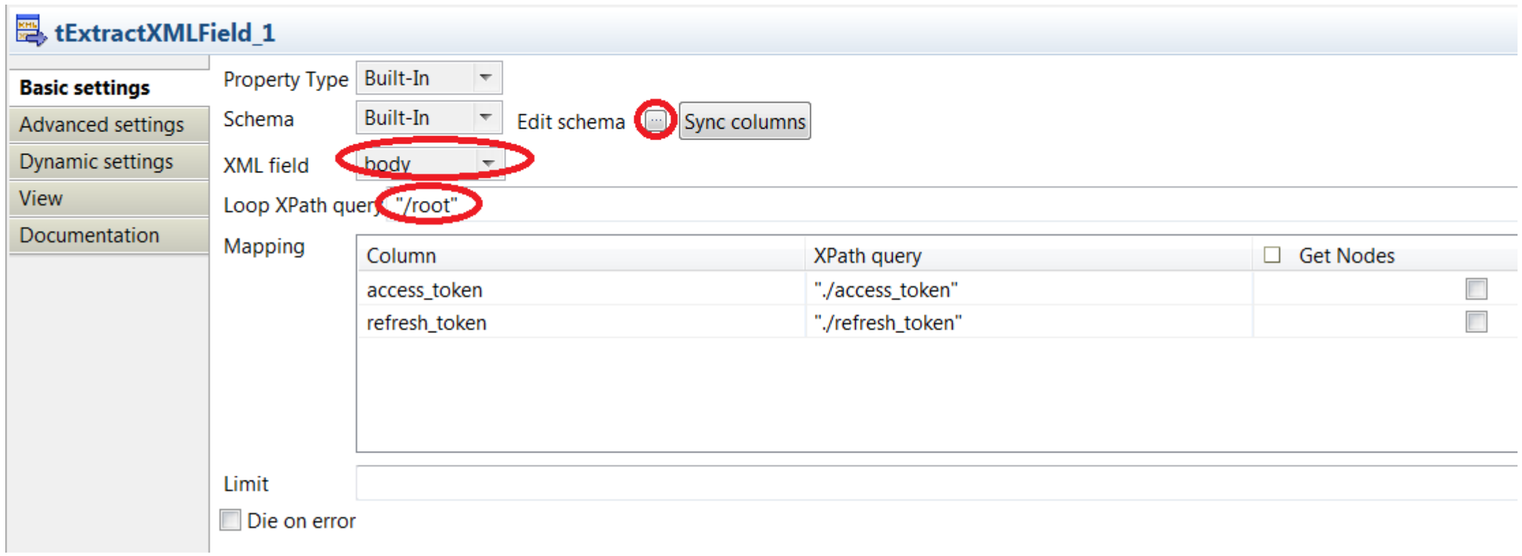
URL: "https://accounts.google.com/o/oauth2/token"

Name	**Value**
"code"	((String)globalMap.get("code"))
"client_id"	context.client_id
"client_secret"	context.client_secret
"redirect_uri"	"http://localhost"
"grant_type"	"authorization_code"

It should be noted that although we are receiving JSON back, this component will automatically convert it to a DOM document with the JSON wrapped with a "ROOT" element by default.

16) "tExtractXMLField_1" (tExtractXMLField)

This component is used to retrieve the access token and refresh token from the returned JSON string which has been converted to an XML document. The configuration of this component can be seen below.....



An output schema is required. To set this up click on the "Edit schema" button circled in red. Two columns called "access_token" and "refresh_token" are required.

Ensure that the areas circled in red are configured as seen above.

The "XPath query" required for the access_token column is "./access_token".

The "XPath query" required for the refresh_token column is "./refresh_token".

17) "tJavaRow_1" (tJavaRow)

This component is used to take the "access_token" and "refresh_token" column values from the previous component and set them as the current values of the Context variables "access_token" and "refresh_token". The code to do this is below....

```
String atoken = input_row.access_token;
String rtoken = input_row.refresh_token;

context.setProperty("access_token", atoken);
if(rtoken!=null){
    context.setProperty("refresh_token", rtoken);
}
```

The "input_row...." bits of code represent the values coming in. The "context.setProperty(..." sections assign the "access_token" and "refresh_token" values.

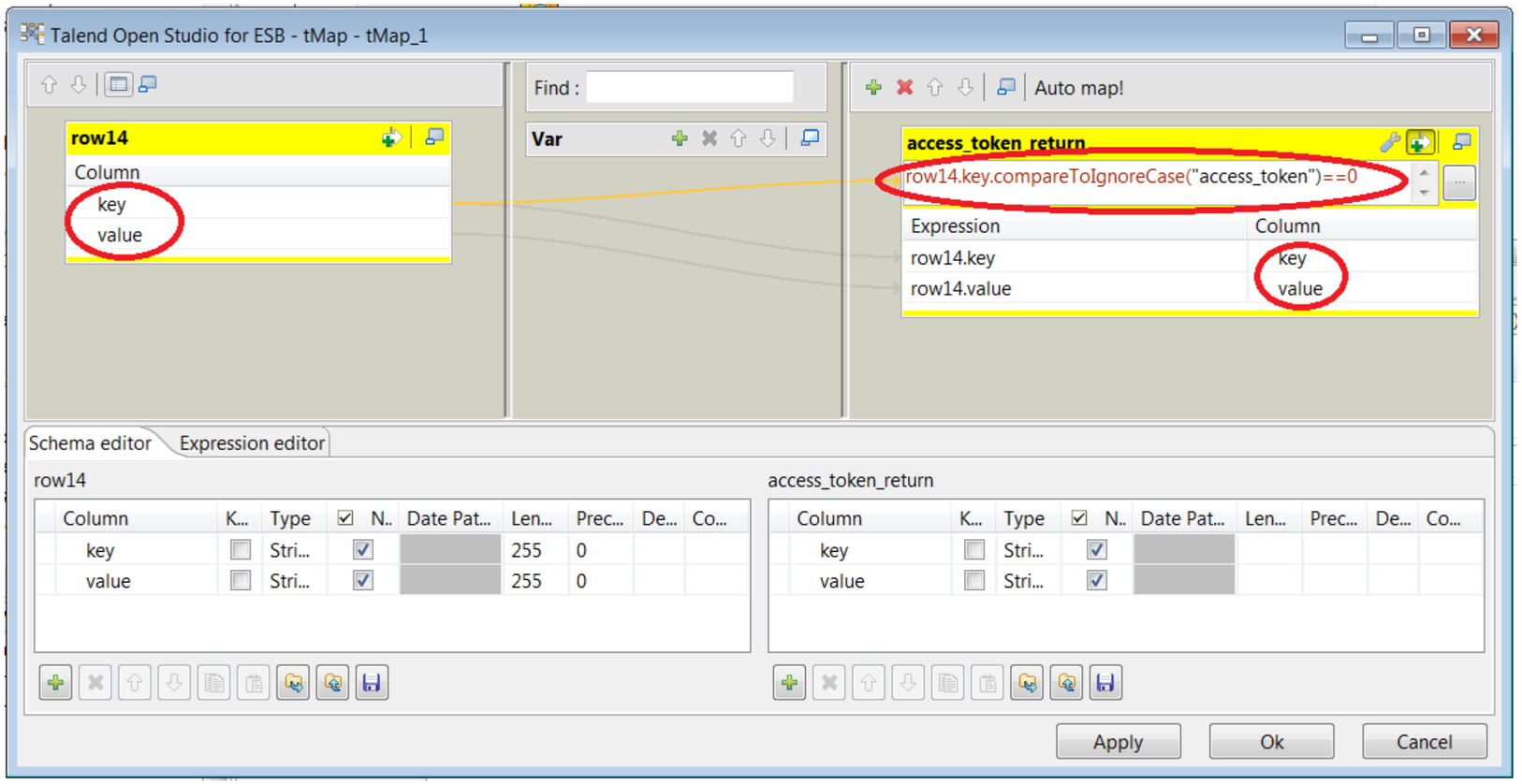
An "IF Condition" is used to cover situations where a "refresh_token" is not received. This should not happen, but this code prevents the Job from falling over if it does.

19) Read the newly set Context variables into the Job and output just the Access Token

This subjob is run at the end of the Job. It will always run, no matter which path the code has taken. It is used to return the access token that has been retrieved/generated. As it has no idea where the access token has come from, it reads the latest value from the Context variable file. As ALL Context variables will be returned from this file, a tMap component is used to filter the return values.

The tFileInputDelimited component points to the Context variable file using the context_file Context variable. It also has the schema that can be seen in the tMap "row14" table. This needs to be configured.

The tMap component can be seen below....



The filter that is used in the "access_token_return" table can be seen below...

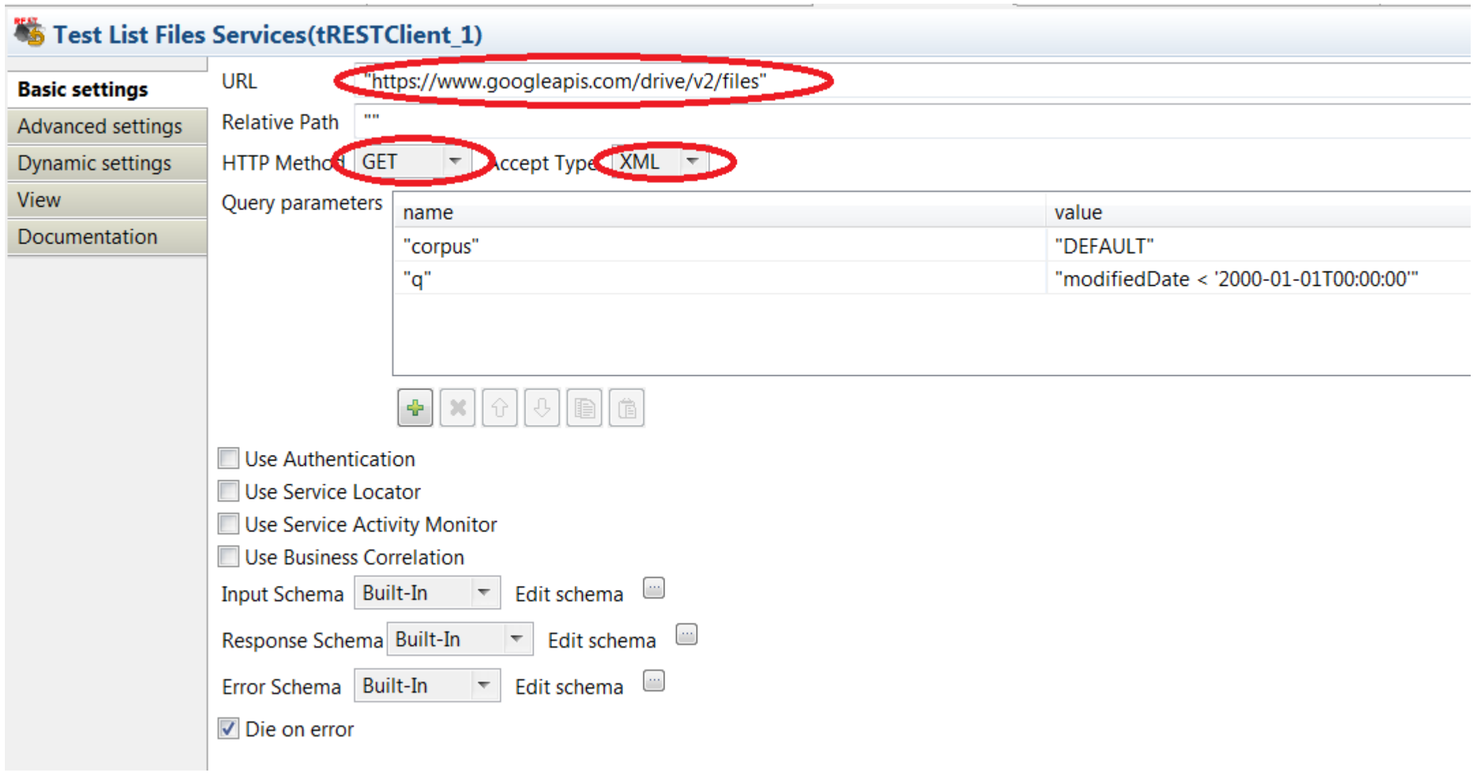
row14.key.compareToIgnoreCase("access_token")==0

Remember that "row14" might be named differently in a version you write. If you have errors here, check the input row name.

20) "Test List Files Services" (tRESTClient)

This component is to simply test the access_token that is said to exist. If it tests successfully, the Job will end. If it fails, the error trigger will be used and the Job will attempt to generate a new one.

To configure this component copy the configuration shown below. Ensure that the sections circled in red are set correctly. To add the "Query parameters" use the green plus symbol circled in red.



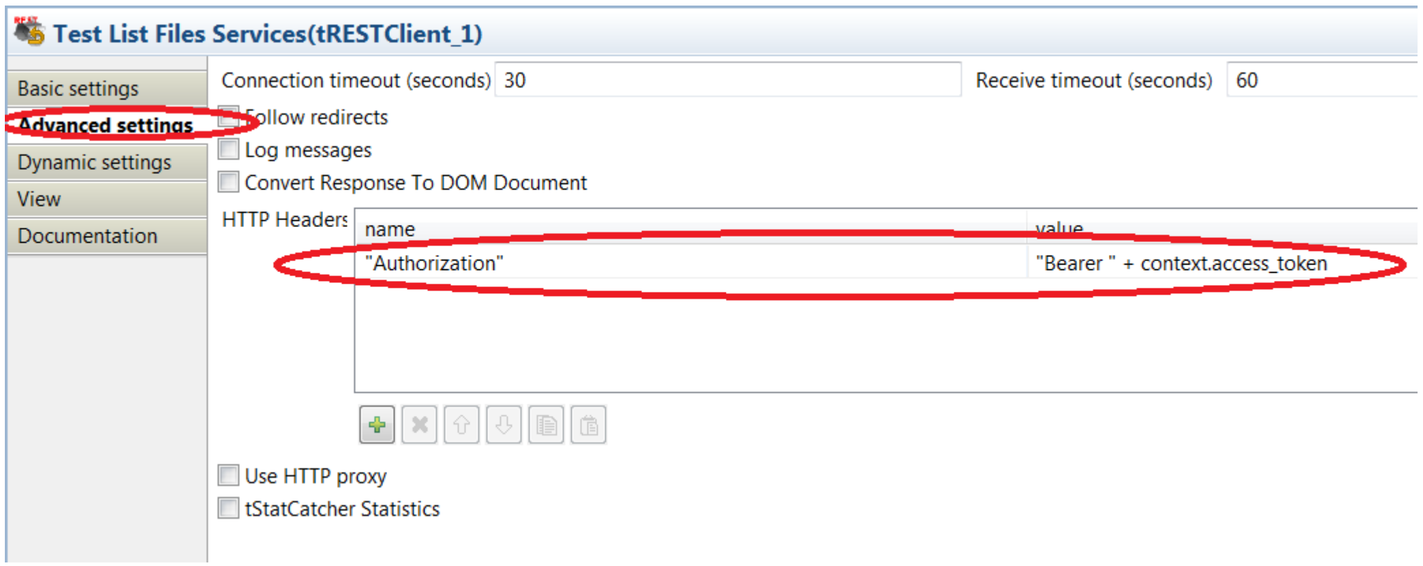
The values required can be seen above, but you can find them below so that you can copy and paste them.....

URL: "https://www.googleapis.com/drive/v2/files"

Name	Value
"corpus"	"DEFAULT"
"q"	"modifiedDate < '2000-01-01T00:00:00'"

This HTTP request is described by Google here. I have used a query to search for files with a modified date less than 2000/01/01. This has been done so that a successful response will return no data.

In order for the HTTP request to work, we need to provide the access token. This is done via the "Advanced Settings" tab as can be seen below....



The access token is provided by the HTTP header "Authorization". Its value must be a combination of the word "Bearer " (with a space) and the access token that has been supplied.

The Context Variable File

Below we can see an example of what the Context variable file will need to look like when it is first run. The variables that are assigned values here must be assigned values in your version. When the Job has been run for the first time, all of the values will be populated.

```
client_secret;YIMgcQ24ghjt65GHy8wTtiSpn8
refresh_token;
redirect_uri;http%3A%2F%2Flocalhost
scope;https://www.googleapis.com/auth/drive
context_file;C:/Talend/OpenSource/5.5.1/Studio/workspace/contextGoogle.csv
client_id;689878354248.apps.googleusercontent.com
access_token;
```

Notice that the "context_file" variable has been set. It MUST point to its own location.

The "redirect_uri" variable is "http://localhost" where the value has been URL encoded. This could be done inside the Job if you prefer to leave this as a natural value.

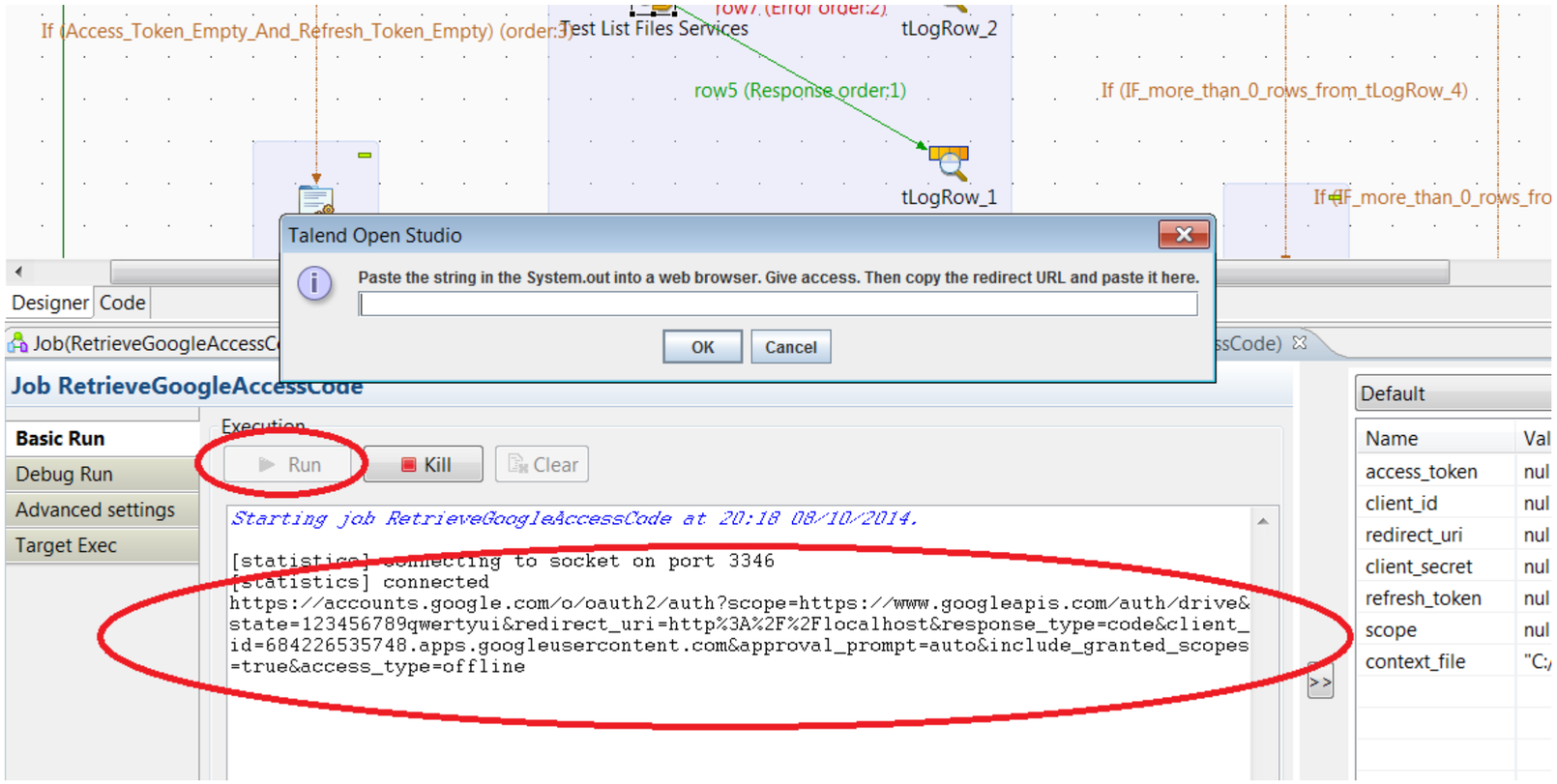
The "scope" variable is described by Google here.

Running the Job for the first time

This Job can be run on its own to demonstrate that it works. It will print the access token to the System.out. It can also be used as a child Job that returns a key/value pair holding the access_token to be used by the parent. This section will demonstrate the Job being run as a standalone Job.

1) Running the Job

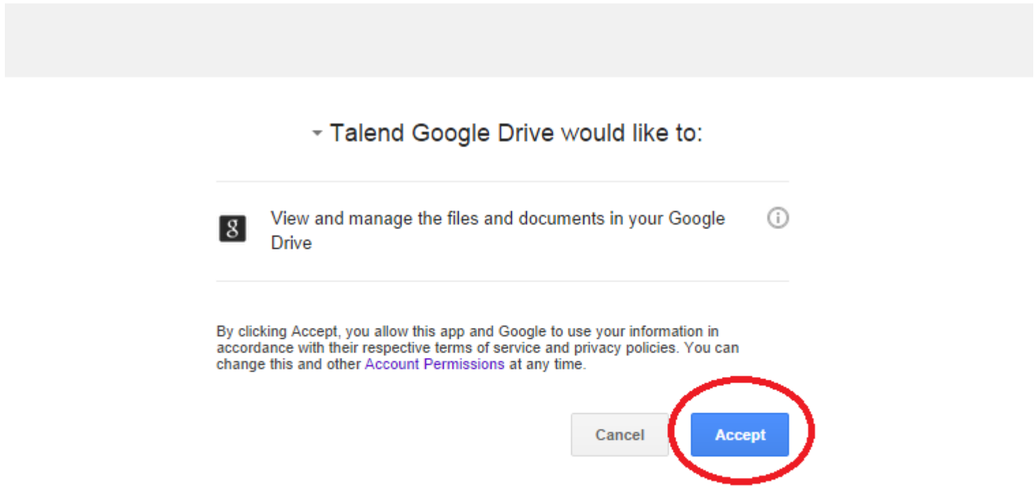
When running for the first time, we need to make sure that the Context variable file is fully configured minus values for the refresh_token and access_token (as seen above). Once that is sorted, load the Job and click on the "Run" button (circled in red).



This will produce a string in the System.out. Copy this string (circled in red) and paste it into a web browser.

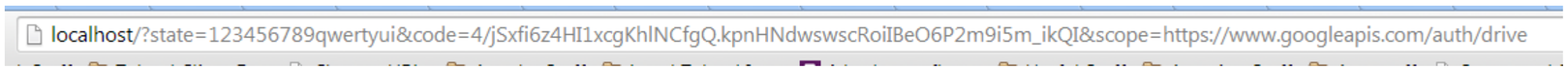
2) Authorise the Talend Job with Google

When the Google authentication page loads, click on the "Accept" button. As below....



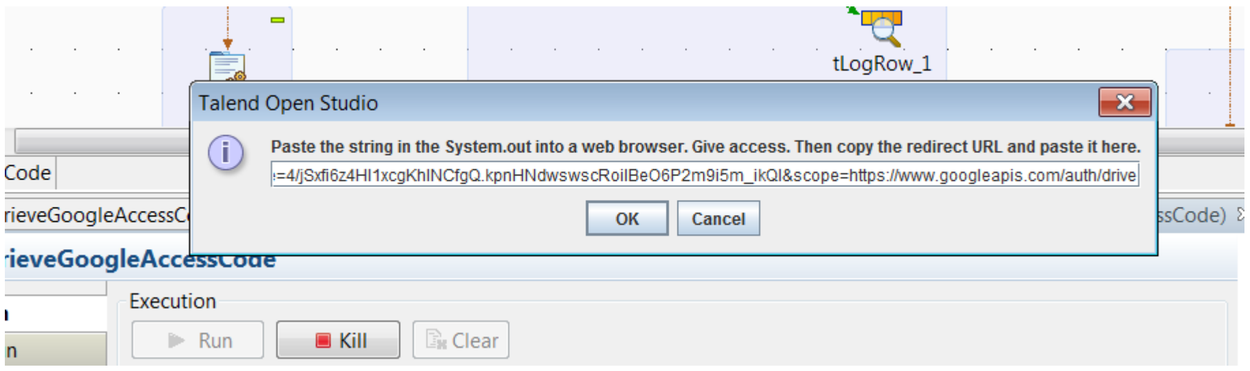
3) Copy the Authorisation Redirect URL

If the authorisation has worked, a redirect URL (like below) will be returned. Copy it.



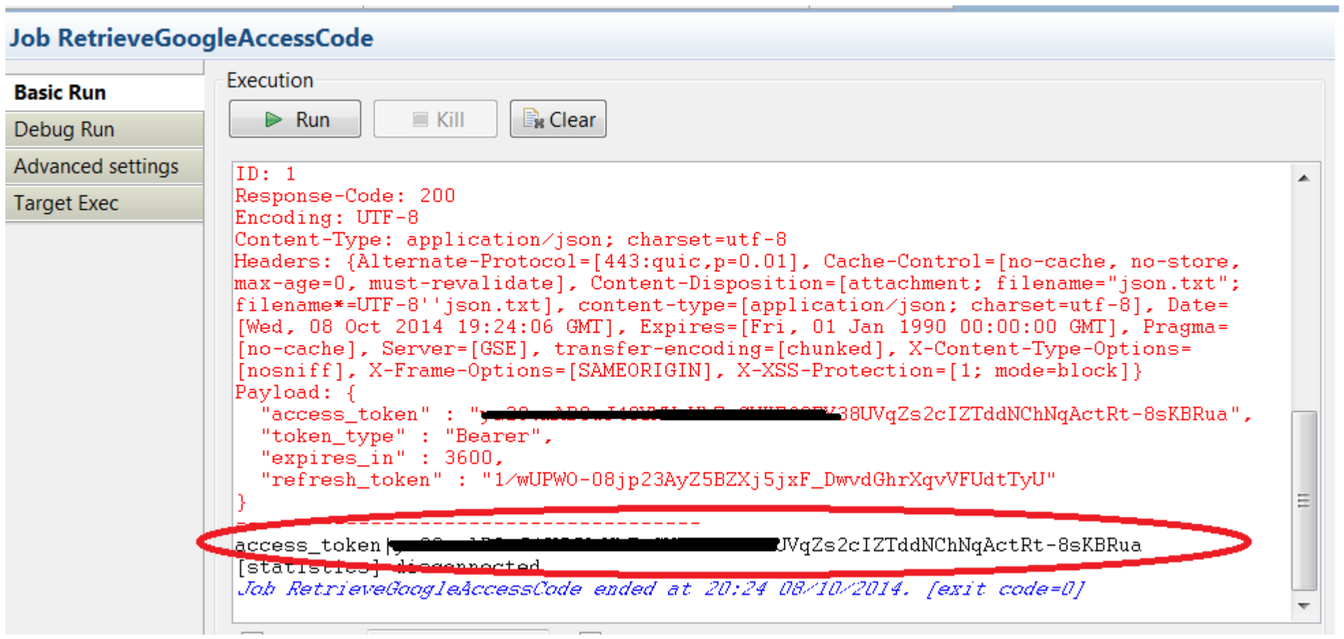
4) Pass the Redirect URL back to the Talend Job

Paste the value copied from the web browser address bar into the message box and click "OK". The Job will then continue.



5) The Access Token is Generated

As can be seen below, the Access Token will be displayed at the botton of the System.out (circled in red). It will also be added to the Context variable file along with the Refresh Token.

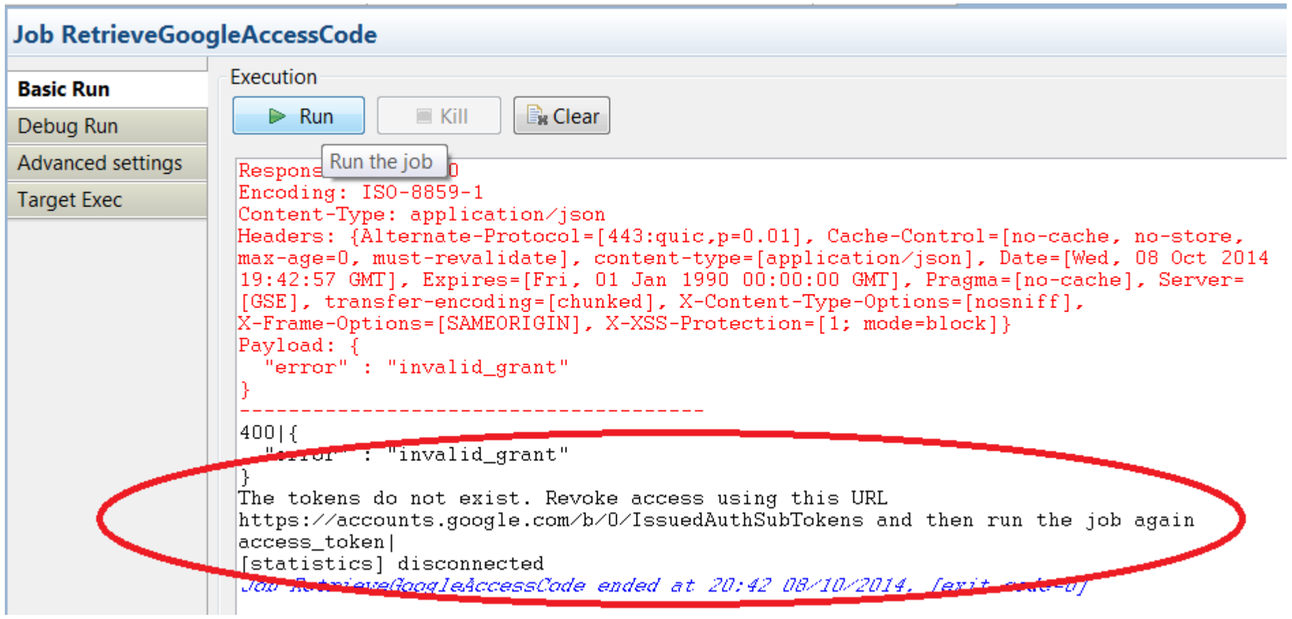


Refreshing the Access Token from the Refresh Token

After the Refresh Token and Access Token have been generated for the first time, there should be no need for future human interaction unless the Refresh Token has been lost. To show this, open the Context variable file and add a few random characters to the Access_Token variable. Then run the Job as above. You will notice that there is no user interaction required and that a new Access_Token is generated.

Resetting the Refresh Token

You may find that for whatever reason the Refresh Token is not working or has been lost. If this is the case then the Talend Job will need it's authentication revoked before the Job can be run again from scratch. This is an unusual situation, but needs to be covered. To emulate this, open the Context variable file and alter some of the characters of the Access_Token and Refresh_Token. Then run the Job. You will see a screen like below informing you to revoke the access and giving you a URL to use....



Open the URL in a web browser and revoke access to your Talend Job (using the name you specified when you created the Google Project). Then start from scratch.

Running the Job as a child Job

This Job can be run as any child Job in Talend. Ensure that you remember to configure a schema for the child Job that returns exactly what is output by the tBufferOutput component.