JSON REST Web Services Connector

Table of Contents

Introduction

Description

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

Managed Systems

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

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

- JIRA
- Oracle Filed Service Cloud (OFSC)
- Office 365
- Dropbox

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.

For download and install the addon you could review our generic documentation about this process: Addons installation

Agent configuration

Basics

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
User name	User to authenticate
Password	Password of the user to authenticate
Authentication method	Three options: • "None": no authentication (User and Password are not used) • "Basic": it uses the User and Password to generate the authentication token • "Token": generate a token from a specific authentication URL
Authentication URL	URL to retrieve the token for the authentication of the server (for "Token" method)
Enable debug	Two options: "Yes", "No": it enables or not more log traces in the Synchronization Server log

Attribute mapping

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

Properties

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

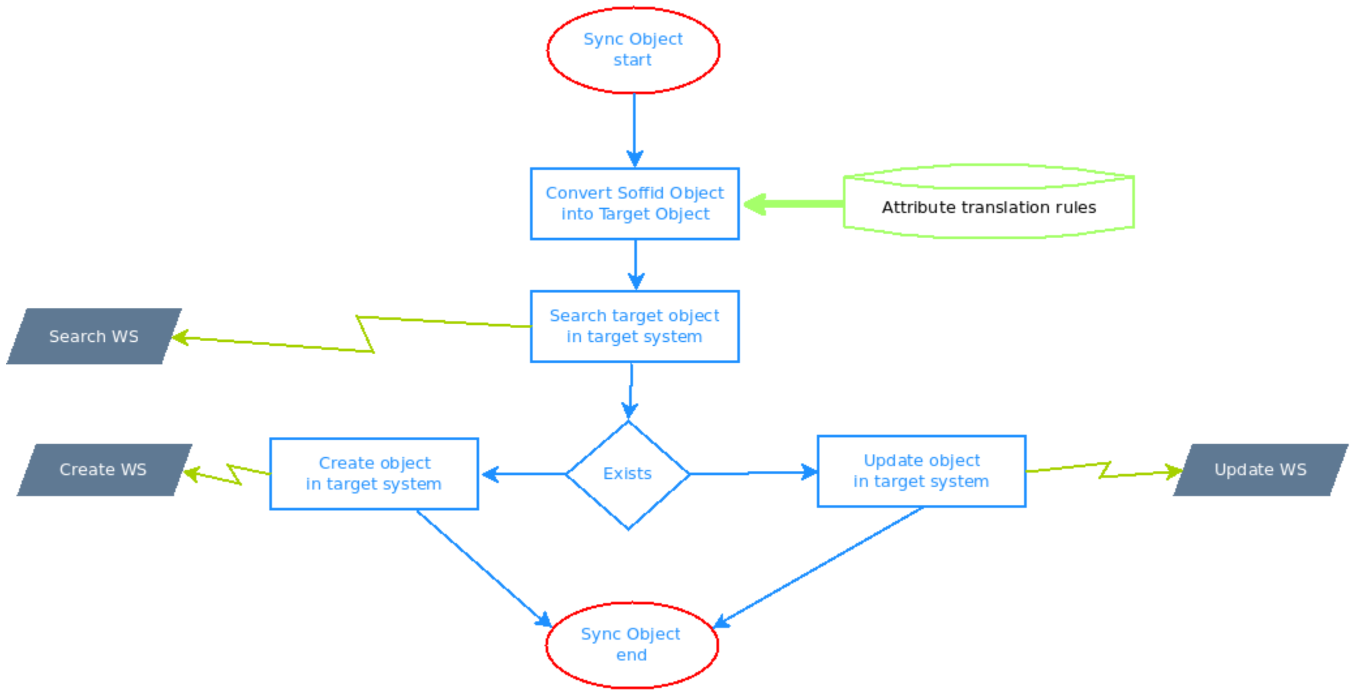
This agent has five families of properties:

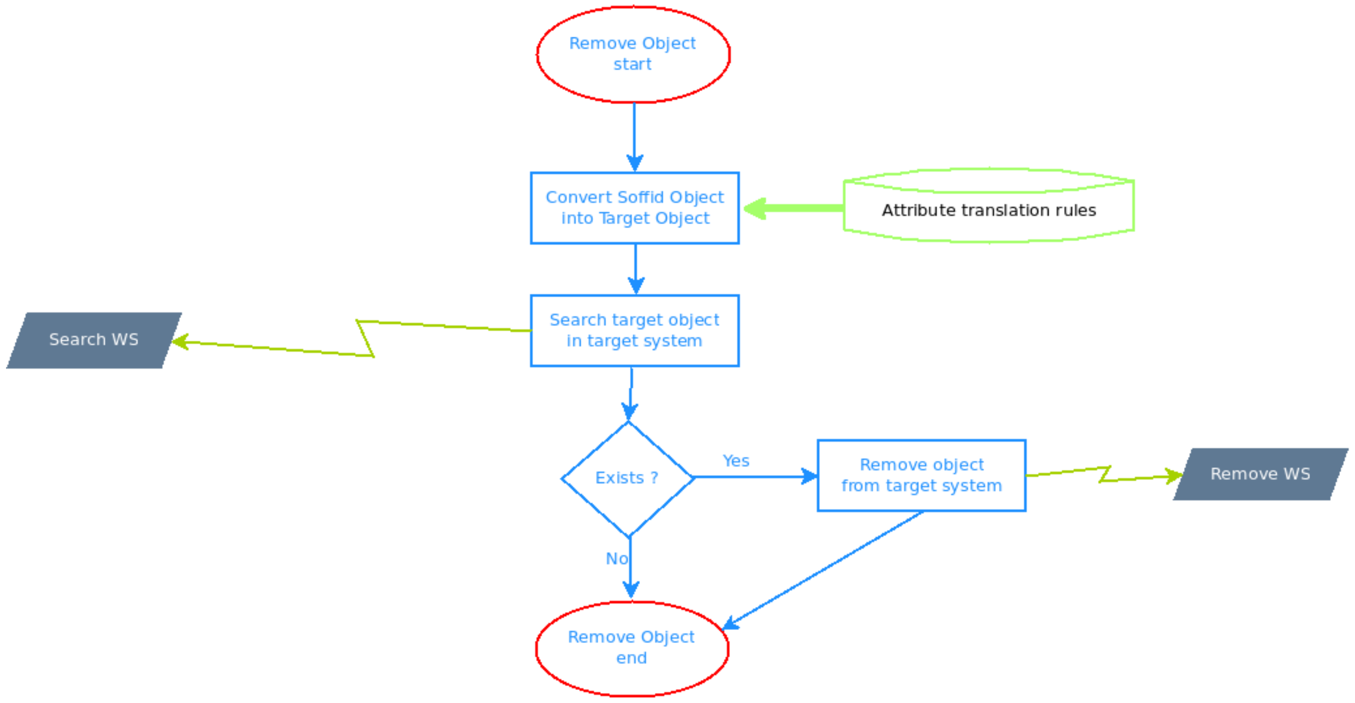
Family	Description
Load	Used to retrieve all the objects in the target system
Select	Used to retrieve an object in the target system
Insert	Used to create an object in the target system
Update	Used to update an object in the target system
Delete	Used to remove an object in the target system

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 pictures of the mechanisms used to synchronize objects:


Sync Object start
Convert Soffid Object into Target Object
Attribute translation rules
Search WS
Search target object in target system
Create WS
Create object in target system
Exists
Update object in target system
Update WS
Sync Object end


Remove Object start
Convert Soffid Object into Target Object
Attribute translation rules
Search WS
Search target object in target system
Exists ?
Yes
Remove object from target system
Remove WS
No
Remove Object end

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

loadMethod (required)	Denotes the HTTP method to use: PUT, POST, GET and DELETE are allowed
loadParams (required)	Put the character '-' in case you would avoid its value
loadCheck (optional)	Denotes a script that will check wether the invokation has been successful or not. Each json attribute received from target web service will be available as context variables
loadResults (optional)	But highly recommended) denotes the json portion that containes current data for the user. It this element is not present, or empty, the connector will conclude the user does not exist yet. This property will contain a simple json attribute name, but complex scripts are also allowed
loadHeader (optional)	Optional HTTP header(s) to send. More than one header can be sent by adding multiple propertis .....Header1, .Header2, and so on

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
selectEncoding (optional)	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
selectCheck (optional)	Denotes a script that will check whether the invocation has been successful or not. Each JSON attribute received from target web service will be available as context variables
selectResults (optional)	Denotes the JSON portion that contains current data for the user. It this element is not present, or empty, the connector will conclude the user does not exist yet. This property will contain a simple JSON attribute name, but complex scripts are also allowed
selectHeader (optional)	Optional HTTP header(s) to send. More than one header can be sent by adding multiple propertis .....Header1, .Header2, and so on

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 (optional)	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
insertCheck (optional)	Denotes a script that will check whether the invocation has been successful or not. Each json attribute received from target web service will be available as context variables
insertHeader (optional)	Optional HTTP header(s) to send. More than one header can be sent by adding multiple propertis .....Header1, .Header2, and so on
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.

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 (optional)	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
updateCheck (optional	Denotes a script that will check whether the invocation has been successful or not. Each JSON attribute received from target web service will be available as context variables
updateHeader (optional)	Optional HTTP header(s) to send. More than one header can be sent by adding multiple propertis .....Header1, .Header2, and so on
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.

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 (optional)	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
deleteCheck (optional)	Denotes a script that will check wether the invokation has been successful or not. Each json attribute received from target web service will be available as context variables
deleteHeader (optional)	Optional HTTP header(s) to send. More than one header can be sent by adding multiple propertis .....Header1, .Header2, and so on
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.
preventDeletion (required)	Set to false to enable delete method

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"}
```

Attributes

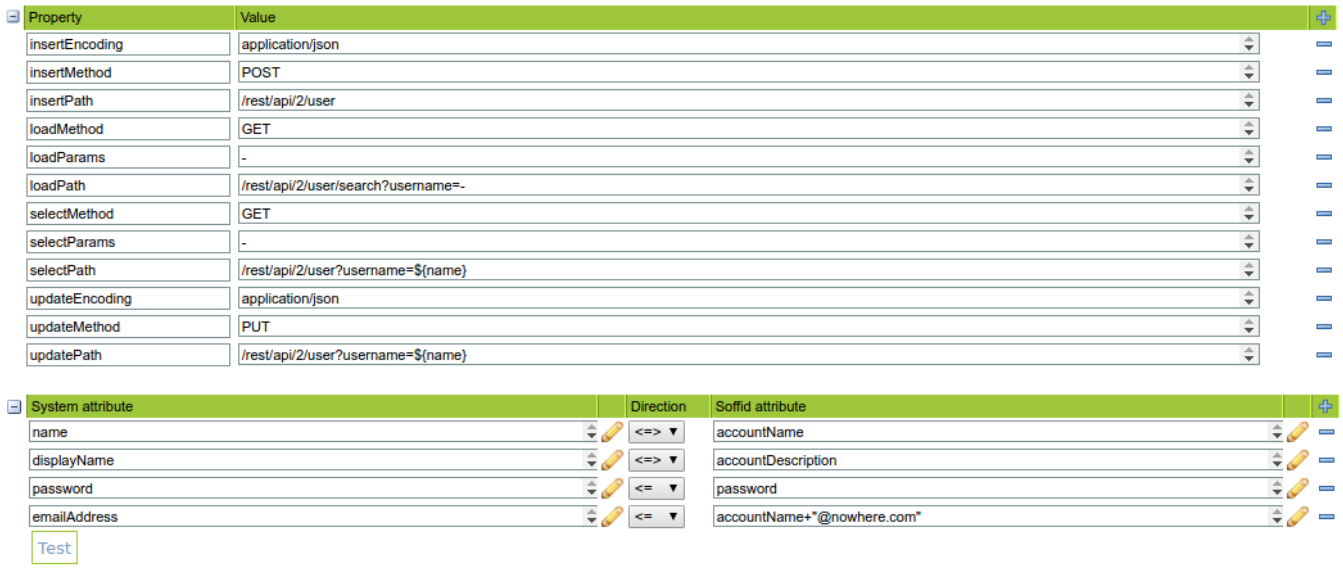
You may map the attributes of the target system with the Soffid available attributes.

- 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 example:

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	/rest/api/2/user
loadMethod	GET
loadParams	-
loadPath	/rest/api/2/user/search?username=-
selectMethod	GET
selectParams	-
selectPath	/rest/api/2/user?username=${name}
updateEncoding	application/json
updateMethod	PUT
updatePath	/rest/api/2/user?username=${name}

System attribute	Direction	Soffid attribute
name	<=>	accountName
displayName	<=>	accountDescription
password	<=	password
emailAddress	<=	accountName+"@nowhere.com"

Triggers

Pending to be documented.

Load triggers

Pending to be documented.

Account metadata

Pending to be documented.

Operational

Monitoring

After the agent configuration you could check in the monitoring page if the service is running in the Synchronization Server, please go to "*Start Menu > Monitoring and reporting > System monitoring*".

Tasks

Authoritative

If you are checked *"Authorized identity source*", an automatic task to load identities from the managed system to Soffid is available, please go to *"Start Menu > Processes and Tasks > Manage automatic tasks", and you will something like "Import authoritative data from <AGENT_NAME>*".

Reconcile

If your are configured the "*Attribute Mapping*" tab with some of our objects: "*user, account, role, group or grant*", an automatic task to synchronize these objects from the managed system to Soffid is available, please go to *"Start Menu > Processes and Tasks > Manage automatic tasks", and you will something like* "*Reconcile all accounts from <AGENT_NAME>*".

Synchronization

About the synchronization of the objects, there are two possible options:

- If you are checked the generic attribute *"Read Only*" in the "*Basics*" tab, only the changes in the managed systems will be updated in Soffid. We recommend this options until the global configuration of Soffid will be tested.
- If you are not checked the generic attribute *"Read Only*" in the "*Basics*" tab, all the changes in Soffid or the managed system will be updated in the other. Note that this synchronization must be configured in the "Attribute mapping" tab correctly.