Accessing SAP SuccessFactors from Denodo

Revision 20230328

Denodo Inc.: 525 University Avenue, Suite 31, Palo Alto, CA 94301. USA www.denodo.com

CONTENTS

1 INTRODUCTION

SAP SuccessFactors is a SaaS solution for Human Resource management, the tool is part of the SAP suite from 2011 and it provides features in order to support HR processes.

As part of a business requirement, you may need to integrate the SAP SuccessFactors information with other data sources, and for that, you can use the Denodo Platform. SAP SuccessFactors provides an ODATA API v2 that you can use to connect Denodo to SuccessFactors in order to retrieve data from it.

In this document we will explain how to access the SAP SuccessFactors API from Denodo and we will use an example of how to retrieve users information using this API like a REST API.

2 ACCESSING THE API

2.1 OAUTH CONNECTION

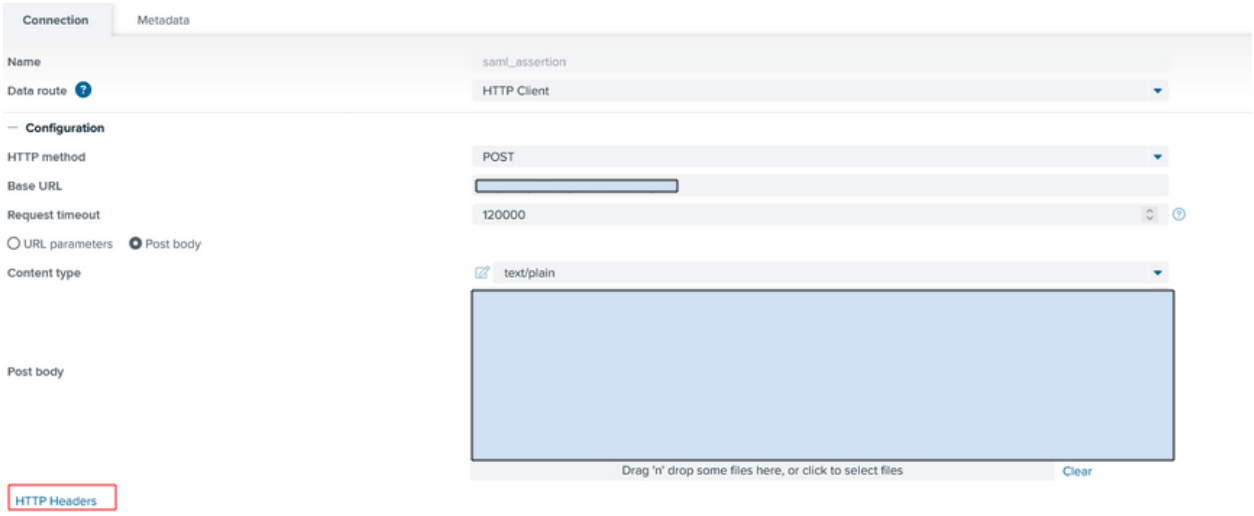
According to the technical article of the SAP community OAuth connection to SuccessFactors Employee Central to access the API we have to follow these steps: we need to register an OAuth2 Client and then generate a SAML assertion to request a user token using this assertion. The referenced document provides more insight about how the process works. In this document, we will explain how to achieve this using Denodo.

2.1.1 Generate a SAML assertion

First, we need to generate a SAML assertion. To do that, we have to create a Delimited file data source (`saml_assertion`) because this API endpoint returns a response with just a single text value. To create the data source go to the Server Explorer and then to `New > Data source > Delimited file`.

In the dialog to create the data source:

1. `Data route`: `HTTP client`.
2. HTTP method: `POST`
3. Base URL: `https://<server>/oauth/idp`. Here, we need to specify our API server.
4. Check "`Post body`".
5. Content type: `text/plain`.
6. Post body: `client_id=<client_id>&user_id=<api_user>&token_url=<token_url>&private_key=<private key>`. In the Post body we have to indicate the next parameters:
    a. **client_id**: this is the API Key.
    b. **user_id**: Interface User ID to call the API.
    c. **token_url**: `https://<server>/outh/token`. Here, we need to specify our API server.
    d. **private_key**: this is the private key from the X.509 certificate.
7. Finally, we need to create a HTTP Header so we click on "`HTTP headers`" and we create a header with name "`Content-Type`" and value "`application/x-www-form-urlencoded`". Also, we set a comma as column delimiter and click on "`Save`".



2.1.1.1 SAML Base view

After creating the data source `saml_assertion` we need to create a base view to obtain the assertion. So, we click on "`Create Base View`". From this base view we get the assertion that we will need it in the next step.

2.1.2 Request a User Token using the SAML Assertion

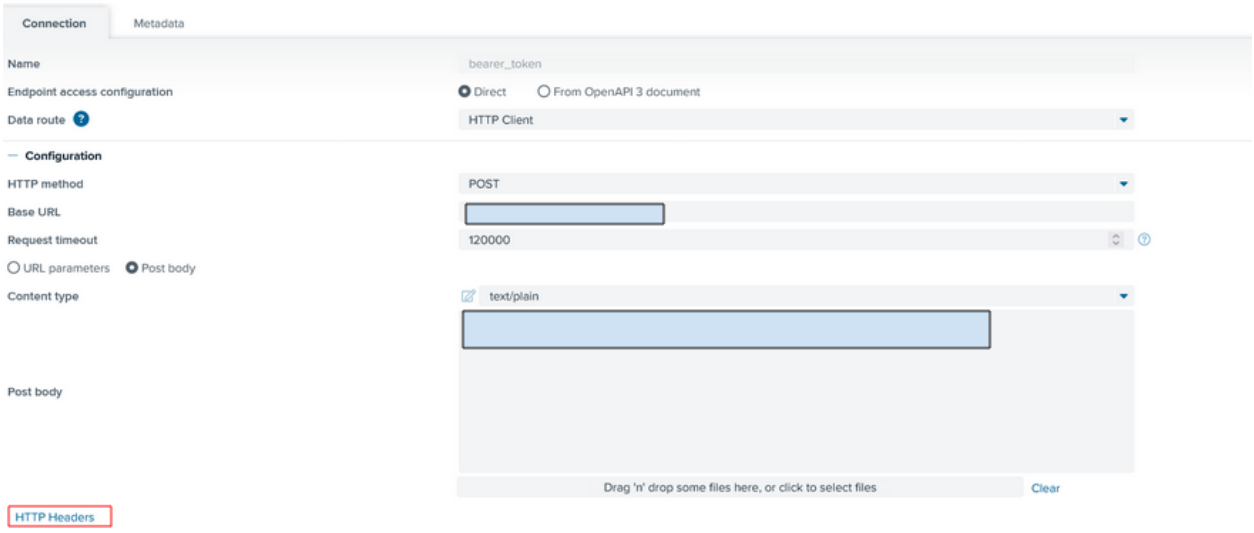
In order to get a bearer token to access the API, we have to create a JSON data source, because the request that we have to use returns the response in JSON format. To create it, go to the Server Explorer and then to `New > Data source > JSON`.

1. `Data route`: `HTTP client`.
2. HTTP method: `POST`
3. Base URL: `https://<server>/oauth/token`. Here, we need to specify our API server and also, this URL is the token url that we have used before.
4. Check "`Post body`".
5. Content type: `text/plain`.
6. Post body:
   `company_id=<company_id>&client_id=<client_id>&grant_type=urn:ietf:params:oauth:grant-type:saml2-bearer&assertion=@assertion`

   In the Post body we have to establish the next parameters:

   a. **company_id**: SuccessFactors Company ID.
   b. **client_id**: this is the API Key.
   c. **assertion**: this is the SAML assertion that we can get from `bv_saml_assertion`. We will use it later as an interpolation variable with the assertion of that view.
7. Finally, we need to create a HTTP Header so we click on "`HTTP headers`" and we

create a header with name “Content-Type” and value “application/x-www-form-urlencoded”. Then, click on “Save”.



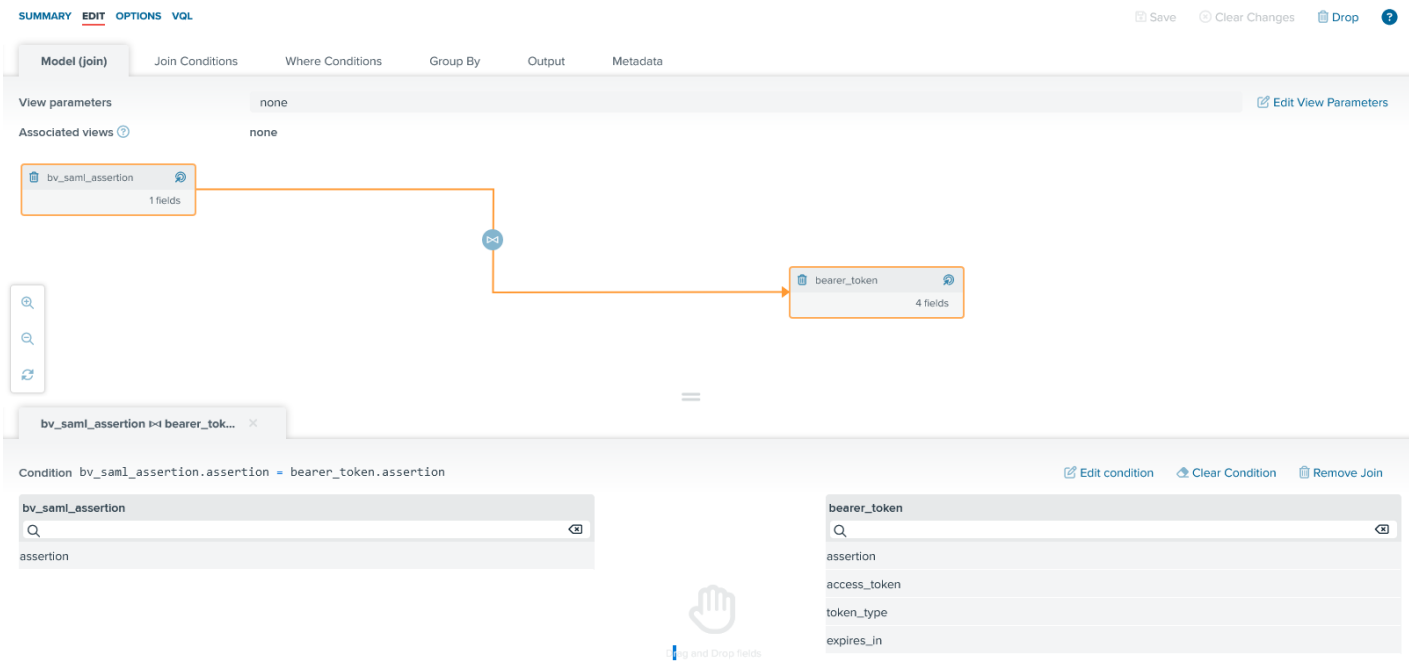
2.1.2.1 Base view

After creating the data source bearer_token we can create a base view in order to get the token. So, we click on “Create Base View”, introduce the assertion value for the interpolation variable and confirm the next steps.

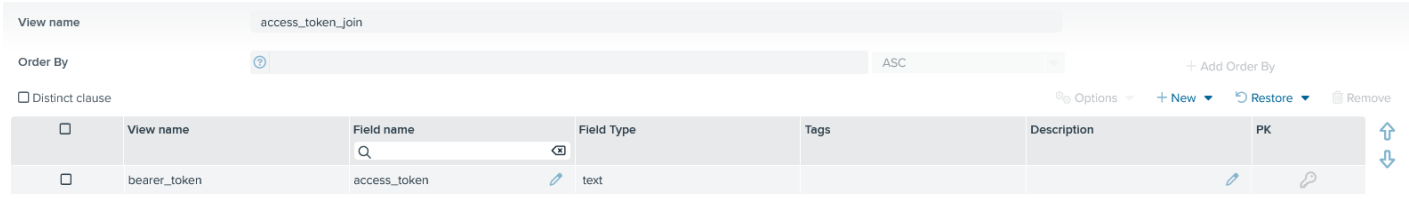
2.1.2.2 JOIN View

With the purpose of automating the process of introducing an assertion each time that we want to get the access token, we can make a join view with the views bv_saml_assertion and bv_bearer_token in order to associate the assertion with the assertion parameter. Click on the base view bv_saml_assertion and go to New > Join.

1. We have to make an inner join where the join conditions are: bv_saml_assertion.assertion = bearer_token.assertion



3. Finally, we can remove in the output section all the fields except the field `access_token` which contains the bearer token.



We get the result shown in the screenshot.

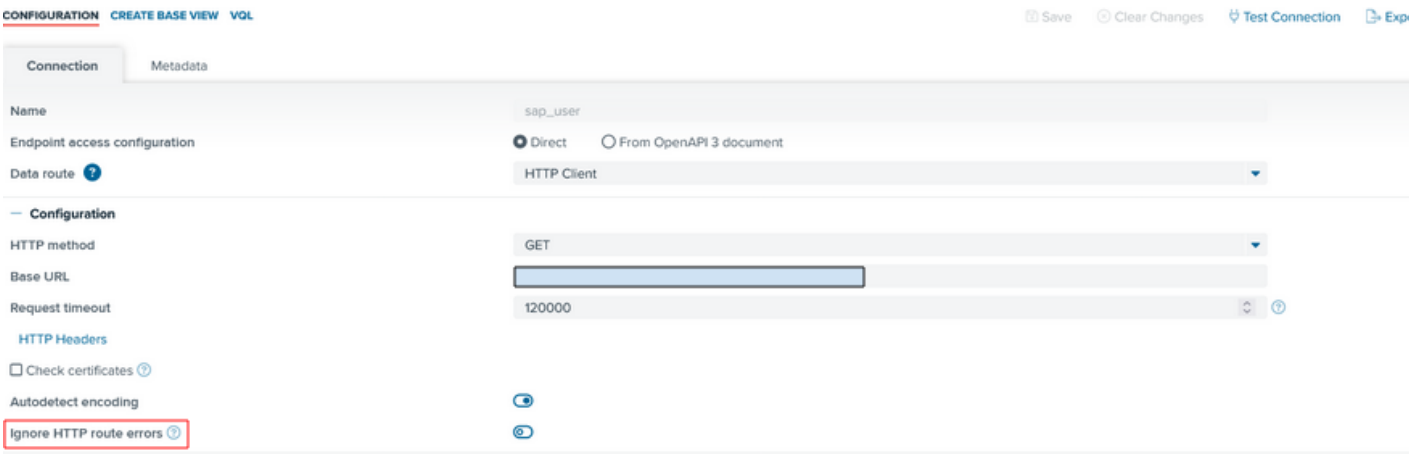
2.2 USING THE API

2.2.1.1 Data source

Now, we are going to use the API to retrieve users' information. So, first we need to create a JSON data source. Go to the Server Explorer and then to `New > Data source > JSON`.

1. `Data route`: `HTTP client`.

2. HTTP method: `GET`.
3. Base URL: `https://<server>/odata/v2/User?$format=json`. Here, we need to specify our API server.
4. Finally, we need to create a HTTP Header so we click on "`HTTP headers`" and we create a header with name "`Authorization`" and value "`Bearer @bearer_token`". Then, click on "`Save`".



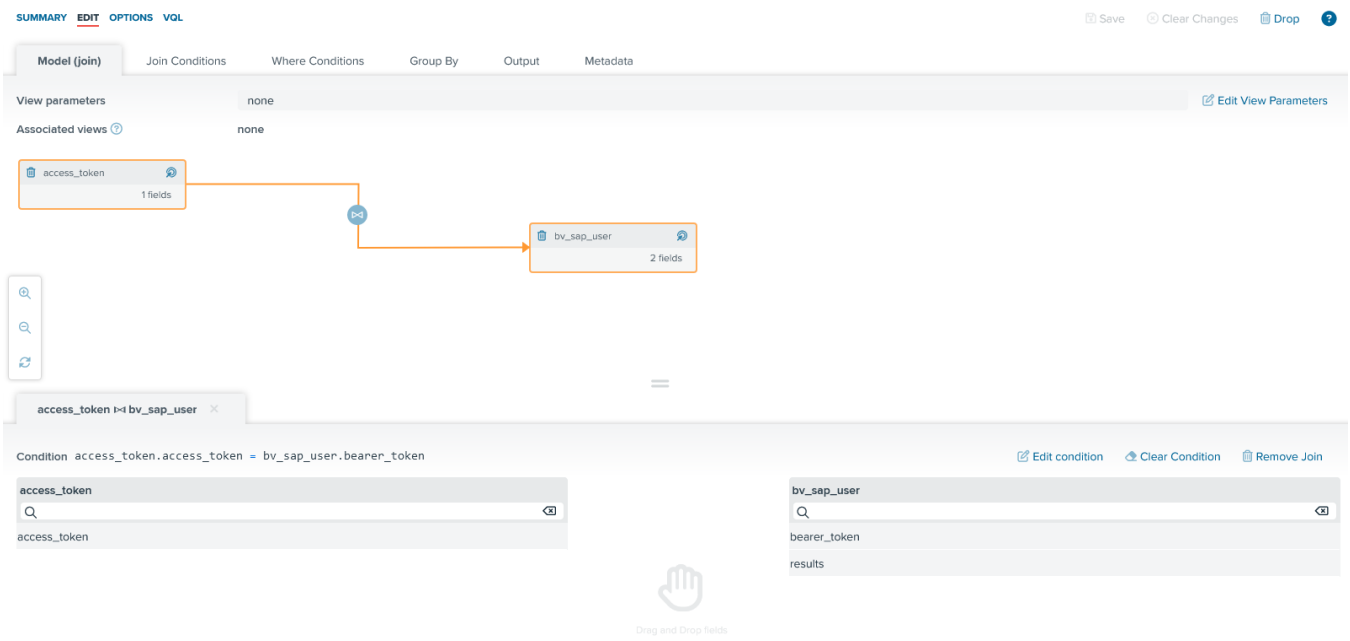
2.2.1.2 Base View

We have to create a base view to retrieve the information of the users. From the data source click on "`Create Base View`" and introduce the interpolation variable which is the Bearer token that we have obtained in the previous join view.

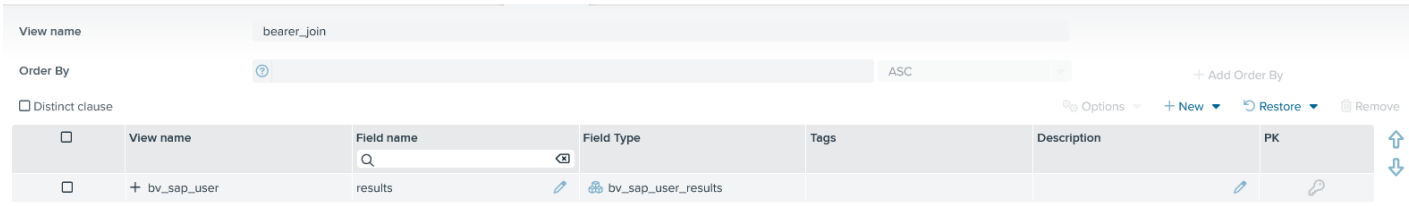
2.2.1.3 Join View

Once the users base view is created, we can make a join view to associate the bearer token parameter with the bearer token that we can obtain from the previous join view. Click on the access token join view and go to `New > Join`.

1. We have to make an inner join where the join conditions are: `access_token.acces_token = bv_sap_user.bearer_token`.

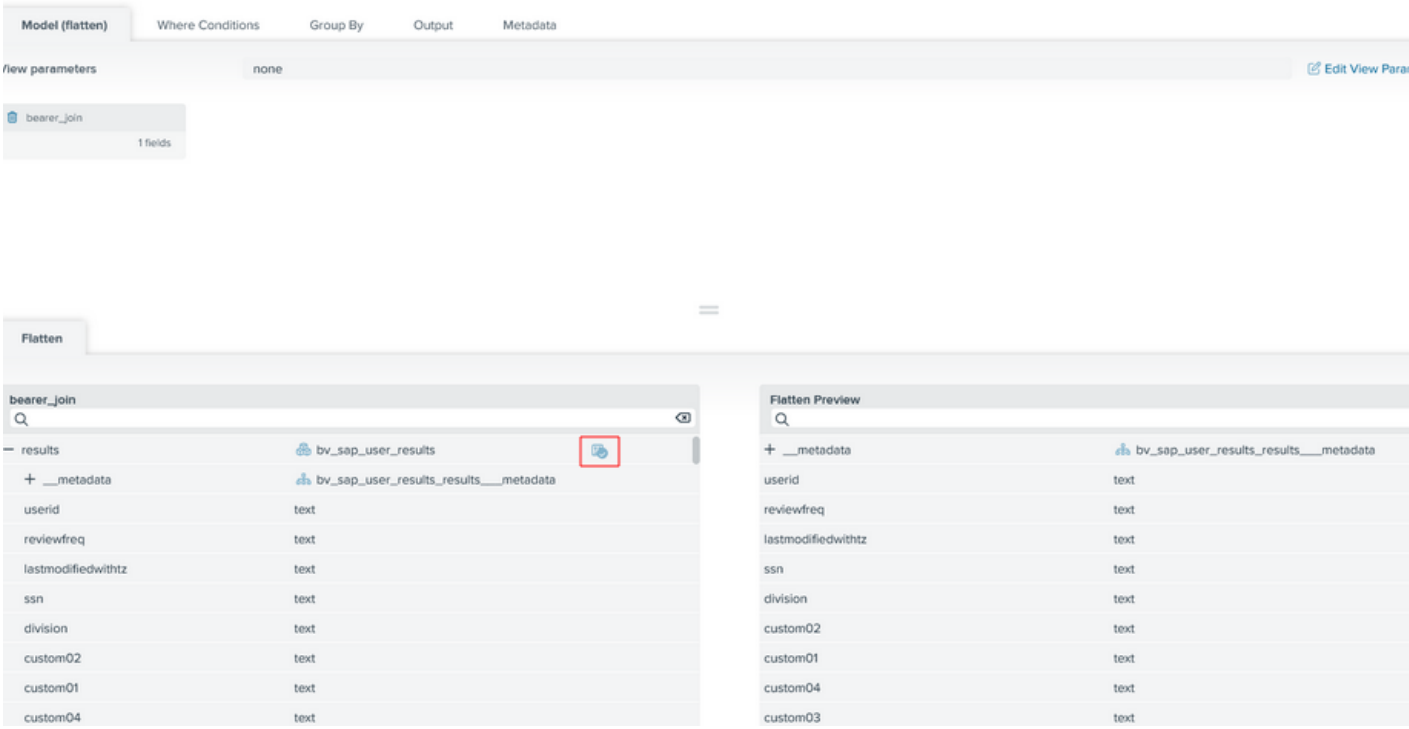


2. Finally, we remove in the output section the token fields.



2.2.1.4 Final result

Finally, we will create a flatten view from the join view to get the results flatten. From the join view `New > Flatten`. There, click on the results field to flatten the array.



In our example, we are going to remove from the output some fields that we are not interested in. You can select the fields in the output section and there you check in the field that you want to remove and finally click on "Remove". We have selected the fields shown in the screenshot below.

☐	Field name	Field Type
☐	userid	text
☐	defaultlocale	text
☐	lastname	text
☐	loginmethod	text
☐	status	text
☐	email	text
☐	country	text
☐	firstname	text
☐	timezone	text
☐	username	text
☐	displayname	text
☐	defaultfullname	text

If we execute the view we obtain the result shown in the image. The values for some of the columns are masked. You can obtain information about the users like: the ID of the user, the last name, the login method, the email, etc.

RESULTS EXECUTION TRACE — Refresh | Copy results to clipboard | Save trace | Copy trace to clipboard

13 rows 12 columns received SELECT * FROM admin.user_final_result CONTEXT('cache_wait_for_load' = 'true') TRACE

userid	defaultlocale	lastname	loginmethod	status	email	country	firstname	timezone	username	displayname	defaultfullname
<null>	en_US	<null>	PWD	t	<null>	<null>	<null>	US/Eastern	<null>	<null>	<null>
<null>	en_US	<null>	PWD	t	<null>	SPAIN	<null>	US/Eastern	<null>	<null>	<null>
<null>	en_US	<null>	PWD	t	<null>	SPAIN	<null>	US/Eastern	<null>	<null>	<null>
<null>	en_US	<null>	PWD	t	<null>	SPAIN	<null>	US/Eastern	<null>	<null>	<null>
<null>	en_US	<null>	PWD	t	<null>	SPAIN	<null>	US/Eastern	<null>	<null>	<null>
<null>	en_US	<null>	PWD	t	<null>	SPAIN	<null>	US/Eastern	<null>	<null>	<null>
<null>	en_US	<null>	PWD	t	<null>	SPAIN	<null>	US/Eastern	<null>	<null>	<null>
<null>	en_US	<null>	PWD	t	<null>	SPAIN	<null>	US/Eastern	<null>	<null>	<null>
<null>	en_US	<null>	PWD	t	<null>	SPAIN	<null>	US/Eastern	<null>	<null>	<null>
<null>	en_US	<null>	PWD	t	<null>	SPAIN	<null>	US/Eastern	<null>	<null>	<null>
<null>	en_US	<null>	PWD	t	<null>	<null>	<null>	US/Eastern	<null>	<null>	<null>
<null>	en_US	<null>	PWD	t	<null>	<null>	<null>	US/Eastern	<null>	<null>	<null>
<null>	en_US	<null>	PWD	t	<null>	<null>	<null>	US/Eastern	<null>	<null>	<null>

3 REFERENCES

OAuth connection to SuccessFactors Employee Central
2613670 - What are the available APIs for SuccessFactors?
SAP SuccessFactors API documentation