JSON REST Web Services Connector

- JSON REST Web Services Connector
- JSON REST Web Services Connector - Properties
- How to configure the Office 365 agent?
- How to configure the Jira Atlassian agent?

JSON REST Web Services Connector

Introduction

Description

This connector allows the integration with any Web Service able to consume and generate JSON documents through REST communication.

Managed System

Every commercial product or custom web application allows REST communication with JSON documents.

There are a lot of products that use this standard, for instance:

- JIRA.
- Oracle Field Service Cloud (OFSC).
- Office 365.
- AWS.
- Google Cloud.

> If your system is not in the previous list, it's possible to include it easily!
>
> For more information to check if your system may be synchronized with this connector you do not hesitate to contact us through our Contact form

Prerequisites

It is needed a user with access and permissions to the endpoints and operations required in the scope of the integration.

Also, the documentation, specification, or tutorial of the implementation of the JSON REST Web Service is required to apply the mapping configuration.

Download and Install

This addon is located in the Connectors section and its name is **REST (json) plugin**.

> You can visit the Addons Getting started page for more information about the installation process.

Agent Configuration

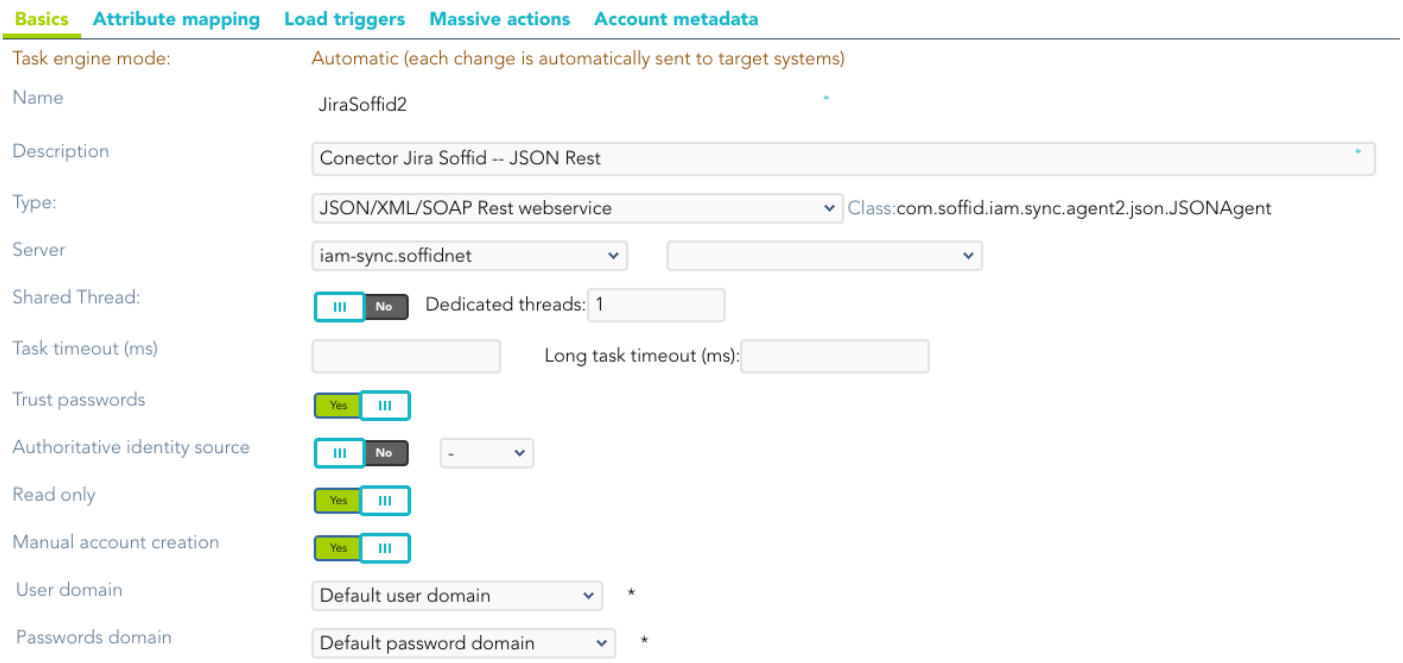
Basic

Generic parameters

After the installation of the addon, you may create and configure agent instances.

To configure this JSON REST Web Service Connector you must select "JSON Rest Webservice" in the attribute "Type" of the generic parameters section in the agents' page configuration.

> For more information about how you may configure the generic parameters of the agent, see the following link: Agents configuration



Custom parameters

Below there are the specific parameters for this agent implementation:

Parameter	Description
Server URL	URL of the REST web service. Base URL for making calls.
Authentication method	Available options: • **None**: no authentication (User and Password are not used). • **Basic**: it uses the User and Password to generate the authentication token. • **Bearer token (Simple token in Soffid 4)**: it is provided by the application to which we are trying to connect. • **Token (Token got from user and password in Soffid 4)**: generate a token from a specific authentication URL. It is no longer used. • **Token oAuth Client Credentials**: authenticates based on a client ID and a client secret. • **Token oAuth Password Grant**: authenticates based on a client ID and a client secret plus a user name and a password. *(*) You can find more information in the Authentication method section.*
Enable debug	Two options: "Yes", "No": it enables or not more log traces in the Synchronization Server log
Proxy host	Only when the proxy is needed.
Proxy port	Only when the proxy is needed.

Parameter	Description
XML Templates	Allows you to add new XML templates with SOAP requests and then configure them at attribute mappings.

Authentication *method*

None: no authentication is needed. There are no parameters to configure.



Basic: the username and password are sent with each request.

- **User Name**: user to authenticate.
- **Password**: the password of the user to authenticate.



Bearer token (Simple token in Soffid $)

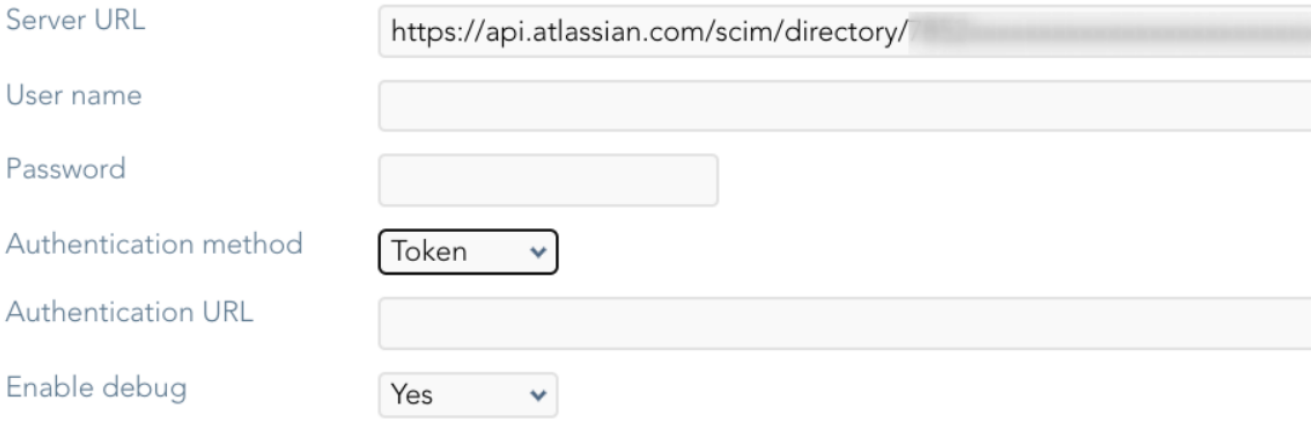
- **Bearer token (Token in Soffid 4)**: this token is provided by the application to which we are going to connect.



Token (Token got from user and password in Soffid 4): calls the authentication URL with the POST method and with the username and password, and the response will be the token. It is no longer used.

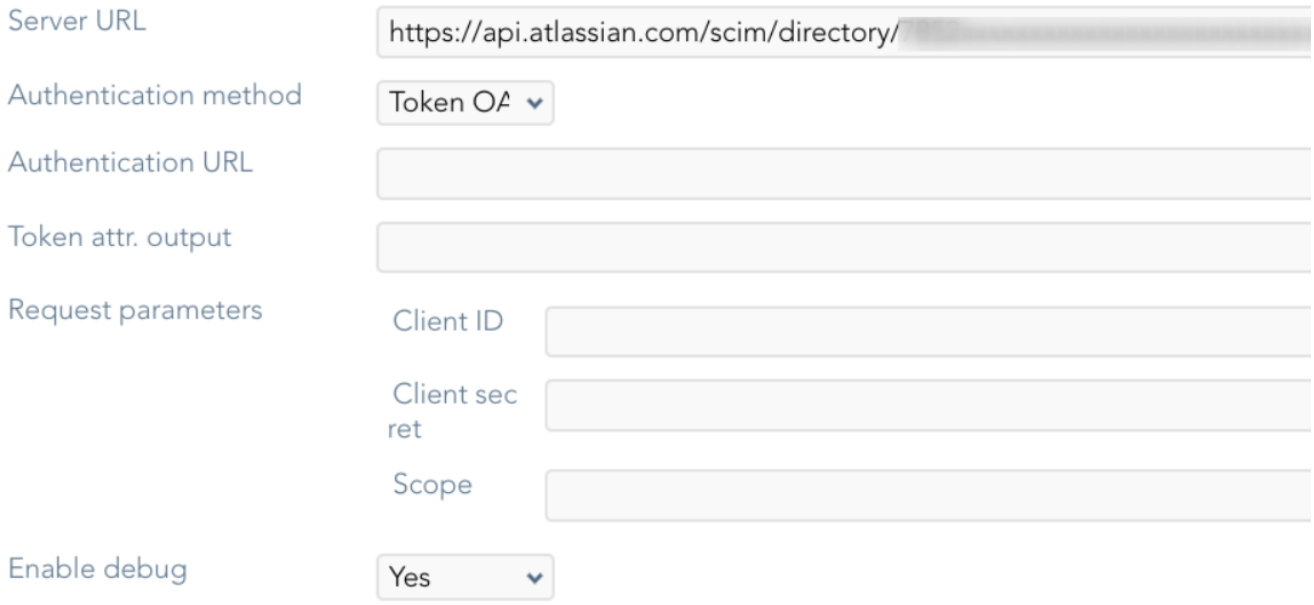
- **User Name**: user to authenticate.

- **Password**: the password of the user to authenticate.
- **Authentication URL**: URL to retrieve the token for the server's authentication (for the "Token" method).



Token oAuth Client Credentials

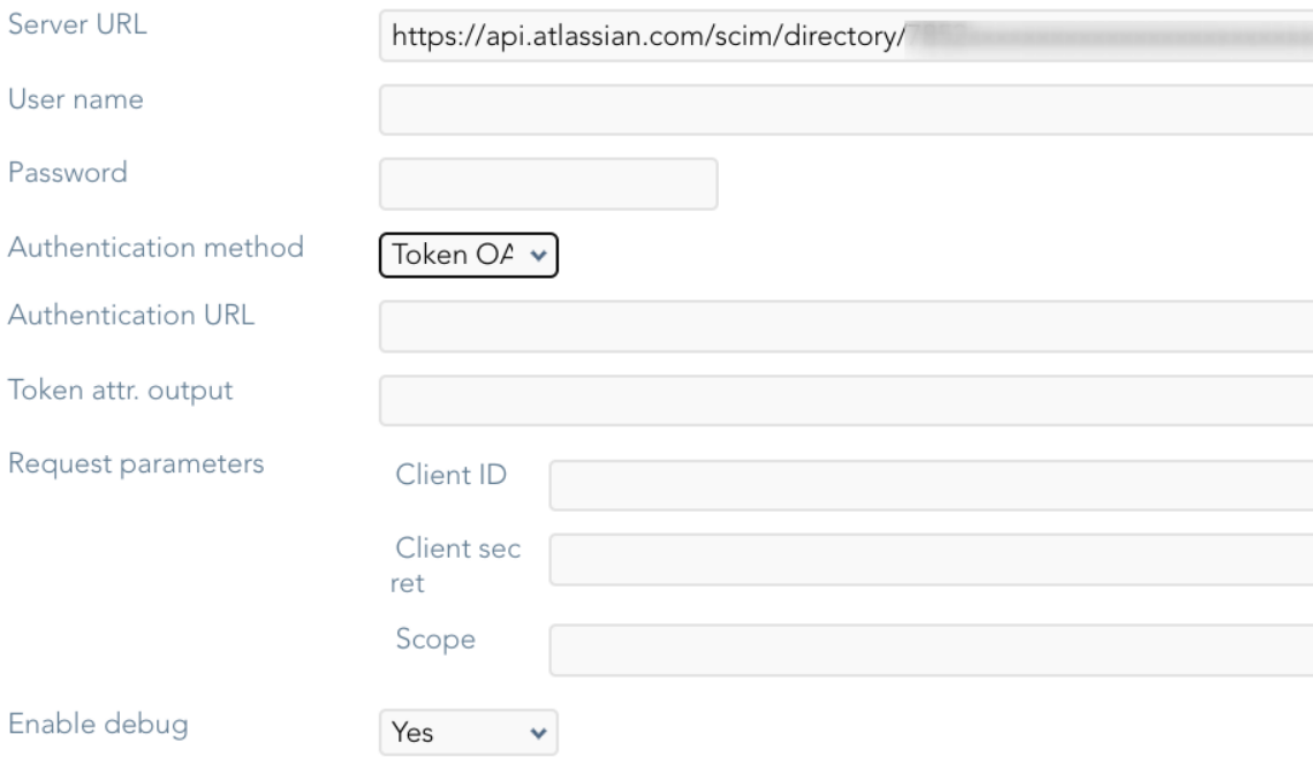
- **Authentication URL**: URL to retrieve the token for the server's authentication (for the "Token" method).
- **Token attr. output**: the value is always *access_token*.
- **Request parameters**:
  - **Client ID**: it is like the user.
  - **Client secret**: it is the password.
  - **Scope**: it is the permissions.



Token oAuth Password Grant

- **User Name**: user to authenticate.
- **Password**: the password of the user to authenticate.
- **Authentication URL**: URL to retrieve the token for the server's authentication (for the "Token" method).

- **Token attr. output**: the value is always *access_token*.
- **Request parameters:**
  - **Client ID**: it is like the user.
  - **Client secret**: it is the password.
  - **Scope**: it is the permissions.



Attribute mapping

This connector can manage users, accounts, roles, groups, and grants.

> Note that any changes made to the methods will affect the properties and vice versa.

Methods

This agent allows you to define methods to be called using the defined properties. There are some default methods, but you can customize your own methods.

Default methods:

- load
- delete
- update
- insert
- select

For each method, the properties to set up are as follows:

Properties	Description
Path	A valid URL to call. This path must be the continuation of the Server URL for making calls.
Method	Available methods to call a Rest API (GET, POST, PUT, DELETE, PATCH)
Encoding	The specific type of encoded data that will be used. There are three supported types: • application/x-www-form-urlencoded • application/json • text/xml
XML Template	Applies only if it is *text/xml*. You need to write the name of the corresponding template defined on the XML Templates.
Parameters	Applies with *application/x-www-form-urlencoded* and *application/json* • You must type which attributes, defined on the System attributes, will be sent. • If none are to be sent, you must write the hyphen character "-". • If nothing is typed, all parameters are sent.

Success HTTP Codes	HTTP codes that should be interpreted as OK. If no code is entered, Soffid will take as valid codes the following: 200, 201, 204 and 404.
	If you type the Success HTTP codes, it will be not necessary to type the Failure HTTP codes.
	`204 201 200`
	`200,212`
	You can use blanks or commas to separate the codes.

Failure HTTP Codes	Soffid will take by default as failure all codes not indicated in Success HTTP Codes.
	If you type the Failure HTTP codes, it will be not necessary to type the Success HTTP codes.
	`400 403`
	`403,405, 400`
	You can use blanks or commas to separate the codes.

Properties	Description
Results	Gets the object or object list from the response received. You need to indicate a JSON attribute name to check and get the data. If this element is not present, or empty, the connector will conclude the user does not exist yet. You can type simple attribute names or even complex scripts.
Pagination URL	Often, the response from the API Rest service does not contain all the data because the data is too large. In these cases, you can use the paging options to request the data in blocks. When the response gives us the URL of the next page to fetch, you must type the tag name of this attribute. `return links{"next"};` You have to choose one of the paging methods, using both is not compatible.

Pagination script	Often, the response from the API Rest service does not contain all the data because the data is too large. In these cases, you can use the paging options to request the data in blocks.
	You can type a complex script to get the next call that has to be done. There are two available objects:
	• response: JSON response as received
	• request: allows you to update the attributes and return true if you want to make a new call or false in another case
	
	`o = response{"paging"};`
	`if (o{"has_next_page"}) {`
	`  nextPage = o{"page_number"} + 1;`
	`  request.put("page", nextPage);`
	`  return true;`
	`} else {`
	`  return false;`
	`}`
	
	You have to choose one of the paging methods, using both is not compatible.

Condition script	Return false if you want to prevent a call.
Optional header	Use this property to send HTTP header(s). More than one header can be sent by adding multiple properties Optional Header1. The value of the header is "HEADER:VALUE", for instance, "Accept:application/json".

Load

load	* Path	/Groups		
	* Method	GET		
	* Encoding			
	XML Template	application/json		
	Parameters			
	Success HTTP Codes			
	Failure HTTP Codes			
	Results	Resources		
	Pagination URL			
	Pagination script			
	Condition script			
	Optional header			
	Optional header			
	Optional header			

Select

select	* Path	/Groups?$filter=displayName+eq+${displayName}		
	* Method	GET		
	* Encoding	application/json		
	XML Template			
	Parameters	-		
	Success HTTP Codes			
	Failure HTTP Codes			
	Results	Resources		
	Pagination URL			
	Pagination script			
	Condition script			
	Optional header			
	Optional header			
	Optional header			

Insert

insert	* Path	/Groups		
	* Method	POST		
	* Encoding	application/json		
	XML Template			
	Parameters			
	Success HTTP Codes			
	Failure HTTP Codes			
	Results			
	Pagination URL			
	Pagination script			
	Condition script			
	Optional header			
	Optional header			
	Optional header			

Update

update	* Path	/Groups/${id}		
	* Method	PUT		
	* Encoding	application/json		
	XML Template			
	Parameters			
	Success HTTP Codes			
	Failure HTTP Codes			
	Results			
	Pagination URL			
	Pagination script			
	Condition script			
	Optional header			
	Optional header			
	Optional header			

Delete

delete	* Path	/Groups/${id}
	* Method	DELETE
	* Encoding	application/json
	XML Template	
	Parameters	
	Success HTTP Codes	
	Failure HTTP Codes	
	Results	
	Pagination URL	
	Pagination script	
	Condition script	
	Optional header	
	Optional header	
	Optional header	

Properties

In this agent, the configuration of the properties attributes is very important due to they define the functionality of the integration:

This agent has five families of properties:

Family	Description
Load	Used to retrieve all the objects in the target system
Select	Used to retrieve an object in the target system
Insert	Used to create an object in the target system
Update	Used to update an object in the target system
Delete	Used to remove an object in the target system

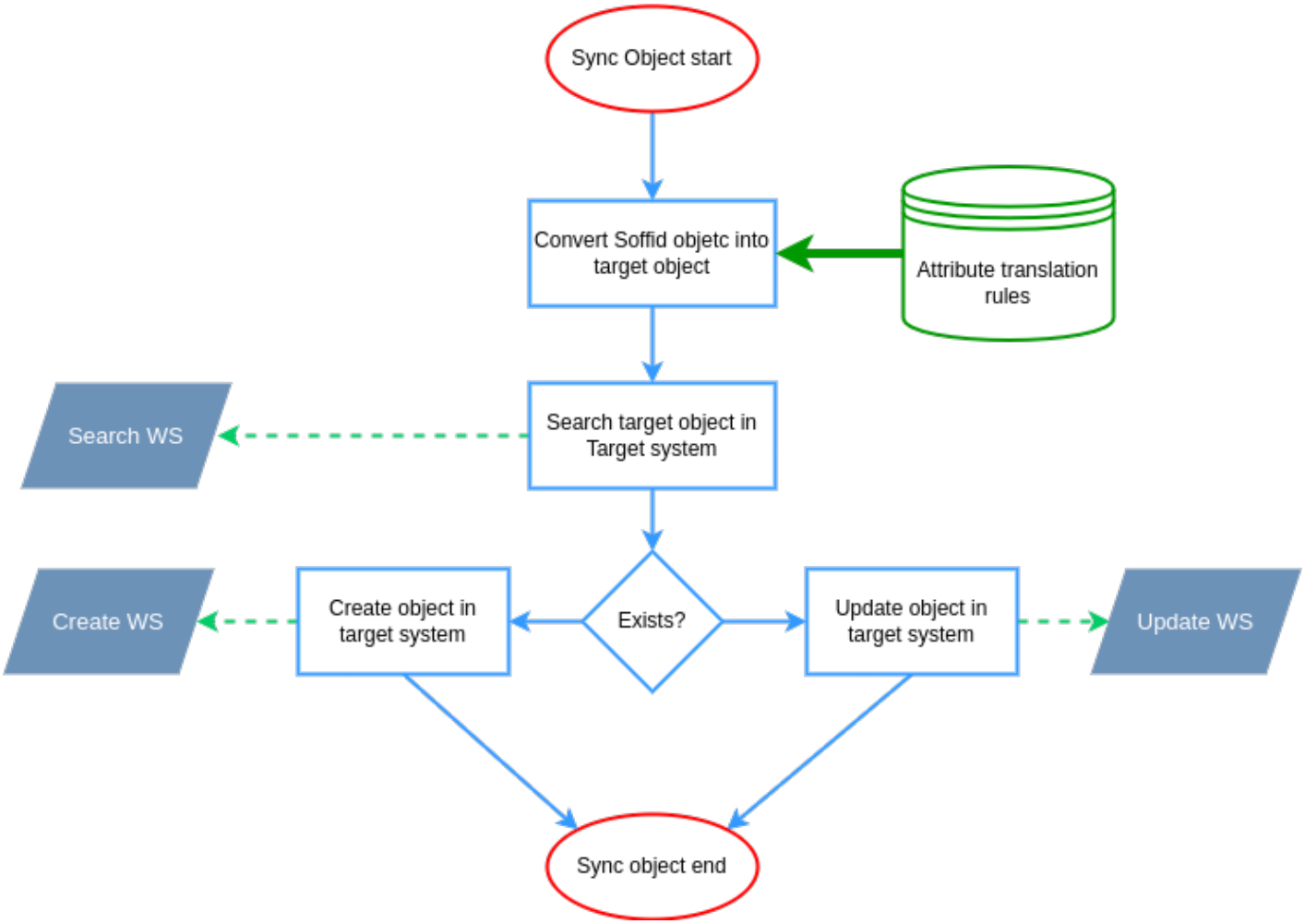
These families are involved in the following processes:

Process	Families
Reconcile automatic task	Load + select
Authoritative automatic task	Load + select
Sync new object	Select + Insert
Sync updated object	Select + Update

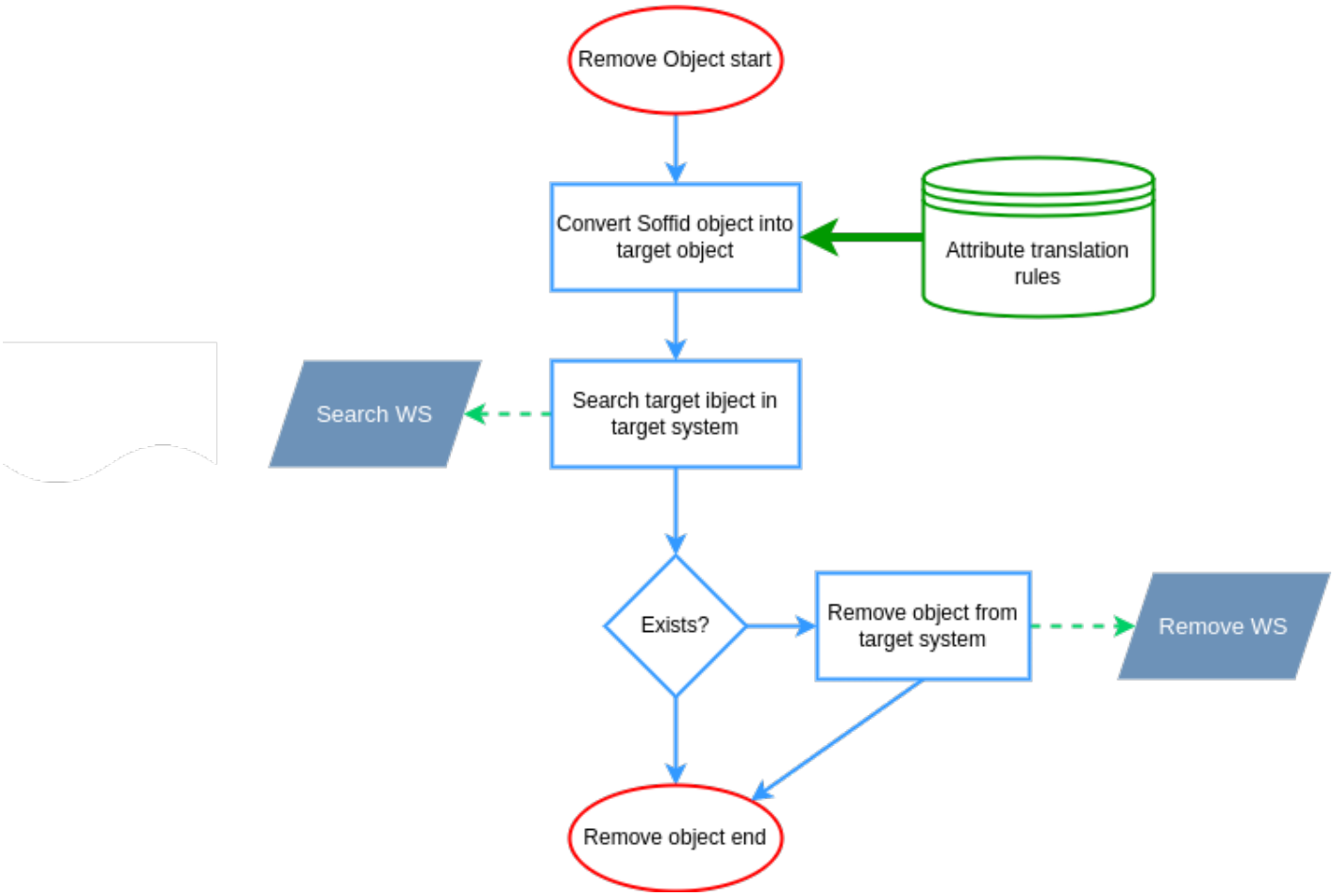
Process	Families
Sync deleted object	Select + Delete

These are the pictures of the mechanisms used to synchronize objects:

Sync object



Remove object



> You can find more information by visiting the Properties attributes page.

Attributes

You can customize attribute mappings, you only need to select system objects and the Soffid objects related, manage their attributes, and make either inbound and outbound attribute mappings.

You may map the attributes of the target system with the Soffid available attributes.

- For the target system attributes is required to be access to its specification.
- For the Soffid attributes, you may follow the next link.

> For more information about how you may configure attribute mapping, see the following link: Soffid Attribute Mapping Reference

For instance:

As an example, below is how JSON connector will look like in order to manage JIRA accounts:

Property	Value
insertEncoding	application/json
insertMethod	POST
insertPath	/Users
loadMethod	GET
loadParams	-
loadPath	/Users
selectEncoding	application/json
selectMethod	GET
selectParams	-
selectPath	/Users?filter=userName+eq+%22${userName}%22
selectResults	Resources
updateEncoding	application/json
updateMethod	PUT
updatePath	/Users/${id}

System attribute		Direction	Soffid attribute	
userName		⇄	accountName	
name{"formatted"}		⇄	fullName	
active		←	! accountDisabled	
emails		←	l = new LinkedList();	
groups		←	l = new LinkedList();	

Triggers

You can define BeanShell scripts that will be triggered when data is loaded into the target system (outgoing triggers). The trigger result will be a boolean value, true to continue or false to stop.

Triggers can be used to validate or perform a specific action just before performing an operation or just after performing an operation on target objects.

> To view some examples, visit the Outgoing triggers examples page.

JSON REST Web Services Connector - Properties

In this agent, the configuration of the properties attributes is very important due to they define the functionality of the integration:

This agent has five families of properties:

Family	Description
Load	Used to retrieve all the objects in the target system
Select	Used to retrieve an object in the target system
Insert	Used to create an object in the target system
Update	Used to update an object in the target system
Delete	Used to remove an object in the target system

These families are involved in the following processes:

Process	Families
Reconcile automatic task	Load + select
Authoritative automatic task	Load + select
Sync new object	Select + Insert
Sync updated object	Select + Update
Sync deleted object	Select + Delete

These are the properties attributes grouped by family:

Load

Property	Description
loadPath (required)	Denotes the path (relative to webserver root) where the WebService is located. It can contain variable names in the form of **${variableName}**. JSON connector will replace that name for the actual value. Eventually, complex expressions can be written in, but it's discouraged

Property	Description
loadMethod (required)	Denotes the HTTP method to use: PUT, POST, GET and DELETE are allowed
loadEncoding (required)	Type of encoded data that will be used.
loadParams (optional)	Put the character '-' in case you would avoid its value
loadTemplate (optional)	Name of the corresponding template defined on the XML Templates.
loadResults (optional)	But highly recommended) denotes the JSON portion that contains current data for the object. If this element is not present, or empty, the connector will conclude the object does not exist yet. This property will contain a simple JSON attribute name, but complex scripts are also allowed.
loadSuccessCodes (optional)	The HTTP codes to be interpreted as OK.
loadFailureCodes (optional)	The HTTP codes to be interpreted as Error.
loadNext (optional)	Next page to fetch. When the response gives us the URL of the next page to fetch, you must type the tag name of this attribute.
loadPagination (optional)	Complex script to get the next call that has to be done.
loadCondition (optional)	Script to prevent a call. To prevent the call must return false.
loadHeader (optional)	Use this property to send HTTP header(s). More than one header can be sent by adding multiple properties loadHeader1, loadHeader2, and so on. The value of the header is "HEADER:VALUE", for example "Accept:application/json".

Select

Property	Description
selectPath (required)	Denotes the path (relative to webserver root) where the WebService is located. It can contain variable names in the form of **${variableName}**. JSON connector will replace that name for the actual value. Eventually, complex expressions can be written in, but it's discouraged
selectMethod (required)	Denotes the HTTP method to use: PUT, POST, GET and DELETE are allowed

Property	Description
selectEncoding (required)	Denotes the encoding used to send to the target webservice. **application/json** and **application/x-www-form-urlencoded** are supported. The first one is used by default to POST and PUT requests. The second one is used by default for GET and DELETE requests
selectParams (optional)	Put the character '-' in case you would avoid its value
selectTemplate (optional)	Name of the corresponding template defined on the XML Templates.
selectResults (optional)	Denotes the JSON portion that contains current data for the object. If this element is not present, or empty, the connector will conclude the object does not exist yet. This property will contain a simple JSON attribute name, but complex scripts are also allowed
selectSuccessCodes (optional)	The HTTP codes to be interpreted as OK.
selectFailureCodes (optional)	The HTTP codes to be interpreted as Error.
selectNext (optional)	Next page to fetch. When the response gives us the URL of the next page to fetch, you must type the tag name of this attribute.
selectPagination (optional)	Complex script to get the next call that has to be done.
selectCondition (optional)	Script to prevent a call. To prevent the call must return false.
selectHeader (optional)	Use this property to send HTTP header(s). More than one header can be sent by adding multiple properties selectHeader1, selectHeader2, and so on. The value of the header is "HEADER:VALUE", for instance, "Accept:application/json".

Insert

Property	Description
insertPath (required)	Denotes the path (relative to webserver root) where the webservice is located.
insertMethod (required)	Denotes the HTTP method to use: PUT, POST, GET and DELETE are allowed
insertEncoding (required)	Denotes the encoding used to send to the target webservice. **application/json** and **application/x-www-form-urlencoded** are supported. The first one is used by default to POST and PUT requests. The second one is used by default for GET and DELETE requests
insertTemplate (optional)	Name of the corresponding template defined on the XML Templates.

Property	Description
insertParams (optional)	Type in the attributes that will be sent to the rest server. If this property is not set, all attributes will be sent.
insertResults (optional)	Denotes the JSON portion that contains current data for the object. If this element is not present, or empty, the connector will conclude the object does not exist yet. This property will contain a simple JSON attribute name, but complex scripts are also allowed
insertSuccessCodes (optional)	The HTTP codes to be interpreted as OK.
insertFailureCodes (optional)	The HTTP codes to be interpreted as Error.
insertCondition (optional)	Script to prevent a call. To prevent the call must return false.
insertHeader (optional)	Use this property to send HTTP header(s). More than one header can be sent by adding multiple properties insertHeader1, insertHeader2, and so on. The value of the header is "HEADER:VALUE", for example "Accept:application/json".

Update

Property	Description
updatePath (required)	Denotes the path (relative to webserver root) where the webservice is located
updateMethod (required)	Denotes the HTTP method to use: PUT, POST, GET and DELETE are allowed
updateEncoding (required)	Denotes the encoding used to send to the target webservice. **application/json** and **application/x-www-form-urlencoded** are supported. The first one is used by default to POST and PUT requests. The second one is used by default for GET and DELETE requests
updateParams (optional)	Type in the attributes that will be sent to the rest server. If this property is not set, all attributes will be sent.
updateResults (optional)	Denotes the JSON portion that contains current data for the object. If this element is not present, or empty, the connector will conclude the object does not exist yet. This property will contain a simple JSON attribute name, but complex scripts are also allowed

Property	Description
updateSuccessCodes (optional)	The HTTP codes to be interpreted as OK.
updateFailureCodes (optional)	The HTTP codes to be interpreted as Error.
updateCondition (optional)	Script to prevent a call. To prevent the call must return false.
updateHeader (optional)	Use this property to send HTTP header(s). More than one header can be sent by adding multiple properties updateHeader1, updateHeader2, and so on. The value of the header is "HEADER:VALUE", for example "Accept:application/json".

Delete

Property	Description
deletePath (required)	Denotes the path (relative to webserver root) where the webservice is located
deleteMethod (required)	Denotes the HTTP method to use: PUT, POST, GET and DELETE are allowed
deleteEncoding (required)	Denotes the encoding used to send to the target webservice. **application/json** and **application/x-www-form-urlencoded** are supported. The first one is used by default to POST and PUT requests. The second one is used by default for GET and DELETE requests
deleteParams (optional)	Type in the attributes that will be sent to the rest server. If this property is not set, all attributes will be sent.
deleteResults (optional)	Denotes the JSON portion that contains current data for the object. If this element is not present, or empty, the connector will conclude the object does not exist yet. This property will contain a simple JSON attribute name, but complex scripts are also allowed
deleteSuccessCodes (optional)	The HTTP codes to be interpreted as OK.
deleteFailureCodes (optional)	The HTTP codes to be interpreted as Error.
deleteCondition (optional)	Script to prevent a call. To prevent the call must return false.
deleteHeader (optional)	Use this property to send HTTP header(s). More than one header can be sent by adding multiple properties deleteHeader1, deleteHeader2, and so on. The value of the header is "HEADER:VALUE", for example "Accept:application/json".

How to retrieve data from the response with the *Results properties

a) One level

If the JSON has one level you have to avoid the property

```
{
   "userName" : "soffid"
}
```

b) Two level

If the JSON has two levels you have to create the property *Result and put the name of the parent attribute, for example:

```
{
   "user" : {
      "userName" : "soffid"
   }
}
```

And the property must be for example loadResults = user

c) More than two levels

If the JSON has more than two levels you have to create the property *Result and put the atributes in the next pattern

*Results = attribure1{"attribute2"}{"attribute3"}...

For example:

```
{
"data" : {
      "user" : {
         "userName" : {
            "string" : "soffid"
         }
      }
   }
}
```

And the property must be for example:

loadResults = data{"user"}{"userName"}

How to configure the Office 365 agent?

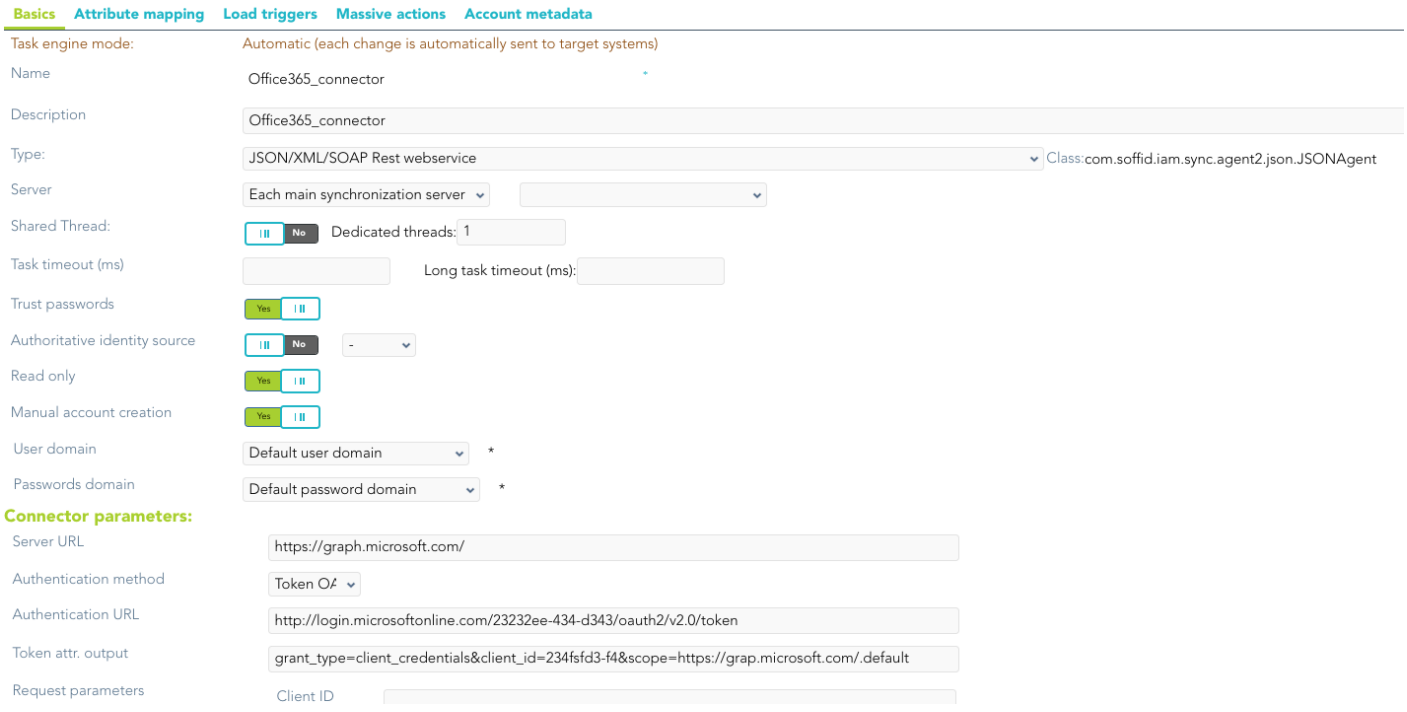
Office 365 integration

Prerequisites

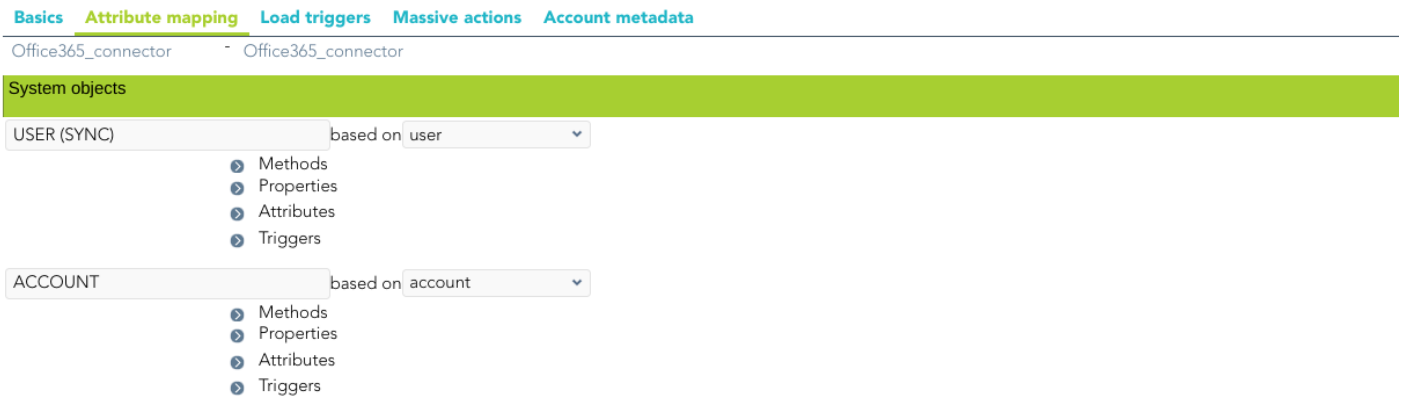
- You need to install the last version of JSON Rest Connector

Configuration

Configure the Basic data to establish the connection



Then, configure the attribute mappings



> Soffid provides you versions of the attribute mappings to import into the agent configuration:

- *Basic* attribute mappings: agent-config-Office365-Basic.xml
- Attribute mappings with *immutable ID and Azure groups*: agent-config-Office365-MoreComplex.xml

How to configure the Jira Atlassian agent?

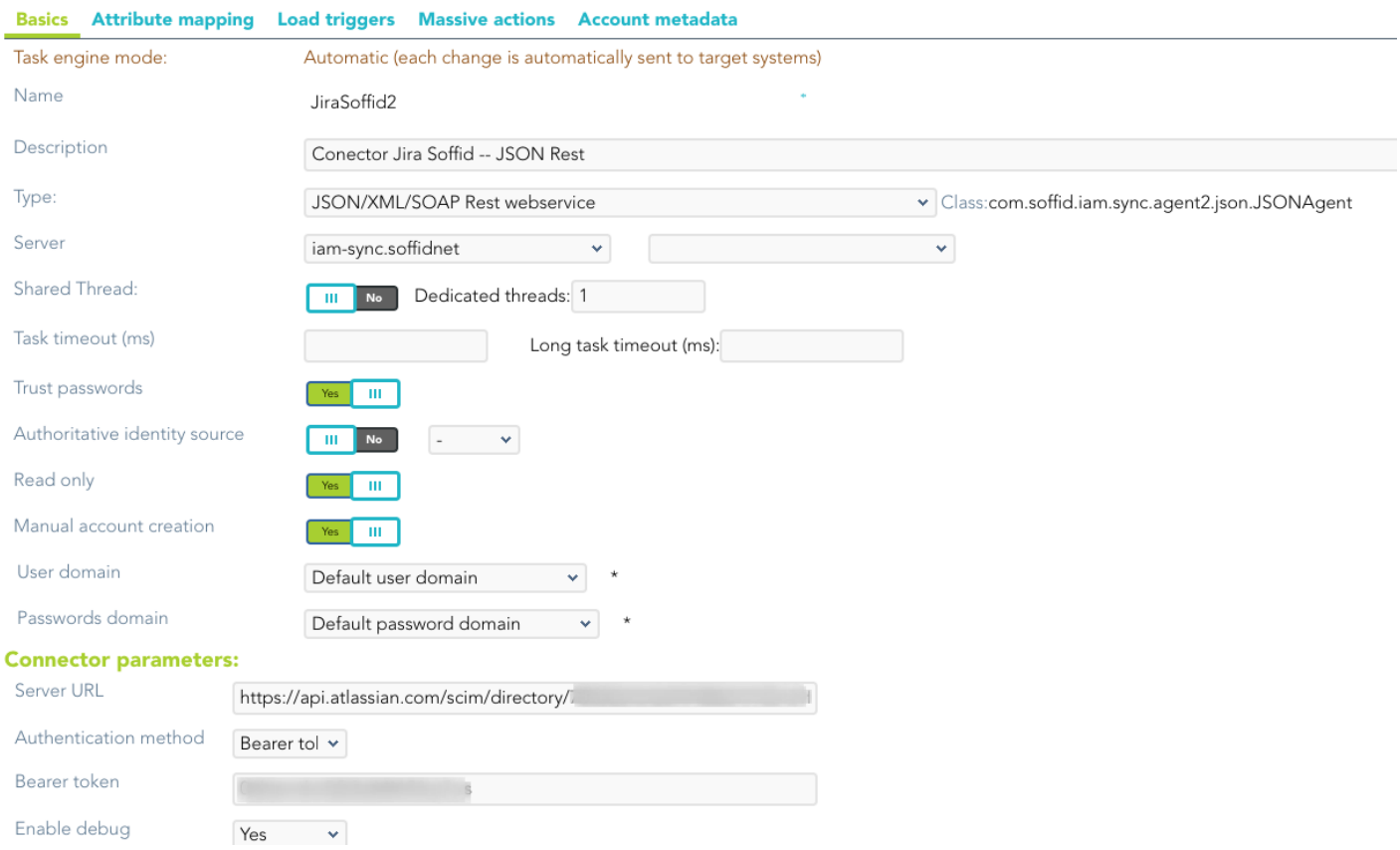
Jira integration

Prerequisites

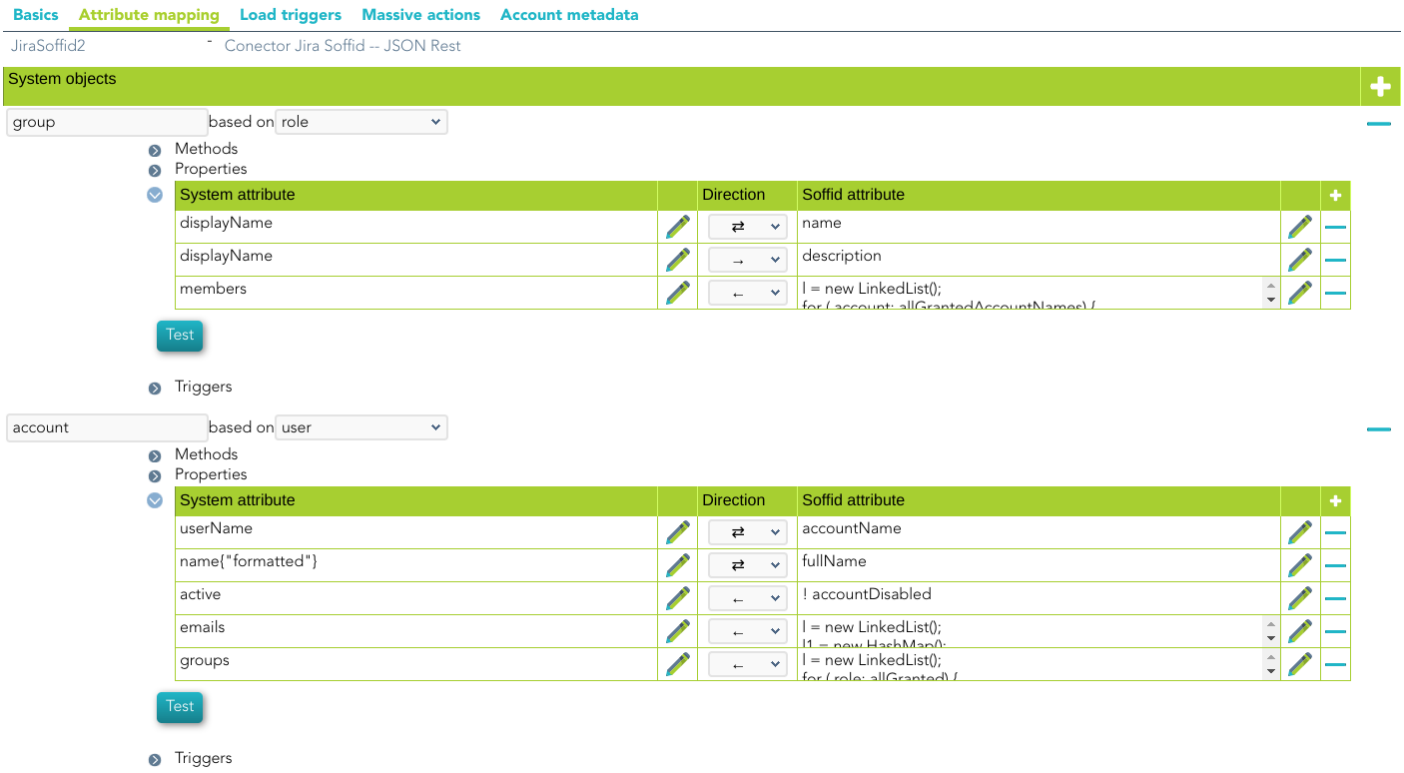
- You need to install the last version of JSON Rest Connector.

Configuration

Configure the Basic data to establish the connection



Then, configure the attribute mappings



Soffid provides you an XML file with the basic attribute mappings to import into the agent configuration JIRA Soffid agent-config.xml