Contents

[**Setting Up OAuth Authentication in SAP SuccessFactors** 2](#_Toc140155188)

[Register New Client Application in SAP SuccessFactors 2](#_Toc140155189)

[**1.** **Turn on the permission for Manage OAuth2 Client Application:** 2](#_Toc140155190)

[**2.** **Add register the Client Application.** 3](#_Toc140155191)

[**3.** **Generate the Certificate** 5](#_Toc140155192)

[**Establish a connection in Postman.** 7](#_Toc140155193)

[Obtain the Assertion token. 7](#_Toc140155194)

[Obtain the OAuth Token 11](#_Toc140155195)

[Use the Access Token 14](#_Toc140155196)

# **Setting Up OAuth Authentication in SAP SuccessFactors**

To Set up connection between services and SAP SuccessFactors, you need to set up OAuth Authentication in SAP SuccessFactors. OAuth is the authentication protocol.

To Set up OAuth Authentication, you need to complete the following procedures.

## Register New Client Application in SAP SuccessFactors

Registering the service in the SAP SuccessFactors system. This enables authentication for OData API Access using OAuth 2.0

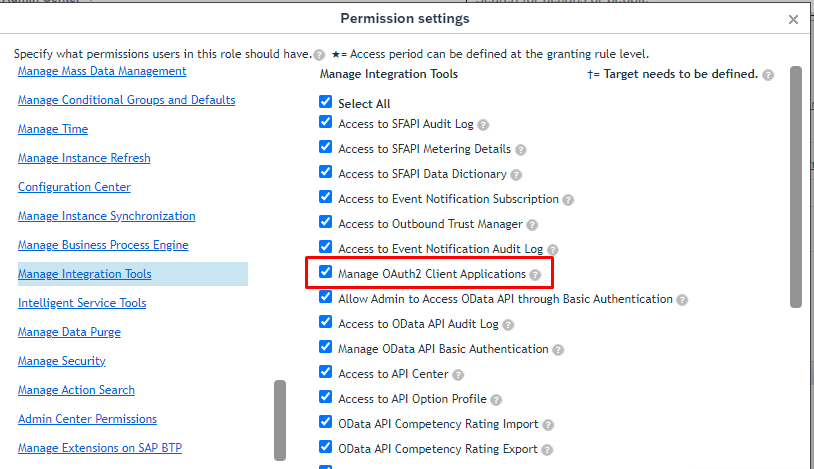
Steps to Register New Client Application n SAP SuccessFactors is:

1. Turn on the permission for OAuth2 Client Application
2. Add register the Client Application.
3. Generate the Certificate

### **Turn on the permission for Manage OAuth2 Client Application:**

This permission is given to user, so they have access to the OAuth client application.

1. Login to SAP SuccessFactors Instance
2. Navigate to the Admin Centre
3. In the Tools Search field, type **Manage Permission Roles**
4. In the Permission Settings section, click Permissions. The Permission Settings page is displayed.
5. In the Administrator Permission section, click **Manage Integration Tools** and then select Manage Integration Tools and select **Manage OAuth2 Client Applications.**



Navigate to the Admin Centre. In the Tools Search field, enter OAuth and choose **Manage OAuth2 Client Applications** from search result list.

### **Add register the Client Application.**

1. Click on the **Register Client Application** in the Upper Right corner.

A white rectangular object with a black text

Description automatically generated

1. Enter the Relevant Information

* Company: This field is automatically filled with the current company name
* Application Name: Enter a description (optional)
* Application URL: <https://localhost/>

A screenshot of a computer

Description automatically generated

After adding the details click on Generate X.509 certificate and in next step it will ask to fill the other details. Fill in the details as shown below and click on Generate.

A screenshot of a computer

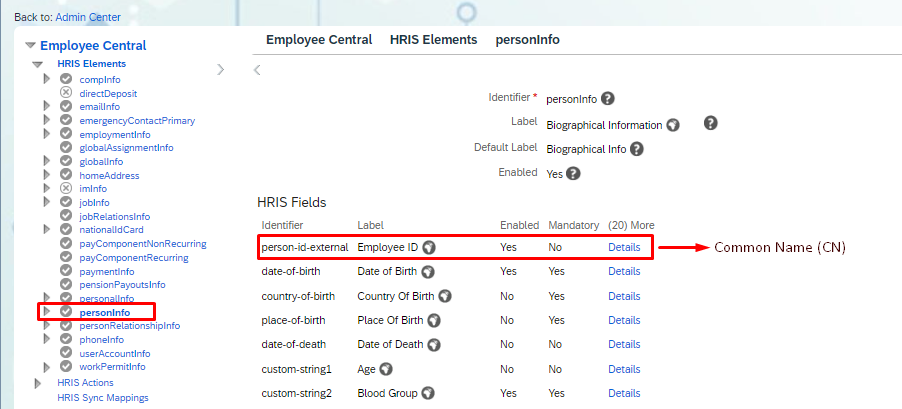
Description automatically generated

### **Generate the Certificate**

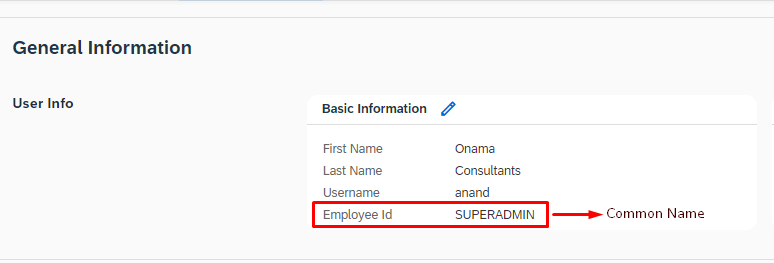
**X.509 Certificate:** To generate the new X.509 Certificate, click the Generate X.509 Certificate. Enter the relevant information and click Generate.

**Common Name (CN):** Enter the User\_id of the SuccessFactors Instance.

* The Common Name must be the value that is maintained to identifier person-id-external in SAP SuccessFactors.
* To check this field login into SAP SuccessFactors instance and navigate to **Manage Business Configuration**
* Go to the personInfo HRIS element and check for the field person-id external as shown in Screenshot below:



* To check the value of this field, navigate to people profile and under General Information > User Info, this field will be available.

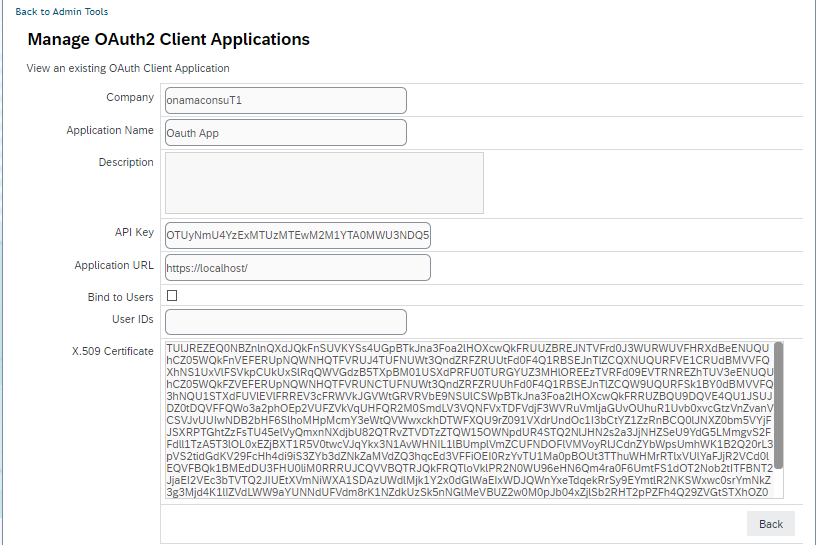


* Add this Employee Id in the Common Name field while generating the certificate.

A screenshot of a computer

Description automatically generated

* Enter certificate content such as Organization, Organization Unit, Locality, State/Province, Country, and Validity.
* After you click on Generate, the X.509 Certificate text field is automatically filled by new generated Certificate.
* Click Register
* The system generates an API Key in the API Key field. Note down the API key generated, that will be used in further steps.



**Note**: SuccessFactors does not store the copy of the private key , therefore in order to retrieve the X.509 private key , you must download the full X.509 certificate PEM file . The PEM file includes both the private key and the public key.

# **Establish a connection in Postman.**

**Procedure**

To obtain the Access token you need to follow two steps and then the access token is generated.

## Obtain the Assertion token.

* Open Postman or download it from the official site.

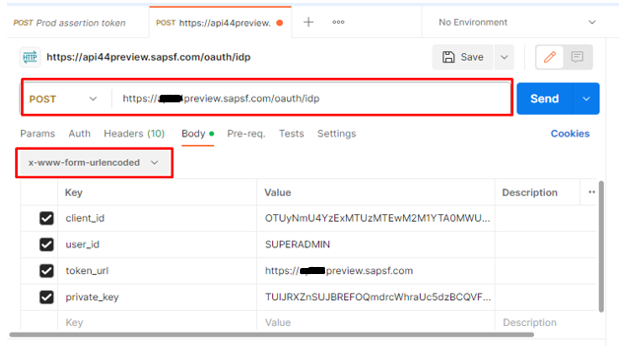
<https://www.postman.com/>

* Create a new request by clicking on the “+” button on the top-left corner.
* Select the request method as “POST”
* Set the request URL as:

https://api44preview.sapsf.com/oauth/idp

* The request URL will defer according to the data centres.
* The list of URL for different data centres can be found in the below KBA .

<https://userapps.support.sap.com/sap/support/knowledge/en/3260445>

* In the request body choose the **x-www-form-urlencoded** and provide the below parameters, shown in the below screenshot.
* **client\_id**- client id is the API key generated in the SAP SuccessFactors under Manage OAuth2 Client Application. A screenshot of a computer

  Description automatically generated
* user\_id- Add User Id as that was maintained in the Common Name field under the Manage OAuth2 Client Application.

A screenshot of a computer

Description automatically generated

* **token\_url** : token URL will again will defer according to the data centres.

The list of URL for different data centres can be found in the below KBA .

<https://userapps.support.sap.com/sap/support/knowledge/en/3260445>

* **private\_key** : The private key will be from the downloaded X.509 certificate PEM file.

Open the file in text editor and copy the content between

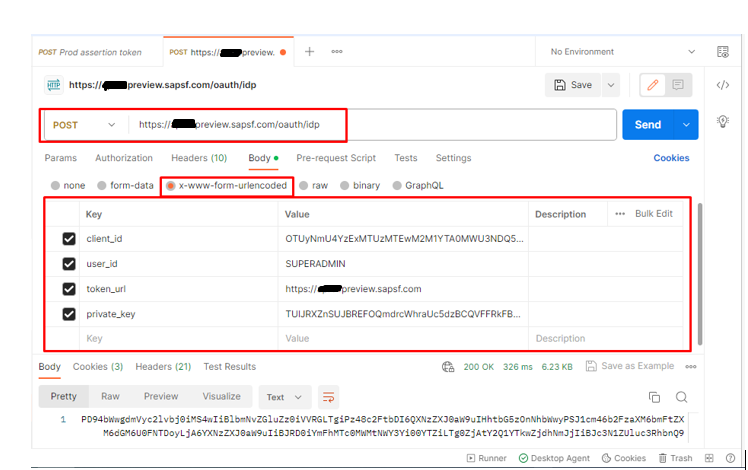
-------BEGIN ENCRYPTED PRIVATE KEY---------and------END ENCRYPTED PRIVATE KEY .

This is the key that needs to be added in private\_key.

A screenshot of a computer

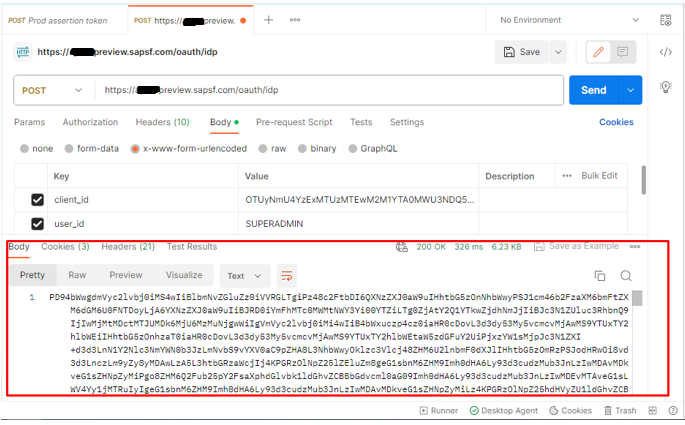
Description automatically generated

* Check the attached screenshot of the all the details added in the postman.



* After adding all the details click on Send and then the Assertion token will be generated. Note the Assertion token as that will be used in the further step to obtain the Access Token.

Check the attached screenshot of the assertion token generated.



## Obtain the OAuth Token

* Create a new request by clicking on the “+” button on the top-left corner.
* Select the request method as “POST”
* Set the request URL as

https://<SF\_API\_URL>.sapsf.com/oauth/token

The request URL will defer according to the data centres.

The list of URL for different data centres can be found in the below KBA .

<https://userapps.support.sap.com/sap/support/knowledge/en/3260445>

* In the request body choose the **x-www-form-urlencoded** and provide the below parameters, shown in the below screenshot.
* **client\_id:** client id is the API key generated in the SAP SuccessFactors under Manage OAuth2 Client Application

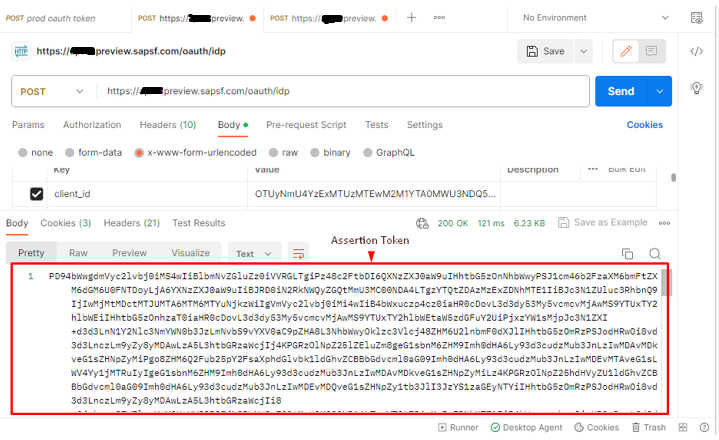
A screenshot of a computer

Description automatically generated

* **grant\_type**: Add the value for grant type as

**“urn:ietf:params:oauth:grant-type:saml2-bearer”**

* **company\_id**: Put company id as the SuccessFactors instance Id that is used to identify the organization before logging in to SuccessFactors instance.
* **assertion**: In the Assertion value add the token generated in the previous step. Check the attached screenshot for the reference.



Click the Send button in Postman to execute the request.

* Receive the Access Token – If the request is successful, you will receive a response containing the access token and the related details.

The response will typically include an ‘access\_token’, ‘token\_type’, and the ‘expires\_in’.

A screenshot of a computer

Description automatically generated

* Take the note of the ‘access\_token’ value as that will be needed in the API Request

## Use the Access Token

* Copy the access\_toke value from the response.
* To use the access\_token in Postman for subsequent requests:
* Add the Auhorization header to your API request.
* Set the header value to Bearer and add token generated in the previous step.

Check the attached screenshot for the same:

A screenshot of a computer

Description automatically generated

C Sharp Code

using System;

using System.Collections.Generic;

using System.Net.Http.Headers;

using System.Net.Http;

using System.Windows.Forms;

using System.Net;

using Newtonsoft.Json;

//using static System.Net.WebRequestMethods;

using System.IO;

using System.IO.Compression;

namespace SF\_Auth

{

public partial class Form1 : Form

{

public string oAuthUserId = "YOUR SF PERSON ID";

public string oAuthCompanyId = "YOUR SF COMPANY ID";

public string oAuthClientId = "ODhkMjY2Njg2ZWI4ZjUyNmRmNmVlNzRmYjEyNA";

public string oAuthGrantType = "urn:ietf:params:oauth:grant-type:saml2-bearer";

public string oAuthAssertionToken = string.Empty;

public TokenRoot oAuthToken { get; set; }

public string oAuthAPIUrl = https://YOUR\_SF\_API\_URL/odata/v2/;

public string oAuthAssertionURL = https://YOUR\_SF\_API\_URL/oauth/idp;

public string oAuthTokenURL = https://YOUR\_SF\_API\_URL/oauth/token;

public string oAuthClientSecret = "";

public Form1()

{

InitializeComponent();

}

public bool GetAssertionToken()

{

bool breturn = false;

try

{

oAuthAssertionToken = null;

using (var client = new HttpClient())

{

var postData = new List<KeyValuePair<string, string>>();

postData.Add(new KeyValuePair<string, string>("client\_id", oAuthClientId));

postData.Add(new KeyValuePair<string, string>("user\_id", oAuthUserId));

postData.Add(new KeyValuePair<string, string>("token\_url", oAuthTokenURL));

postData.Add(new KeyValuePair<string, string>("private\_key", oAuthClientSecret));

HttpContent content = new FormUrlEncodedContent(postData);

content.Headers.ContentType = new MediaTypeHeaderValue("application/x-www-form-urlencoded");

var responseResult = client.PostAsync(oAuthAssertionURL, content).Result;

oAuthAssertionToken = responseResult.Content.ReadAsStringAsync().Result;

responseResult.Dispose();

breturn = true;

}

}

catch

{

oAuthAssertionToken = null;

breturn = false;

}

return breturn;

}

public bool GetOAuthToken()

{

bool breturn = false;

try

{

GetAssertionToken();

using (var client = new HttpClient())

{

if (string.IsNullOrEmpty(oAuthAssertionToken) == false)

{

var postData = new List<KeyValuePair<string, string>>();

postData.Add(new KeyValuePair<string, string>("client\_id", oAuthClientId));

postData.Add(new KeyValuePair<string, string>("grant\_type", oAuthGrantType));

postData.Add(new KeyValuePair<string, string>("company\_id", oAuthCompanyId));

//postData.Add(new KeyValuePair<string, string>("assertion", assertiondata));

postData.Add(new KeyValuePair<string, string>("assertion", oAuthAssertionToken));

//postData.Add(new KeyValuePair<string, string>("client\_secret", oAuthClientSecret));

HttpContent content = new FormUrlEncodedContent(postData);

content.Headers.ContentType = new MediaTypeHeaderValue("application/x-www-form-urlencoded");

var responseResult = client.PostAsync(oAuthTokenURL, content).Result;

if (responseResult.StatusCode == HttpStatusCode.OK)

{

oAuthToken = JsonConvert.DeserializeObject<TokenRoot>(responseResult.Content.ReadAsStringAsync().Result);

breturn = true;

responseResult.Dispose();

}

else

{

string exMessage = "ErrorStatusCode: " + responseResult.StatusCode + " | " + "Error Message: " + responseResult.Content.ReadAsStringAsync();

responseResult.Dispose();

throw new Exception(exMessage);

}

}

}

}

catch (Exception ex)

{

oAuthToken = null;

breturn = false;

throw ex;

}

return breturn;

}

public string ExecuteODataAPIParam(string parameters)

{

string responseData = string.Empty;

try

{

//try to get token again

if (oAuthToken == null)

{

GetOAuthToken();

}

//if (/\* token refreshed, repeat the request using the new access token \*/)

using (var client = new HttpClient())

{

//TokenRoot deserializedToken = JsonConvert.DeserializeObject<TokenRoot>(oAuthToken);

client.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("Bearer", oAuthToken.access\_token);

//HttpResponseMessage result = client.GetAsync(EndPoint + parameters).Result;

//for queries returning large data set we need to provide timeouts

client.Timeout = TimeSpan.FromMinutes(30);

HttpResponseMessage result = client.GetAsync(oAuthAPIUrl + parameters, HttpCompletionOption.ResponseContentRead).Result;

if (result.StatusCode == HttpStatusCode.OK)

{

responseData = result.Content.ReadAsStringAsync().Result;

}

else //if (result.StatusCode == HttpStatusCode.Unauthorized)

{

// Process the error

responseData = "ErrorStatusCode: " + result.StatusCode + " | " + "Error Message: " + result.Content.ReadAsStringAsync();

}

result.Dispose();

if (string.IsNullOrEmpty(responseData) == false)

{

//we received data or error

if ((responseData.StartsWith("Exception:") == true) || (responseData.StartsWith("ErrorStatusCode:") == true))

{

throw new Exception(responseData);

}

}

}

}

catch (Exception ex)

{

throw ex;

}

return responseData;

}

private void btnGetEmpJob\_Click(object sender, EventArgs e)

{

tbOutput.Text = ExecuteODataAPIParam("EmpJob?$format=json");

}

private void button1\_Click(object sender, EventArgs e)

{

string perPersonalUrl = "PerPersonal?$filter=personIdExternal eq '{0}' &$select=personIdExternal,firstName,lastName,middleName &$format=json";

tbOutput.Text = ExecuteODataAPIParam(string.Format(perPersonalUrl, tbUserId.Text));

}

}

public class TokenRoot

{

public string access\_token { get; set; }

public string token\_type { get; set; }

public int expires\_in { get; set; }

}

}