



**SAP SuccessFactors** 

**PUBLIC**

Document Version: 2H 2021 – 2022-05-07

# **SAP SuccessFactors HXM Suite OData API: Developer Guide (V2)**

# Content

<b>1</b>	<b>About HXM Suite OData APIs (V2).</b>	<b>5</b>
1.1	List of SAP SuccessFactors API Servers.	6
<b>2</b>	<b>Change History.</b>	<b>9</b>
<b>3</b>	<b>Authentication.</b>	<b>11</b>
3.1	Restricting Access.	11
	Restricting OData API Access through Basic Authentication.	12
	Setting the Password Policy for API Access.	13
3.2	HTTP Basic Authentication (Deprecated).	14
	Permissions for Using HTTP Basic Authentication.	14
	Authenticating from a Browser.	15
3.3	Authentication Using OAuth 2.0.	15
	Registering Your OAuth2 Client Application.	17
	Generating a SAML Assertion.	23
	Requesting an Access Token.	28
	Viewing the Validity of an Access Token.	29
3.4	Enabling Session Reuse.	31
	Validating a Reused Session.	34
<b>4</b>	<b>Permissions.</b>	<b>35</b>
<b>5</b>	<b>Operations.</b>	<b>37</b>
5.1	Composing the OData URI.	37
5.2	Handling Special Characters.	38
5.3	Date and Time.	41
	Using the DateTime Format.	41
	Using the DateTimeOffset Format.	42
5.4	Query Operations.	43
	Query with Key Predicate.	44
	System Query Options.	45
	Pagination.	67
	Querying Effective-Dated Entities.	76
5.5	Edit Operations.	77
	Insert.	78
	Merge.	79
	Replace.	80
	Upsert.	81

	Delete. . . . .	90
	Edit Parameters. . . . .	91
5.6	Working with Links. . . . .	98
	Querying Links. . . . .	99
	Creating Links. . . . .	100
	Updating Links. . . . .	101
	Deleting Links. . . . .	103
	Working with Links in Effective-Dated Entities. . . . .	104
5.7	Batch Operations. . . . .	108
	Upsert Behavior in a \$batch ChangeSet. . . . .	111
	Examples of \$batch Request Bodies. . . . .	113
<b>6</b>	<b>MDF OData API. . . . .</b>	<b>116</b>
6.1	Configuring API Visibility for MDF Generic Objects. . . . .	116
	API Subversioning. . . . .	117
6.2	MDF Foundation Object Entities. . . . .	119
6.3	Data Mapping Between MDF and OData API. . . . .	120
6.4	Locale Handling in MDF Entities. . . . .	130
6.5	MDF OData API Operations. . . . .	135
	Permissions. . . . .	136
	Query Operation. . . . .	136
	Edit Operations. . . . .	160
<b>7</b>	<b>API Center. . . . .</b>	<b>182</b>
7.1	Permissions for API Center. . . . .	182
7.2	Data Dictionary. . . . .	183
	OData API Data Dictionary. . . . .	183
	Legacy SFAPI Data Dictionary. . . . .	184
7.3	Enabling API Audit Logs. . . . .	185
7.4	OData API Metadata Management. . . . .	188
7.5	Creating and Editing Users Using an API Option Profile. . . . .	189
7.6	SFAPI Metering Details. . . . .	193
<b>8</b>	<b>Metadata. . . . .</b>	<b>195</b>
8.1	Retrieving Metadata. . . . .	196
8.2	OData Annotations for EntitySet. . . . .	198
8.3	OData Annotations for Properties. . . . .	199
8.4	OData Annotations for Navigation Properties. . . . .	200
8.5	OData Service Catalog. . . . .	201
<b>9</b>	<b>Errors. . . . .</b>	<b>202</b>
9.1	Common OData Errors. . . . .	202
9.2	Logon Errors. . . . .	206

9.3 OAuth 2.0 Errors. . . . . 208

10 OData API Best Practices. . . . . 213

# 1 About HXM Suite OData APIs (V2)

The Open Data Protocol (OData) is a standardized protocol for creating and consuming data APIs. OData builds on core protocols like HTTP, and commonly accepted methodologies like REST. The result is a uniform way to expose full-featured data APIs. OData provides both a standard for how to represent your data and a metadata method to describe the structure of your data, and the operations available in your API. This document focuses on OData version 2.0.

The SAP SuccessFactors HXM Suite OData API is a Web Service API feature based on the OData protocol. It's intended to enable access to SAP SuccessFactors data in the system. SAP SuccessFactors HXM Suite OData API provides methods for CRUD (Create, Read, Update and Delete) operations.

We offer the following OData API user guides on the Help Portal:

- The SAP SuccessFactors HXM Suite OData API: Developer Guide (V2) provides framework level information for developers, such as authorization setup, OData operations, available OData tools in the API center, etc.
- The SAP SuccessFactors HXM Suite OData API: Reference Guide (V2) provides a complete list of the latest API references with entity-specific details and use cases.
- The SAP SuccessFactors Employee Central OData API: Reference Guide provides a complete list of the latest API references in Employee Central with entity-specific details and use cases.

## Enabling OData API

The OData API feature is enabled for all instances by default, unless you request to manually turn it off.

## Related Information

[SAP SuccessFactors API Packages on SAP API Business Hub](#) 

## 1.1 List of SAP SuccessFactors API Servers

Learn about the API servers of your company instance and how to construct the endpoint URLs.

### Endpoint URL Patterns

#### ! Restriction

We don't support IP addresses in URLs as part of our reference architecture. Use domain names instead. If you think you have a special case that requires IP addresses instead of domain names, contact Product Support.

Protocol	URL Pattern	Endpoint Example
OData v2	/odata/v2/	<a href="https://api17.sapsf.com/odata/v2/">https://api17.sapsf.com/odata/v2/</a>
OData v4	/odatav4/	<a href="https://api17.sapsf.com/odatav4/">https://api17.sapsf.com/odatav4/</a>
SFAPI	/sfapi/v1/soap	<a href="https://api17.sapsf.com/sfapi/v1/soap">https://api17.sapsf.com/sfapi/v1/soap</a>
WSDL	/sfapi/v1/soap?wsdl	<a href="https://api17.sapsf.com/sfapi/v1/soap?wsdl">https://api17.sapsf.com/sfapi/v1/soap?wsdl</a>

### API Servers

Here's a list of API servers for SAP SuccessFactors data centers. Use search and filter to find the API server for your company. A data center can have synonyms from its legacy and next generation data center names, indicated by a numeric value. For example, DC17 and DC60 are synonyms of the Toronto data center. The corresponding URLs also point to the same data center. You can use either URL to access APIs.

To view the timezone information of an API server, go to your company login page or open your account on the header bar after login, and choose [Show version information](#).

#### i Note

SAP SuccessFactors revised its data center numbering as part of its Next Generation Cloud Delivery Platform. While it is leading practice to update a domain name or URL to the latest domain name, the domain names that you migrated **from** continue to work. For example, if the **Next Generation Cloud Delivery Platform** migrated you from a URL with DC17 in the domain name to a URL with DC60 in the domain name, the DC17 URL, <https://api17.sapsf.com>, continues to point to the same Toronto data center.

Legacy Numeric Name	Next Generation Numeric Name (Planned)	Environment	Location	API Server
DC2	DC56	Production	Amsterdam, The Netherlands	<a href="https://api2.successfactors.eu/">https://api2.successfactors.eu/</a>
DC2	DC56	SalesDemo	Amsterdam, The Netherlands	<a href="https://apisalesdemo2.successfactors.eu/">https://apisalesdemo2.successfactors.eu/</a>
DC2	DC56	Preview	Amsterdam, The Netherlands	<a href="https://api2preview.sapsf.eu/">https://api2preview.sapsf.eu/</a>
DC4	DC48	Production	Chandler, Arizona, US	<a href="https://api4.successfactors.com/">https://api4.successfactors.com/</a>
DC4	DC48	SalesDemo	Chandler, Arizona, US	<a href="https://apisalesdemo4.successfactors.com/">https://apisalesdemo4.successfactors.com/</a>
DC4	DC48	Preview	Chandler, Arizona, US	<a href="https://api4preview.sapsf.com/">https://api4preview.sapsf.com/</a>
DC8	DC48	Production	Ashburn, Virginia, US	<a href="https://api8.successfactors.com/">https://api8.successfactors.com/</a>
DC8	DC48	SalesDemo	Ashburn, Virginia, US	<a href="https://apisalesdemo8.successfactors.com/">https://apisalesdemo8.successfactors.com/</a>
DC8	DC48	Preview	Ashburn, Virginia, US	<a href="https://api8preview.sapsf.com/">https://api8preview.sapsf.com/</a>
DC10	DC66	Production	Sydney, Australia	<a href="https://api10.successfactors.com/">https://api10.successfactors.com/</a>
DC10	DC66	Preview	Sydney, Australia	<a href="https://api10preview.sapsf.com/">https://api10preview.sapsf.com/</a>
DC12	DC26	Production	Rot, Germany	<a href="https://api012.successfactors.eu/">https://api012.successfactors.eu/</a>
DC12	DC26	Preview	Rot, Germany	<a href="https://api12preview.sapsf.eu/">https://api12preview.sapsf.eu/</a>
DC15	DC64	Production	Shanghai, China	<a href="https://api15.sapsf.cn/">https://api15.sapsf.cn/</a>
DC17	DC60	Preview	Toronto, Canada	<a href="https://api17preview.sapsf.com/">https://api17preview.sapsf.com/</a>
DC17	DC60	Production	Toronto, Canada	<a href="https://api17.sapsf.com/">https://api17.sapsf.com/</a>
DC18	DC28	Preview	Moscow, Russia	<a href="https://api18preview.sapsf.com/">https://api18preview.sapsf.com/</a>
DC18	DC28	Production	Moscow, Russia	<a href="https://api18.sapsf.com/">https://api18.sapsf.com/</a>
DC19	DC62	Preview	Sao Paulo, Brazil	<a href="https://api19preview.sapsf.com/">https://api19preview.sapsf.com/</a>
DC19	DC62	Production	Sao Paulo, Brazil	<a href="https://api19.sapsf.com/">https://api19.sapsf.com/</a>
DC22	DC22	Preview	Dubai, UAE	<a href="https://api22preview.sapsf.com/">https://api22preview.sapsf.com/</a>
DC22	DC22	Production	Dubai, UAE	<a href="https://api22.sapsf.com/">https://api22.sapsf.com/</a>

Legacy Numeric Name	Next Generation Numeric Name (Planned)	Environment	Location	API Server
DC23	DC23	Preview	Riyadh, Saudi Arabia	<a href="https://api23preview.sapsf.com/">https://api23preview.sapsf.com/</a>
DC23	DC23	Production	Riyadh, Saudi Arabia	<a href="https://api23.sapsf.com/">https://api23.sapsf.com/</a>
DC41	DC41	Preview	US East (Microsoft Azure)	<a href="https://api41preview.sapsf.com">https://api41preview.sapsf.com</a>
DC41	DC41	Production	US East (Microsoft Azure)	<a href="https://api41.sapsf.com">https://api41.sapsf.com</a>
DC44	DC52	Preview	Singapore	<a href="https://api44preview.sapsf.com/">https://api44preview.sapsf.com/</a>
DC44	DC52	Production	Singapore	<a href="https://api44.sapsf.com/">https://api44.sapsf.com/</a>
DC47	DC47	Preview	Canada Central (Microsoft Azure)	<a href="https://api47preview.sapsf.com/">https://api47preview.sapsf.com/</a>
DC47	DC47	Production	Canada Central (Microsoft Azure)	<a href="https://api47.sapsf.com/">https://api47.sapsf.com/</a>
DC55	DC55	Preview	Europe West 3	<a href="https://api55preview.sapsf.eu/">https://api55preview.sapsf.eu/</a>
DC55	DC55	Production	Europe West 3	<a href="https://api55.sapsf.eu/">https://api55.sapsf.eu/</a>



## 2 Change History

Learn about changes to the documentation for SAP SuccessFactors HXM Suite OData API: Developer Guide in recent releases.

### 2H 2021

Type of Change	Description	More Info
December 17, 2021		
Changed	We improved the descriptions of the rules of subversioning. There is no behavior change.	<a href="#">API Subversioning [page 117]</a>
Changed	We improved the description for API permissions. There is no behavior change.	<a href="#">Permissions [page 136]</a>
October 8, 2021		
Changed	When you register an OAuth client, you can now bind the client to multiple users, including both technical users and business users.	<a href="#">Registering Your OAuth2 Client Application [page 17]</a>
Added	We added information about the planned data center names available after the Next Generation Cloud Delivery Platform migration.	<a href="#">List of SAP SuccessFactors API Servers [page 6]</a>

### 1H 2021

Type of Change	Description	More Info
August 20, 2021		
Changed	We corrected the MDF foundation objects for FOCCompany, FOJobCode, FOJobClassLocal<Country/Region>, and FOLegalEntityLocal<Country/Region>.	<a href="#">MDF Foundation Object Entities [page 119]</a>
April 9, 2021		
Change	We updated the topics where instance-level and API-specific IP restrictions and exceptions apply.	<a href="#">HTTP Basic Authentication (Deprecated) [page 14]</a> <a href="#">Requesting an Access Token [page 28]</a>

Type of Change	Description	More Info
Change	We updated the supported signing algorithms and our recommendation for generating SAML assertions using a third-party IdP. We also added the required elements in the SAML assertions.	<a href="#">Generating a SAML Assertion [page 23]</a>

## 2H 2020

What's New	Description	More Information
Planned retirement of HTTP Basic Authentication	We plan to retire HTTP Basic Authentication in favor of more secure authentication methods such as OAuth.	<a href="#">HTTP Basic Authentication (Deprecated) [page 14]</a>
Return all upsert errors in a \$batch ChangeSet	You can now use the <code>enableUpsertResponseExtensionInChangeset</code> parameter to return all upsert errors in a \$batch ChangeSet.	<a href="#">Upsert Behavior in a \$batch ChangeSet [page 111]</a> <a href="#">enableUpsertResponseExtensionInChangeset [page 96]</a>

# 3 Authentication

This document explains the types of authentication used to access OData API, how to enable session reuse for OData API access, and how to set exceptions for API login.

SAP SuccessFactors provides the following two types of authentication for OData API:

- HTTP Basic Authentication (deprecated): Requires username, company ID, and password to log in.
- OAuth 2.0: A more secure way to authenticate users without providing passwords.

## ⚠ Caution

HTTP Basic Authentication will soon be retired. Please migrate to OAuth as soon as possible. For detailed retirement plan, see the announcement in the SAP SuccessFactors Migrating Features guide.

## 3.1 Restricting Access

Use the Admin Center tools to control who can access APIs in your instance.

### Context

API access restrictions can be set in several different tools in the Admin Center. Depending on the level of restrictions you want to set and which APIs you want to restrict access, choose the right tools.

If you want to ...	Use this tool
Restrict all API access through Basic Authentication	<a href="#">OData API Basic Authentication Configuration</a>
Restrict all API access on instance level	<a href="#">IP Restriction Management</a>
Restrict all API access on user level	<a href="#">Password &amp; Login Policy Settings</a> > <a href="#">Set API login exceptions...</a>

## i Note

If both instance-level and user-level IP restrictions are set, a user can access APIs if either condition is met.

### Procedure

- To restrict API access through Basic Authentication, follow the instructions in [Restricting OData API Access through Basic Authentication \[page 12\]](#).

- To restrict API access on instance level, follow the instructions in [IP Restriction Management](#).
- To restrict API access for specific users, follow the instructions in [Setting the Password Policy for API Access \[page 13\]](#).

### 3.1.1 Restricting OData API Access through Basic Authentication

When HTTP Basic Authentication (Basic Auth) is used to access OData API, you can control which IP addresses are allowed the access using the OData IP allowlisting tool. Please note that this feature is called [OData API Basic Authentication Configuration](#) in the Admin Center and [OData IP Allowlisting](#) in the API Center.

You have the following options to limit the access of OData API using Basic Auth:

- *Always*: Users are allowed to log in using Basic Auth from any IP address.
- *Never*: No one is allowed to log in using Basic Auth.
- *Restrict access to below IPs*: Only users from specified IP addresses are allowed to log in using Basic Auth. With this option enabled, you can enter a list of IP addresses separated by comma. For example: **10.20.30.40**, **10.20.30.41**, **10.20.\*.\***.

#### i Note

When you use wildcards, you must enter all four bits of the IP address. For example, **10.10.\*.\*** is a valid entry and **10.10.\*** isn't.

#### i Note

The [OData IP Allowlisting](#) tool can only control OData API accesses using HTTP Basic Auth. To control all API accesses, use the [Password & Login Policy Settings](#) tool in Admin Center. For more information, see [Setting Password Policy and Exceptions for API Login](#).

## Related Information

[HTTP Basic Authentication \(Deprecated\) \[page 14\]](#)

## 3.1.2 Setting the Password Policy for API Access

Use the [Password & Login Policy Settings](#) page to set the maximum password age and login exceptions for OData API and SFAPI users.

### Context

Your SAP SuccessFactors HXM Suite administrator sets password policies for all users in the system, including the timing for password expirations. However, you may want to set different expiration times for passwords for integrations that are built against a specially designed API user account. You can set exceptions to the password policy for your API users, and specify a different password expiration (maximum password age) policy. To maintain security, users who have password policy exceptions are required to have an IP address restriction to connect to the API server.

Follow the instructions below to set password policy and login exceptions for your API users.

### Procedure

1. Go to ► [Admin Center](#) ► [Password & Login Policy Settings](#) ►, and choose [Set API login exceptions....](#) You can also access the [Password & Login Policy Settings](#) tool from ► [API Center](#) ► [Legacy SFAPI IP Whitelisting](#) ►.
2. Click [Add](#) to create a new policy. In the pop-up dialog, enter the following:

Field	Description
Username	Enter the username for whom you want to apply the policy.
Maximum password age (days)	Enter the number of days the password is valid for. To set a password to never expire, enter <b>-1</b> . <div><b>i Note</b> For security reasons, we do not recommend setting a password to never expire.</div>
IP address restrictions	Enter the IP addresses separated by comma. For example: <b>10.20.30.40, 10.20.30.1-10.20.30.10</b> . <div><b>i Note</b> The maximum password age applies only when the user is accessing from the IP addresses specified here. Only IPv4 addresses are supported.</div>

3. Save and close.

## 3.2 HTTP Basic Authentication (Deprecated)

HTTP Basic Authentication (Basic Auth) allows you to access OData API by providing username, company ID, and password information on an HTTP user agent, such as a web browser.

### ⚠ Caution

HTTP Basic Authentication will soon be retired. Please migrate to OAuth as soon as possible. For detailed retirement plan, see the announcement in the SAP SuccessFactors Migrating Features guide.



To log in with Basic Auth, you must have the [Allow Admin to Access OData API through Basic Authentication](#) permission and your client IP address is on the allow list to access the API server. For more information, see the Restricting API Access by IP section.

The authorization header of Basic Auth is constructed in the following way:

- Username, company ID, and password are combined into a string as such: `username@company ID:password`
- The resulting string literal is then encoded using Base64.
- The authentication method ("Basic") followed by a space is then put before the encoded string. For example, `Authorization: Basic QWxhZGRpbjpvcGVuIHNlc2FtZQ==`

## Restricting API Access by IP

HTTP Basic Authentication is generally considered less secure than authentication using OAuth 2.0. However, you can increase the security by controlling the API access based on clients' IP address in the following settings:


- [Admin Center](#) > [OData API Basic Authentication Configuration](#) : This setting affects only the access to OData APIs using Basic Auth.
- [Admin Center](#) > [Password & Login Policy Settings](#) > [Set IP Login Exceptions](#) : This setting affects all OData API and SFAPI calls.

## Related Information

[Restricting OData API Access through Basic Authentication \[page 12\]](#)

### 3.2.1 Permissions for Using HTTP Basic Authentication

Grant the required permissions to your role so that you can access OData APIs using HTTP Basic authentication.

Assign the [Allow Admin to Access OData API through Basic Authentication](#) permission under [Admin Center](#) > [Permissions](#) > [Manage Integration Tools](#)  to the relevant role so that you can access OData APIs using Basic Auth.

## 3.2.2 Authenticating from a Browser

You can use a web browser based authentication to enable querying..

To use this feature, you need:

1. Input URL of query OData in address bar. e.g. `https://<hostname>/odata/v2/Attachment?$top=1`
2. Enter your username and password when prompted.

You will not need to re-enter this information for subsequent queries until the browser is closed. If the user name and password verification fails, you will need to open a new window and try again

### i Note

When you query in atom format in Firefox, the xml result will not be displayed, because Firefox regards atom format as RSS content by default. You can check the result in JSON format or view the source code to check the query result.

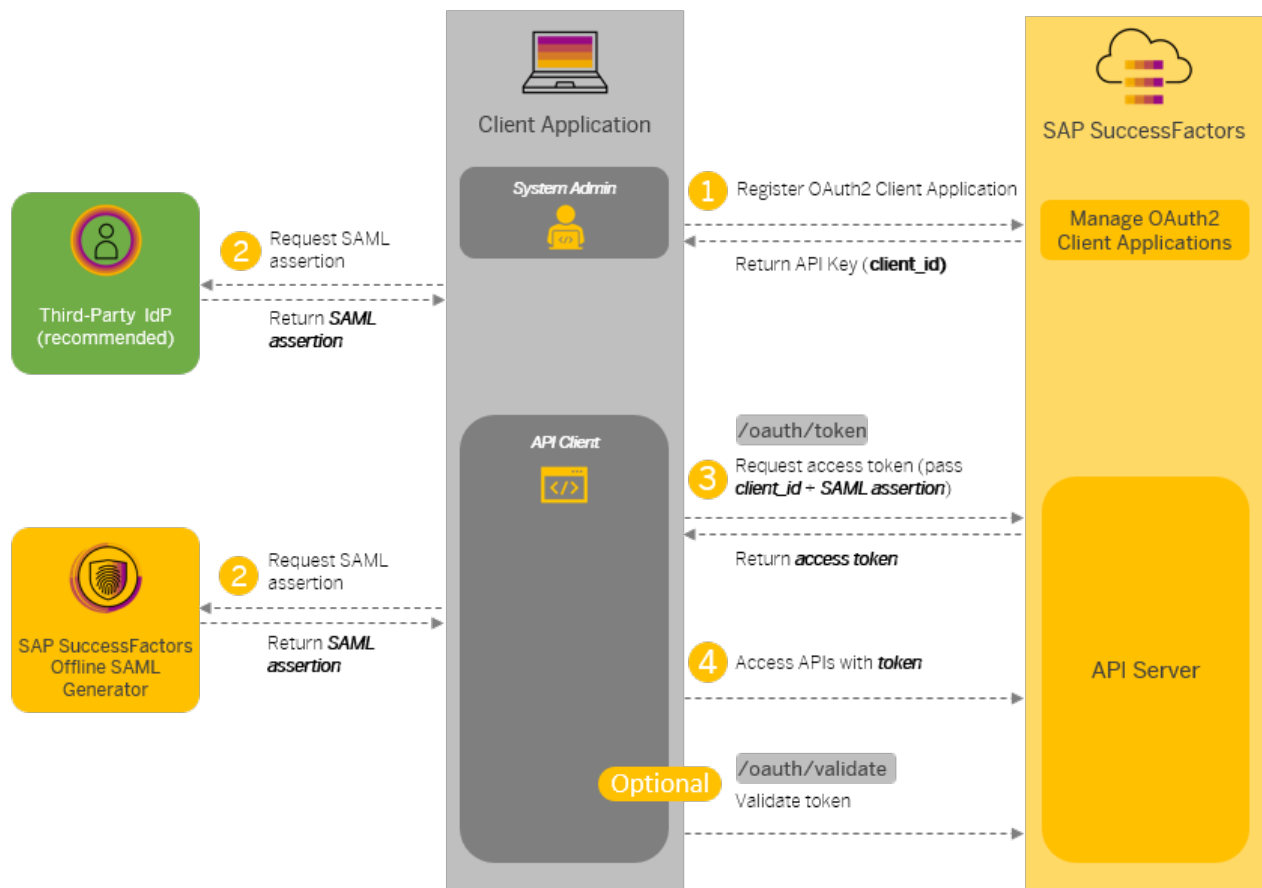
## 3.3 Authentication Using OAuth 2.0

Learn how to set up and use OAuth 2.0 for authenticating API users.

SAP SuccessFactors supports OAuth 2.0 to authenticate OData API and SFAPI users. Compared with HTTP Basic Auth, OAuth 2.0 is considered to be more secure in that it doesn't require users to provide their passwords during authentication. With OAuth 2.0, you can also use a third-party identity provider (IDP) for user management and provisioning.

### Process Overview

The following diagram explains how OAuth 2.0 works with SAP SuccessFactors.



- [Registering Your OAuth2 Client Application \[page 17\]](#)
- [Generating a SAML Assertion \[page 23\]](#)
- [Requesting an Access Token \[page 28\]](#)
- [Viewing the Validity of an Access Token \[page 29\]](#)
- [Follow the documentation of the IdP for requesting SAML assertions. \[page 15\]](#)

1. Register your client application in SAP SuccessFactors to obtain an API key.
2. Obtain a SAML assertion either from your trusted IdP (recommended) or using the offline SAML generator provided by SAP SuccessFactors.
3. Pass your SAML assertion and API key (in the `client_id` field) along with other information to generate an OAuth token.
4. Use the generated token to call APIs.
5. (Optional) Check whether your access token has expired or not.

#### [Registering Your OAuth2 Client Application \[page 17\]](#)

Register your client application so that you can authenticate API users using OAuth2. After you register an application, you'll get an exclusive API key for your application to access SAP SuccessFactors OData APIs.

#### [Generating a SAML Assertion \[page 23\]](#)

Generate a Security Assertion Markup Language (SAML) assertion for requesting an OAuth token. This topic explains how to generate a SAML assertion using the offline tool provided by SAP SuccessFactors.



[Requesting an Access Token \[page 28\]](#)

With a SAML assertion, you can now call API `/oauth/token` to request an access token for authentication with the API server.

[Viewing the Validity of an Access Token \[page 29\]](#)

Use API `/oauth/validate` to verify if an access token is valid.

## 3.3.1 Registering Your OAuth2 Client Application

Register your client application so that you can authenticate API users using OAuth2. After you register an application, you'll get an exclusive API key for your application to access SAP SuccessFactors OData APIs.

### Prerequisites

You have the [Manage Integration Tools](#) [Manage OAuth2 Client Applications](#) permission.

### Procedure

1. Log into your instance as an administrator.
2. Go to [Admin Center](#) [API Center](#) [OAuth Configuration for OData](#) and choose [Register Client Application](#). You can also access the tool by searching [Manage OAuth2 Client Applications](#) in Action Search.
3. On the new OAuth client registration screen, enter the following information:

Option	Description
Company	The name of your company. This value is prefilled based on the instance of the company currently logged in.
Application Name	(Required) A unique name of your OAuth client.
Description	(Optional) A description of your application.
Application URL	(Required) A unique URL of the page that the client wants to display to the end user. The page contains more information about the client application. This is needed for 3-legged OAuth, however it isn't currently supported.
Bind to Users	(Optional) You can enable this option to restrict the access of the application to specific users including business users and technical users.

#### i Note

A business user in this context is a user who has permissions to call SAP SuccessFactors APIs for integration purposes.

Option	Description
	<p>A technical user is a system-generated user created for integrating SAP SuccessFactors with other SAP products and solutions.</p> <p>Refer to <a href="#">About Technical User</a> for more information.</p>
User IDs	<p>(Required if you enabled the <a href="#">Bind to User</a> option) Enter the user IDs separated by comma.</p> <p>The binding of business users and technical users works as follows:</p> <ul style="list-style-type: none"> <li>◦ If you don't bind any user to the application, all business users can request OAuth tokens but technical users can't.</li> <li>◦ If you bind both business users and technical users to the application, only these users can request OAuth tokens.</li> <li>◦ If you bind only technical users to the application, these technical users and any business user can request OAuth tokens.</li> <li>◦ If you bind only business users to the application, only these users can request OAuth tokens.</li> </ul> <div> <p><b>Note</b></p> <p>Contact your system administrator or Product Support if you don't know the technical user ID of your instance.</p> </div>
X.509 Certificate	<p>(Required) The certificate corresponding to the private and public key used in the OAuth 2.0 authentication process. In this flow, SAP SuccessFactors require the public key and the client application has the private key. To register a client application, you must install the public key in SAP SuccessFactors. If you supply that certificate, you must use the RSA-SHA1, RSA-SHA2, or MD5 encryption type for authentication.</p> <p>You can obtain an X.509 certificate from a trusted service provider, or you can use a third-party tool to generate a self-signed certificate. If neither option is available, you can generate an X.509 certificate in SAP SuccessFactors. For more information, see the <a href="#">Related Information</a> section of this topic.</p>

Option	Description
	<p><b>Note</b></p> <p>For better security, we recommend that you use a self-signed certificate or one from your trusted service provider.</p> <p>In a <b>.pem</b> file, the X.509 certificate is a BASE64-encoded string enclosed between -----BEGIN CERTIFICATE----- and -----END CERTIFICATE-----. Enter only the enclosed string without the beginning and ending lines. Otherwise, an error occurs.</p> <p>When you change or regenerate an X.509 certificate for an application, the existing application client configurations are invalidated. This could lead to application failure until you update the configurations with the new certificate information.</p>

4. Choose [Register](#) to save your registration.

## Results

You've successfully registered your client application for OAuth2 authentication. An API key is generated and assigned to your application. You can view the API key by choosing [View](#) on the registered application list.

You can also edit, disable, and delete an OAuth2 client registration.

**Task overview:** [Authentication Using OAuth 2.0 \[page 15\]](#)

## Related Information

[Generating a SAML Assertion \[page 23\]](#)

[Requesting an Access Token \[page 28\]](#)

[Viewing the Validity of an Access Token \[page 29\]](#)


[Creating a Self-Signed X.509 Certificate \[page 20\]](#)

[Creating an X.509 Certificate in SAP SuccessFactors \[page 22\]](#)

### 3.3.1.1 Creating a Self-Signed X.509 Certificate

You can use tools such as OpenSSL to create a self-signed X.509 certificate.

#### Prerequisites

We recommend that you use OpenSSL to create the certificate. For Windows users, you can download the tool at <https://www.openssl.org> . For Mac and Linux users, OpenSSL is available with the native command-line tools such as Terminal.

#### Context

X.509 certificates are used in many Internet protocols, including TLS/SSL. An X.509 certificate consists of a public key and a private key. The public key contains the identity information, such as a hostname, an organization, or an individual. The public/private key pair is used to establish secure communication between your application and SAP SuccessFactors.

#### Procedure

1. Go to the OpenSSL library in your command-line tool.

For Mac and Linux users, you can call OpenSSL directly in the command tool under the default path. For Windows users, the entry point is the openssl binary, located in the installation folder, for example: c :

\Program Files\OpenSSL-Win64\bin\.

2. Follow the examples below to create an X.509 certificate:

- Example of creating a 2048-bit SHA256 key:

```
$ openssl req -nodes -x509 -sha256 -newkey rsa:2048 -keyout private.pem -out public.pem
```

- Example of creating a 1024-bit SHA256 key:

```
$ openssl req -nodes -x509 -sha256 -newkey rsa:1024 -keyout private.pem -out public.pem
```

- Example of creating a 1024-bit MD5 key:

```
$ openssl req -nodes -x509 -md5 -newkey rsa:1024 -keyout private.pem -out public.pem
```

#### Note

**private.pem** and **public.pem** are the example names of the public/private key pair generated with this command. You can change them to any names of your choice.

SAP SuccessFactors support certificates with SHA1, SHA256, and MD5 encryption types and key lengths from 512 bits to 2048 bits. However, for maximum security, we recommend that you use 2048-bit keys with SHA256 encryption.

3. Enter the following information when prompted:

Provide at least one of these values to create a certificate.

Option	Description
<b>Country Name</b>	Enter a two-letter country code of the entity to which the certificate is issued. A country code represents a country or a region. Example: <b>AU</b>
<b>State or Province Name</b>	Name of state or province of the entity to which the certificate is issued.
<b>Locality Name</b>	Name of locality of the entity to which the certificate is issued.
<b>Organization Name</b>	The entity to which the certificate is issued.
<b>Organization Unit Name</b>	The organization unit of the entity to which the certificate is issued.
<b>Common Name</b>	The hostname or IP address for which the certificate is valid. The common name (CN) represents the hostname of your application. It's technically represented by the commonName field in the X.509 certificate. The common name doesn't include any protocol, port number, or path. For example: <b>www.bestrun.com</b>
<b>E-mail Address</b>	Enter your e-mail address.

## Results

A public/private key pair is generated and saved to the local drive with the names you specified in the command.

### ⚠ Caution

Only the public key is required when you register an OAuth2 client application in SAP SuccessFactors. The private key must be kept secure under all circumstances. Do not share the private key with others. If you lose the private key, you must create a new certificate.

Example of a public key:

```
-----BEGIN CERTIFICATE-----
MIIB9jCCAV+gAwIBAgIUkR82LgtkNBccdyPD26K87zZ+vYwDQYJKoZIhvcNAQEE
BQAwDTELMAkGA1UECwwCRVAvHhcNMTkwOTI2MDIwNDUyWhcNMTkwMDI2MDIwNDUy
WjANMQswCQYDVQQLDAJFUdCBnzANBgkqhkiG9w0BAQEFAAOBjQAwYkCgYEAwKva
NZCOGcuY90/BudS+qQic+A3luM8mLtmI60R1iEjgEWGBCxSiDb2h8mQJiXwkul9W
ebaazP7hkqkdNoJgV/6NE7++GKyyS8fIhJgeWSb6EelMFhjQ0nZKzbZX5ms3I91n
twzkHtKCQi/gi/Rouhlk/P/QVcrzSgHUHQJNy0CAwEAAaNTMFEwHQYDVR0OBBYE
FHHbgqnnhm3GAJ4gy2IuEDxpLye7MB8GA1UdIwQYMBaAFHHbgqnnhm3GAJ4gy2Iu
EDxpLye7MA8GA1UdEwEB/wQFMAMBAf8wDQYJKoZIhvcNAQEEBQADgYEAG5CoqcEy
15vUpj5VfJeR/DS70tPIinp/TCC9kRO/++TSnPbqVcfPr8vIyc4L3MPKjXFBsefE
vtfHGGucVtv5N1+4U/b9NxNFbuH2MP7W3swZ4WM72Na+W6iOhwesOr0p3IcOfxc3
RNCnagFmtbDFxAlPXQ0d+m+N5gxLRoCX1hE=
-----END CERTIFICATE-----
```

Example of a private key:

```
-----BEGIN PRIVATE KEY-----
MIICdQIBADANBgkqhkiG9w0BAQEFAASCA18wgGJbAgEAAoGBAMCr2jWQjhnLmPdP
wbnUvqkInPgN9bjPJi7ZiOtEdYhI4BFhgQsUog29ofJkCY18JLpfVnm2msz+4ZKp
```

```
HTaCYFf+jRO/vhisskvHyISYHlkm+hHpTBYY0NJ2Ss22V+ZrNyPdZ7cM5HB7SgkI
v4Iv0aLoZZPz/0FXK80oB1B6iTctAgMBAAECgYAid5vVsUJ6gt2egHobkF97Rbsu
9PBcW1JtVyUTUW/1LYRIF7VKEirbYm0yO4spOTgozxldMLmIqqAX6ID9W114kN/g
lzl2c2/jMg+YGp+FNCjULygjfIwtGfpX8G0qYWza5oarZVbbGA1cvPHjyNMGV7ure
7syrjIXUi9hkaKrxgQJBAObVbGTVr/5xxScB1mPYoBe02JMyTzuVW0ts7NyfxXJu
w9vUoMDLV+2wuDE4w8/gUkKf26eojn3kwD708V6lG4kCQQDVrVC7HcXYfU4wkr5S
JPMQzAln0RUf6LgFpgIDPDKpq7VUti1A9aQUbdddxcudFjO57ksr2yU9sOLQgh3A
+2GFAkAWkRDavsVI48h5asWR11C3YJe3tDhow848DncNjpUX/dop+JyKnJaJBzjK
nxkNjomcN9KajnD3v9BH11ytewi5AkA8IAWscUc/kJrUziXhpWYD3vXykYG5Ndm6
NSkx0dmLprZifNSlB7nAyduqgXTe4eVyNxxN3d9PyZs5ArPuno2lAkAQ8WiHbqGA
JlO6R9+D6HiWywCaQ0oh6H/+84mb1ew2SUw1mFxROxgfsRVNUe+ahs3nSIhoba0
cqS0ZSBtNDxV
-----END PRIVATE KEY-----
```

### 3.3.1.2 Creating an X.509 Certificate in SAP SuccessFactors

You can create an X.509 certificate in SAP SuccessFactors HXM Suite if you're unable to create a self-signed certificate.

#### Context

##### ⚠ Caution

We don't recommend creating the X-509 certificate in API Center and downloading the private key. This method is less secure compared with a self-signed certificate because downloading the private key increases the risk of exposing it. This method should only be used if the client is unable to create a self-signed X.509 certificate.

#### Procedure

1. Log into your instance as an administrator.
2. Go to **Admin Center** > **API Center** > **OAuth Configuration for OData** and choose **Register Client Application**. You can also access the tool by searching **Manage OAuth2 Client Applications** in Action Search.
3. On the new OAuth client registration screen, choose **Generate X.509 Certificate** and enter the following information:

Option	Description
Issued By	Value set to <b>SuccessFactors</b>
Common Name	The hostname or IP address for which the certificate is valid. The common name (CN) represents the hostname of your application. It's technically represented by the common-Name field in the X.509 certificate. The common name doesn't include any protocol, port number, or path. For example: <b>www.bestrun.com</b>

Option	Description
<b>Organization</b>	(Optional) The entity to which the certificate is issued.
<b>Organization Unit</b>	(Optional) The organization unit of the entity to which the certificate is issued.
<b>Locality</b>	(Optional) Name of locality of the entity to which the certificate is issued.
<b>State/Province</b>	(Optional) Name of state or province of the entity to which the certificate is issued.
<b>Country</b>	(Optional) Enter a two-letter country code of the entity to which the certificate is issued. A country code represents a country or a region. Example: <b>AU</b>
<b>Validity</b>	The number of days for which you want the X.509 certificate to be valid.

4. Choose [Generate](#).

## Results

A new X.509 certificate is generated and filled in the [X.509 Certificate](#) field on the new OAuth2 client registration screen. Continue your registration in [Registering Your OAuth2 Client Application \[page 17\]](#) with this certificate.

### ⚠ Caution

Both the public key and private key are available to you in the generated certificate. You must save the private key before you register your client application. Only the public key is available for viewing when the client application is registered. The private key must be kept secure under all circumstances. Do not share the private key with others. If you lose the private key, you create a new one.

## 3.3.2 Generating a SAML Assertion

Generate a Security Assertion Markup Language (SAML) assertion for requesting an OAuth token. This topic explains how to generate a SAML assertion using the offline tool provided by SAP SuccessFactors.

## Prerequisites

You've registered your application in [Manage OAuth2 Client Applications](#) and obtained the API key for the application.

If you want to use the offline tool to generate a SAML assertion, you need to install Apache Maven in your local environment. Apache Maven is required to run the commands to generate SAML assertions in this task. For more information, see [Installing Apache Maven](#) 📖.

## Context

You have the following options to generate a SAML assertion:

- (Recommended) Use a third-party IdP that you trust. Refer to the documentation of that IdP for detailed instructions.

### Note

Both SHA-256 and SHA-1 signing algorithms are supported. However, we recommend that you use SHA-256 for better security.


- Use the offline tool.
- Use SAP SuccessFactors IdP. Note that the SAML assertion is valid only for 10 minutes.

### Caution

This method requires you to pass the private key through an API call and is therefore not recommended.

SAP SuccessFactors provides an offline tool to generate a SAML assertion for your registered application to access APIs. The SAML generator tool processes the input information offline and generates a SAML assertion without having to expose your private key to the Internet.

## Required Elements for Third-Party SAML Assertions

If you choose to use a third-party IdP to generate a SAML assertion, make sure that you follow the [SAML 2.0 standard](#)  and the following elements are included in the assertion:

### Tip

SAML assertions are Base64-encoded. To view the detailed information in XML format, decode the assertion using a Base64 decode tool.

Required Elements for SAML Assertions Generated by Third-Party IdP

Element	Description	Example
<code>&lt;saml2:Issuer&gt;</code>	Issuer information of the SAML assertion	<code>&lt;saml2:Issuer&gt;www.myidp.com&lt;/saml2:Issuer&gt;</code>



Element	Description	Example
<code>&lt;saml2:Subject&gt;</code> , <code>&lt;saml2:NameID&gt;</code> , and Recipient	Enter the SAP SuccessFactors user ID that you use to access the APIs in the NameID element. The recipient attribute must be set as the URL of the API server from which you request the OAuth token.	<pre> &lt;saml2:Subject&gt;   &lt;saml2:NameID     Format="urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified"&gt;admin   &lt;/saml2:NameID&gt;    &lt;saml2:SubjectConfirmation     Method="urn:oasis:names:tc:SAML:2.0:cm:bearer"&gt;      &lt;saml2:SubjectConfirmationData       NotOnOrAfter="2020-08-21T09:23:24.511Z"       Recipient="http://&lt;api-server&gt;/oauth/token"/&gt;     &lt;/saml2:SubjectConfirmationData&gt;   &lt;/saml2:SubjectConfirmation&gt; &lt;/saml2:Subject&gt; </pre>
<code>&lt;saml2:AttributeStatement&gt;</code> and <code>&lt;saml2:Attribute&gt;</code>	The AttributeStatement element must contain the API key (clientId) that you obtained after you register the client application in <a href="#">Registering Your OAuth2 Client Application</a> [page 17].	<pre> &lt;saml2:AttributeStatement&gt;   &lt;saml2:Attribute     Name="api_key"&gt;      &lt;saml2:AttributeValue       xsi:type="xs:string"&gt;NDU0MDE0MDkwYj***5YTE5MwIxMTNkNjc1Zg&lt;/saml2:AttributeValue&gt;     &lt;/saml2:AttributeValue&gt;   &lt;/saml2:Attribute&gt; &lt;/saml2:AttributeStatement&gt; </pre>
<code>&lt;saml2:Conditions&gt;</code> , <code>NotBefore</code> , <code>NotOnOrAfter</code> , and <code>&lt;saml2:Audience&gt;</code>	The NotBefore and NotOnOrAfter attributes in the <code>&lt;saml2:Conditions&gt;</code> element defines the validity period of the SAML assertion. The <code>&lt;saml2:Audience&gt;</code> element is used to tag the SAML assertion. Any value is accepted except empty value. For example, <a href="http://www.successfactors.com">www.successfactors.com</a> .	<pre> &lt;saml2:Conditions   NotBefore="2020-08-21T09:03:24.511Z"   NotOnOrAfter="2020-08-21T09:23:24.511Z"&gt;    &lt;saml2:AudienceRestriction&gt;      &lt;saml2:Audience&gt;www.successfactors.com&lt;/saml2:Audience&gt;   &lt;/saml2:AudienceRestriction&gt; &lt;/saml2:Conditions&gt; </pre>

## Procedure

1. Download the SAML generator tool from [3031657](#) and extract the files to your local directory.
2. Go to the `SAMLAAssertionGen` directory where the extracted files are located, open the `SAMLAAssertion.properties` file with a text editor, and enter the following values:

<code>tokenUrl</code>	Required	The recipient URL from where the access token is to be generated. This is the OAuth API endpoint of your SAP SuccessFactors API server. Example: <code>https://&lt;api-server&gt;/oauth/token</code>
<code>clientId</code>	Required	The API key you obtained after you register the client application.
<code>userName</code>	Required	The SAP SuccessFactors username used to access the APIs.
<code>privateKey</code>	Required	The X.509 private key used to sign SAML assertion. It can be the private key of a self-signed X.509 certificate or the private key of an X.509 certificate generated by SAP SuccessFactors.
<code>expireInDays</code>	Optional	Set the expiration days of the SAML assertion. The default value is set to 10000.

### Note

The SAML assertion expiration time doesn't affect the OAuth token expiration time.

3. Save the changes.
4. You have two options to generate a SAML assertion:
  - Option 1: Open a command-line tool and go to the `SAMLAAssertionGen` directory. Run the following command:

```
mvn compile exec:java -Dexec.args="SAMLAAssertion.properties"
```

This command downloads required files and generates the SAML assertion using the parameters in the properties file. It can take a few minutes to finish.

- Option 2: If you want to generate a SAML assertion on a machine without Internet access, choose this option.
  1. Open a command-line tool and go to the `SAMLAAssertionGen` directory. Run the following command:

```
mvn clean compile package
```

i Note

This step must be done on a machine with Internet access. The command generates an executable `SAMLAssertionGen-1.0.0.jar` file in a `/target` directory.

2. Copy the `SAMLAssertionGen-1.0.0.jar` file and the `SAMLAssertion.properties` file to the same folder.

i Note

You can copy the files to a machine without Internet access.

3. Run the following command to generate a SAML assertion:

```
java -jar SAMLAssertionGen-1.0.0.jar "SAMLAssertion.properties"
```

## Results

A SAML assertion is generated and displayed in the command-line tool. Here's what the result looks like:

The generated Signed SAML Assertion is:

PD94bWwgdMvYc2lvbj01MS4wIiBlbmNvZGlzZ0iVRGLTgiPz48c2FtbDI6QXNzZXJ0aW9uIHhtbG5zOnNhbWwvPSJ1cm46b2FzaXM6bmFtZXN6dGM6U0FNTDoyLjA6YXNzZXJ0aW9uIiBjRD0iOTU1ZDg0ZS9EtNWYyOC00MjY1LTg0YzcyZyJjNiNtVlMDJjNmU5IiBjC3N1ZULic3RbnQ9IjIwMjEtMDI1MDRMDUy6NDY6NDkuNzMs3IlgVmVyc2lvbj0iMi4wIiB4bWxucz94cz0iAHR0cDovL3d3dy53My5vcmcvMjAwMS9YTUxTY2hlbWEiIHhtbG5zOnhzaT0iaHR0cDovL3d3dy53My5vcmcvMjAwMS9YTUxTY2hlbWEtaW5zdGFuZU1pPjxzYW1smjppJc3N1ZXI+<--MASKED CONTENT-->3d3Ln1Y2Ln1Y2Ln3NmYWN0b3JzLnNvbTtwvc2FtbDI6SXNzZWVpPjxkc3NhbWwvYXRlcUgeIjbnM6ZHM9Imh0dHA6LE93d3cudzMub3JnLzIwMDAvMDkveG1zHNp2YmPiPjxkc3NhbWwvYkF0dHJpYnV0ZVN0YXRlbWVudD48L3NhbWwvOkFzc2VydG1vbj4=

## Next Steps

Copy the SAML assertion and store it securely in your local drive.

**Task overview:** [Authentication Using OAuth 2.0 \[page 15\]](#)

## Related Information

[Registering Your OAuth2 Client Application \[page 17\]](#)[Requesting an Access Token \[page 28\]](#)[Viewing the Validity of an Access Token \[page 29\]](#)

### 3.3.3 Requesting an Access Token

With a SAML assertion, you can now call API `/oauth/token` to request an access token for authentication with the API server.

#### Prerequisites

The `/oauth/token` API follows IP restriction settings in the following tools:

- In [Admin Center](#) > [Password & Login Policy Settings](#) > [Set API login exceptions...](#), you can set access restriction for individual users by IP.
- In [Admin Center](#) > [IP Restriction Management](#), you can set access restriction by IP on the instance level.

Before you request an OAuth token, check the above settings and make sure that the client IP address is allowed to access the corresponding API server. For more information about IP restrictions, see the related information section.

#### Note

If both instance-level and user-level IP restrictions are set, a user can access these APIs if either condition is met.

#### Context

The API returns the token type, expiration time in seconds, and the token value you can use to authorize API requests. An access token expires in 24 hours after it's generated.

#### Example

Here is a sample request:

HTTP Method	POST
URI	<code>https://&lt;API-Server&gt;/oauth/token</code>
Headers	<code>Content-Type: application/x-www-form-urlencoded</code>

#### Parameters

- `company_id`: Required. Your company ID.
- `client_id`: Required. API key generated in [Registering Your OAuth2 Client Application \[page 17\]](#).
- `grant_type`: Required. Set the value to "urn:ietf:params:oauth:grant-type:saml2-bearer".
- `assertion`: Required. Enter the Base64-encoded assertion obtained from [Generating a SAML Assertion \[page 23\]](#).
- `new_token`: Optional. If you have already requested an access token with the same SAML assertion and the token hasn't expired yet, your request returns the same token by default with the remaining time indicated in the `expire_in` field. You can use parameter `new_token=true` to force the server to generate a new access token valid for 24 hours.

---

#### Sample response:

```
{
  "access_token": "eyJ0b2t1bWVbnRlbnQ***ZMm5Td0ifQ==",
  "token_type": "Bearer",
  "expires_in": 86399
}
```

**Task overview:** [Authentication Using OAuth 2.0 \[page 15\]](#)

## Related Information

[Registering Your OAuth2 Client Application \[page 17\]](#)

[Generating a SAML Assertion \[page 23\]](#)

[Viewing the Validity of an Access Token \[page 29\]](#)








[IP Restrictions](#)

## 3.3.4 Viewing the Validity of an Access Token

Use API `/oauth/validate` to verify if an access token is valid.

### Prerequisites

The `/oauth/validate` API follows IP restriction settings in the following tools:

- In  [Admin Center](#)  [Password & Login Policy Settings](#)  [Set API login exceptions...](#) , you can set access restriction for individual users by IP.
- In  [Admin Center](#)  [IP Restriction Management](#) , you can set access restriction by IP on the instance level.

Before you use the API, check the above settings and make sure that the client IP address is allowed to access the corresponding API server. For more information about IP restrictions, see the related information section.

### **i Note**

If both instance-level and user-level IP restrictions are set, a user can access these APIs if either condition is met.

## **Context**

An access token expires within 24 hours after it's generated. You can use this API to check if an access token is still valid.

## **Example**

The following sample request shows how to validate an access token:

HTTP Method	GET
URI	<code>https://&lt;API-Server&gt;/oauth/validate</code>
Headers	<code>Authorization: Bearer &lt;Your access token&gt;</code>

A valid token returns status 200 OK in the header. The response body contains the access token, token type, and expiry time in seconds. Example:

```
{
  "access_token": "<Your Bearer token>",
  "token_type": "Bearer",
  "expires_in": 86312
}
```

**Task overview:** [Authentication Using OAuth 2.0 \[page 15\]](#)

## **Related Information**

[Registering Your OAuth2 Client Application \[page 17\]](#)

[Generating a SAML Assertion \[page 23\]](#)

[Requesting an Access Token \[page 28\]](#)

[IP Restrictions](#)

## 3.4 Enabling Session Reuse

Enable session reuse to reduce the number of sessions and optimize the performance of API calls.

### Context

OData API is not stateful. By default, each OData call needs to go through authentication and authorization. In reality, a user session is created for each API call. Creating user session is a resource-intensive process. Large number of concurrent API calls will stress the server and drastically increase response time. To optimize performance, we encourage you to reuse sessions whenever you can.

### Procedure

1. Login providing the correct authentication information.

The first successful login returns a header, `Set-Cookie` and a CSRF token named `X-CSRF-Token`. For example:

```
HTTP/1.1 201 Created
Server: Apache-Coyote/1.1
X-Powered-By: Servlet 2.4; JBoss-4.3.0.GA_CP09 (build:
SVNTag=JBPAAPP_4_3_0_GA_CP09 date=201011090309)/JBossWeb-2.0
Set-Cookie: JSESSIONID=2oeUIX5Glwl4EZXU7fDWMQ**.ps4bsfapi52t; Path=/odata
X-CSRF-Token: 97sHgXdBw6%2bCKGg3sWAahbT8ztI%3d
Location: https://<hostname>/odata/v2/User('admin')
DataServiceVersion: 1.0
SFODataServerTimeZone: America/Los_Angeles
Content-Type: application/atom+xml; charset=utf-8
Content-Length: 1518
Date: Thu, 14 Aug 2014 05:25:06 GMT
```

#### i Note

As of 2019 Q2 Release, the path for OData sessions has changed from "/" to "/odata" to indicate that it's an OData session. Do not reuse sessions between different protocols (OData API and SFAPI).

2. Store the value of the cookie and CSRF token in the authorization header as a HTTP Cookie (header field names, `Cookie` and `X-CSRF-Token`) and use them in subsequent requests to reuse the session.

Example request header:

```
Cookie: JSESSIONID=2oeUIX5Glwl4EZXU7fDWMQ**.ps4bsfapi52t
X-CSRF-Token: 97sHgXdBw6%2bCKGg3sWAahbT8ztI%3d
```

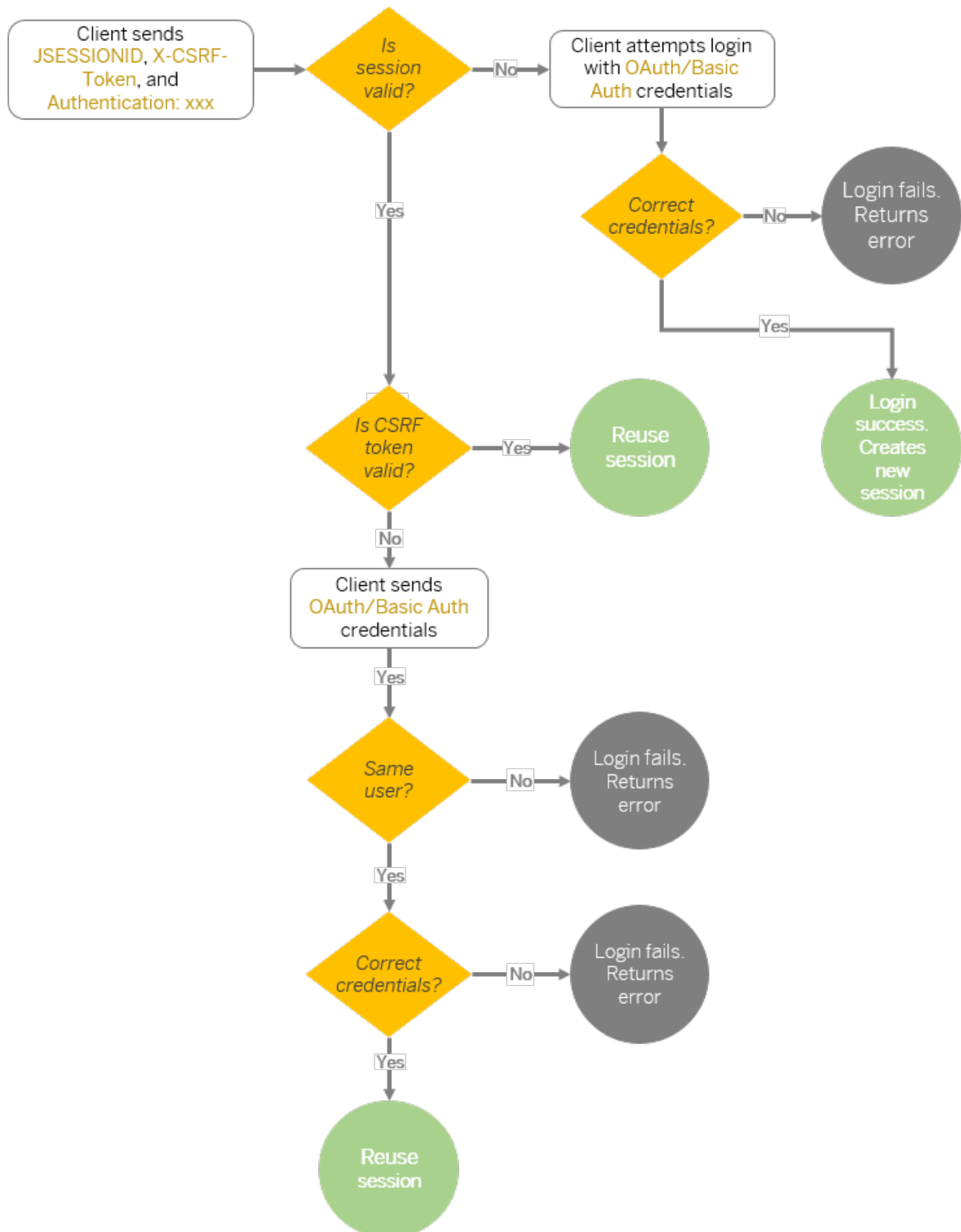
### Results

If the cookie and token are passed correctly, then the login procedure will reuse the session.

If the CSRF token is invalid, you can still reuse the session by providing the correct login credentials of the same user. Note that the login type must be the same as your first successful login.



Session reuse is not possible if you provide incorrect JSESSIONID or enter wrong credentials. The following chart describes how session reuse works:



### Note

- Because sessions use authentication information only once, subsequent requests are not aware of any newer permission changes. Similar to UI login, you need to log out and log in again to allow the latest permissions to take effect.
- An X-CSRF-Token is only valid within the session it was generated. Using the token in a non-matching session will result in an error.
- To optimize server resource, a session is only deactivated when the timeout limit is reached.
- If you have only one API client, we recommend reusing sessions whenever possible. If you have multiple clients running at the same time, reuse sessions within one client only. Do not reuse sessions across clients.

## 3.4.1 Validating a Reused Session

You can validate a reused session with function import `isValidSession`. If a session is valid (value of `name` property is `true`), you can reuse it for your subsequent requests.

To validate a session, make the following call:

HTTP Method	GET
URI	<code>https://&lt;hostname&gt;/odata/v2/isValidSession</code>

If a session is valid, the following response will be returned:

```
{
  "d": {
    "name": true
  }
}
```

## 4 Permissions

List of permissions required for accessing OData API.

Two types of permissions can be set on the entity level:

- Query permissions
- Edit permissions

### Query Permissions

There are three permission levels for querying OData API entities:

Permission Level	Description
Operation level	The operation-level permission checks whether the logged-in user can access the module to which the entities belong.
Row level	The row-level permission checks whether the logged-in user can query an entity.
Field level	The field-level permission checks whether the logged-in user can query a specific property in an entity.

The export permission follows these rules:

Permission Settings	Query Outcome
<a href="#">Export</a> permission is enabled and query entity permission has no target population.	No row-level permission check is required. The user can query all entities.
<a href="#">Export</a> permission is enabled and query entity permission has a target population.	Row-level permission check is required and only entities in the target population are returned.
<a href="#">Export</a> permission is disabled.	Field-level permission check is performed and only fields that the user has permission to access are returned.

### Edit Permissions

There are three permission levels for editing OData API entities:

Permission Level	Description
Operation level	The operation-level checks whether the logged-in user can access the module to which the entities belong.

Permission Level	Description
Row level	The row-level permission checks whether the logged-in user can edit an entity.
Field level	The field-level permission checks whether the logged-in user can edit a specific property of an entity.

The import permission follows these rules:

Permission Settings	Query Outcome
<i>Import</i> permission is enabled.	The user can edit all entities.
<i>Import</i> permission is disabled and the operation type is <code>Insert</code> or <code>Upsert</code> .	The operation can't be executed.
<i>Import</i> permission is disabled and the operation type is <code>Update</code> or <code>Merge</code> .	A field-level permission check is performed and the user is allowed to edit the fields for which they have permission. Note that entities that have a target population also require a row-level permission check before field-level permissions are checked.

# 5 Operations

OData v2 is a standardized protocol for accessing a database. OData entities are described in the metadata document. Accessing the entities is as simple as knowing the entity name, and then referencing the entity through an HTTP call. For example, all SAP SuccessFactors HXM Suite solutions have a `User` Entity that represents a user account. To query user `cgrant`, you can use the following URL:

```
https://<api-server>/odata/v2/User('cgrant')
```

A query can be more complex by including filtering, paging, sorting, and joining of entities (known as expanding in OData).

OData protocol allows the service provider to decide which operations are supported. In SAP SuccessFactors HXM Suite, the supported operations vary by entity. For example, the system does not allow user accounts to be deleted. User accounts can be removed from use only by setting them as inactive. Therefore the `User` entity does not support the delete operation.

This document does not cover the details of each entity behavior for all SAP SuccessFactors HXM Suite solutions. Details for each entity are provided in the OData reference guides. The reference guides provide details about operations, business meaning and business logic, and configuration details such as security and custom-field configuration.

## 5.1 Composing the OData URI

In OData, a URI has the following structure:

```
http://<hostname>/odata/v2/EntitySet[(KeyPredicate)] [/NavPropSingle] [/NavPropCollection] [/ComplexType] [/Property]
```

- **EntitySet:** Name of the OData API entity. An entity set represents the resource exposed by the OData API.
- **KeyPredicate:** A predicate that identifies the key property of the entity. If an entity has a single key property, the key predicate may only include the value of the property. If the entity has a composite key, that is, the entity has multiple key properties, the key predicate must include the key/value pairs to identify an entry. Note that when a key predicate is present in a URI, the `$filter` query option will be disregarded.
- **navPropSingle:** The name of a navigation property defined by the entry associated with the prior path segment. The `navPropSingle` must identify a single entity (that is, have a "to 1" relationship).
- **NavPropCollection:** Same as `NavPropSingle` except it must identify a collection of entries (that is, have a "to many" relationship).
- **ComplexType:** The name of a property of the entity or complex type associated with the prior path segment.
- **Property:** The name of a property of the entity or complex type associated with the prior path segment.

The following table contains examples of a URI and a description of its composition:

Sample URI	Description
<code>http://&lt;hostname&gt;/odata/v2/User</code>	<ul style="list-style-type: none"><li>• Identifies a <code>User</code> Collection.</li><li>• Described by the Entity type named "User" in the meta-data document.</li></ul>
<code>http://&lt;hostname&gt;/odata/v2/User('1')</code>	<ul style="list-style-type: none"><li>• Identifies a single User entry with key value 1.</li><li>• Described by the Entity type named "User" in the service metadata document.</li></ul>
<code>http://&lt;hostname&gt;/odata/v2/User('1')/username</code>	<ul style="list-style-type: none"><li>• Identifies the <code>username</code> property of the User entry with key value 1.</li><li>• Described by the Property named <code>username</code> on the User entity type in the metadata document.</li></ul>
<code>http://&lt;hostname&gt;/odata/v2/User('1')/proxy</code>	<ul style="list-style-type: none"><li>• Identifies the collection of proxy associated with User entry with key value 1.</li><li>• Described by the Navigation Property named <code>proxy</code> on the User entity type in the metadata document.</li></ul>
<code>http://&lt;hostname&gt;/odata/v2/User('1')/proxy/\$count</code>	<ul style="list-style-type: none"><li>• Identifies the number of proxy Entries associated with User 1.</li><li>• Described by the Navigation Property named "proxy" on the "User" entity type in the metadata document.</li></ul>
<code>http://&lt;hostname&gt;/odata/v2/User('1')/proxy('1')/hr/username</code>	<ul style="list-style-type: none"><li>• Identifies the <code>username</code> of the <code>hr</code> for proxy 1 which is associated with User 1.</li><li>• Described by the Property named <code>username</code> on the <code>hr</code> entity type in the metadata document.</li></ul>
<code>http://&lt;hostname&gt;/odata/v2/User('1')/proxy('1')/hr/username/\$value</code>	Same as the URI above, but identifies the raw value of the <code>username</code> property.

## Related Information

[Query with Key Predicate \[page 44\]](#)

## 5.2 Handling Special Characters

URLs are sent over the Internet using the ASCII character set. If a URL contains characters outside the ASCII set, the characters must be converted into a valid ASCII format. Percent encoding, also known as URL encoding, is a mechanism that encodes special characters into ASCII characters that can be understood. It is used in SAP SuccessFactors HXM Suite OData API to handle special characters that appear in the URI or request/response body. In this topic, you'll find a list of special characters and when to encode them in OData API.

## Reserved & unreserved characters

The characters allowed in a URI can be categorized into reserved characters and unreserved characters:

- Reserved characters are the characters for special use in defining the syntax of the URI, for example, the dollar sign ("\$\$") and the question mark ("?"). These special characters may need to be encoded under certain circumstances if you don't want them to be used in their special role.
- Unreserved characters serve no special purposes. They do not need to be percent-encoded.

## List of special characters in SAP SuccessFactors HXM Suite OData API

Percent encoding converts a special character into its corresponding ASCII value with a "%" followed by two hexadecimal digits. Below is a list of special characters, the values after percent encoding, and how to use them:

Special Character	Percent-encoded Value	How to use it in key predicate	How to use it in \$filter
!	%21	As is or encode	As is or encode
#	%23	Encode	Encode
\$	%24	As is or encode	As is or encode
%	%25	Encode	As is or encode
&	%26	As is or encode	Encode
'	%27	Two apostrophes (")	Two apostrophes (")
(	%28	As is or encode	As is or encode
)	%29	As is or encode	As is or encode
*	%2A	As is or encode	As is or encode
+	%2B	As is or encode	Encode
,	%2C	As is or encode	As is or encode
/	%2F	As is	As is or encode
:	%3A	As is or encode	As is or encode
;	%3B	Encode	As is or encode
=	%3D	As is or encode	As is or encode
?	%3F	Encode	As is or encode
@	%40	As is or encode	As is or encode
[	%5B	Encode	As is or encode
]	%5D	Encode	As is or encode

## Examples: When should I percent-encode special characters?

Depending on where the special characters appear, percent-encoding may or may not be needed when you compose a URI. Following the rules listed in the table above, and refer to the following use cases as examples.

### Using special characters in URI

The special characters above should always be percent-encoded when you compose a URI. For example, you want to query cost center **CC#001**. The URI of the query looks like this:

```
https://<api-server>/odata/v2/FOCostCenter?$filter=externalCode eq 'CC%23001'&$select=externalCode,name&$format=JSON
```

In the response body, the externalCode is displayed as "CC#001":

```
{
  "d": {
    "results": [
      {
        "__metadata": {
          "uri": "https://<api-server>/odata/v2/FOCostCenter(externalCode='CC#001',startDate=datetime'2013-01-01T00:00:00')",
          "type": "SFOData.FOCostCenter"
        },
        "externalCode": "CC#001",
        "name": "Special Character Test"
      }
    ]
  }
}
```

### Using special characters in request body

Do not percent-encode special characters in the request body. In the following example, the externalCode of the cost center is **CC%23001** in the payload. This creates a cost center with externalCode **CC%23001** instead of **CC#001**.

```
{
  "__metadata": {
    "uri": "FOCostCenter(externalCode='CC%23001',startDate=datetime'2013-01-01T00:00:00')",
    "type": "SFOData.FOCostCenter"
  },
  "name": "Special Character Test",
  "status": "A"
}
```

When you query the entity, you need to encode the '%' into '%25' to filter the cost center in the result:

```
https://<api-server>/odata/v2/FOCostCenter?$filter=externalCode eq 'CC%2523001'&$select=externalCode,name&$format=JSON
```

In the response body, the externalCode is displayed as "CC%23001":

```
{
  "d": {
    "results": [
      {
        "__metadata": {
          "uri": "https://<api-server>/odata/v2/FOCostCenter(externalCode='CC%23001',startDate=datetime'2013-01-01T00:00:00')",
          "type": "SFOData.FOCostCenter"
        }
      }
    ]
  }
}
```



```

    },
    "externalCode": "CC%23001",
    "name": "Special Character Test"
  }
]
}

```

## Using the apostrophe (') in URI

If the value of a field includes an apostrophe ('), use two apostrophes in the URI. The following example shows how to create a cost center with externalCode **McDonald's**:

```

{
  "__metadata":{
    "uri":"FOCostCenter(externalCode='McDonald's',startDate=datetime'2013-01-01T00:00:00')",
    "type":"SFOData.FOCostCenter"
  },
  "name":"Special Character Test",
  "status":"A"
}

```

Similarly, use two apostrophes to query the created cost center:

```

https://<api-server>/odata/v2/FOCostCenter?$filter=externalCode eq 'McDonald's'&
$select=name,externalCode&$format=JSON

```

## 5.3 Date and Time

The SAP SuccessFactors HXM Suite OData service supports both `Edm.DateTime` and `Edm.DateTimeOffset` data types of the OData protocol. These data types make it easy for integration clients to convert date and time values to different time zones as needed.

### 5.3.1 Using the DateTime Format

For `DateTime` format, the input value is in UTC time, and it is stored and retrieved from the database in the same format.

#### Example of DateTime format

##### Atom Format

**Input:** If the input value in Atom format is 2014-4-22T18:10:10, the OData server treats the input string as UTC time, stores to the database directly.

**Output:** when requested, the date value is retrieved from the database and returned to client with no date conversion.

#### JSON Format

**Input:** If the date is 2014-04-22T18:10:10, then convert it in milliseconds under UTC timezone. Time in milliseconds is 1398190210000, so the input date in json format is /Date(1398190210000)/. When OData receives this value, it converts it to 2014-04-22T18:10:10, and stores it in database.

**Output:** Since the date in database is 2014-4-22T18:10:10, when the date is retrieved, OData converts and formats the time in milliseconds under UTC timezone. The retrieved value is /Date(1398190210000)/

#### i Note

Some modules store the date and truncate the time. In this case, for the input string **2014-04-22T18:10:10**, the stored value would be **2014-04-22T00:00:00**, and the retrieved value might be **2014-04-22T00:00:00**.

In a HANA-based instance, incorrect date time format such as **2014-04-22** is also accepted and treated as **2014-04-22T00:00:00**. In non-HANA instances, an error will occur.

## Query Example for DateTime

The property `lastModifiedWithTZ` exposes handling of date and time at an entity level.

The following API call returns the most recent handling of date and time in Java data format:

```
https://<hostname>/odata/v2/User?$format=json&$select=lastModifiedWithTZ&
$filter=lastModifiedWithTZ ge datetime'2013-12-18T17:19:28'&$top=200
```

Part of the response is:

```
{
  "_metadata": {
    "uri": "https://<localhost>:443/odata/v2/User('eclark1')",
    "type": "SFOData.User"
  },
  "lastModifiedWithTZ": "/Date(1392046843000-0480)/"
}
```

## 5.3.2 Using the DateTimeOffset Format

`DateTimeOffset` takes time zone into account when converting dates.

### Example of DateTimeOffset format (Server location is GMT-4:00)

#### Atom Format

**Input:** If input date is 2014-4-22T18:10:10-04:00, OData converts the input date to server time, because the input date has the same time zone with server time zone, there is no need for time conversion, and 2014-4-22T18:10:10 is stored.

If input date is 2014-4-22T23:10:10+01:00, OData convert the input date to server time, because the input date time zone is different from server time zone (input date is at GMT+1 time zone), the date is stored as 2014-4-22T18:10:10.

**Output:** If date in the database is 2014-4-22T18:10:10, it is returned as 2014-4-22T18:10:10-04:00.

### JSON Format

**Input:** For input date 2014-4-22T18:10:10 in milliseconds of UTC is 1398219010000, so the input date string in json is `"/Date(1398219010000-0240)/"`

**Output:** If the date in the database is 2014-4-22T18:10:10, when retrieved, it is converted to milliseconds of UTC, `/Date(1398219010000-0240)/`.

## Query Example for DateTimeOffset

The property `lastModifiedWithTZ` exposes handling of date and time at an entity level.

The following API call returns the most recent handling of date and time in Java data format:

```
https://<hostname>/odata/v2/User?$format=json&$select=lastModifiedWithTZ&
$filter=lastModifiedWithTZ%20ge%20datetimeoffset%272013-12-18T17:19:28-07:00%27&
$top=200
```

Part of the response is:

```
"uri" : "https://<localhost>:443/odata/v2/User('eclark1')", "type" : "SFOData.User" },
"lastModifiedWithTZ" : "\/Date(1392046843000-0480)\/"
```

### Note

As of Release Q1 2019, `DateTimeOffset` values without timezone information are also accepted in query URLs and edit request payloads. The value will be processed by the server as UTC time.

## 5.4 Query Operations

Learn how query operations work in SAP SuccessFactors HXM Suite OData API.

The query operation uses the `GET` HTTP method against a resource URI to retrieve data. You can query a single entry using a key predicate. You can refine query results with filters, sort them in a particular order, page large amount of data, and limit data size by specifying which properties to be returned. You can also expand to related entities and use date constraints to retrieve effective-dated entries.

[Query with Key Predicate \[page 44\]](#)

[System Query Options \[page 45\]](#)

[Pagination \[page 67\]](#)

The OData API provides several pagination options for query results.

[Querying Effective-Dated Entities \[page 76\]](#)

## 5.4.1 Query with Key Predicate

In an OData URI, a key predicate identifies the key property or key properties of the resource specified. If an entity has a single key property, the key predicate may include only the value to be able to uniquely identify an entry. If an entity has multiple key properties, that is, a composite key, the key predicate must include all key/value pairs to be able to identify a unique entry.

Below are two examples of querying entries with a key predicate:

```
https://<API-Server>/odata/v2/FormHeader('123')
```

```
https://<API-Server>/odata/v2/PicklistLabel(locale='en_US',optionId='123')
```

### Query with key predicate and system query options

A URI with key predicate identifies only one entry from the specified OData resource. Therefore, when a key predicate is present in a URI, the following system query options are ignored:

- \$filter
- \$top
- \$skip
- \$orderby

**Parent topic:** [Query Operations \[page 43\]](#)

### Related Information

[System Query Options \[page 45\]](#)

[Pagination \[page 67\]](#)

[Querying Effective-Dated Entities \[page 76\]](#)

[System Query Options \[page 45\]](#)

## 5.4.2 System Query Options

System query options are query string parameters that can be used to control the order and amount of data returned for the URI. All system query options start with the "\$" character. You can join multiple options with the "&" character. In this section, you'll learn how each system query options work and how they work together.

### [\\$orderby \[page 46\]](#)

Use the `$orderby` system query option to specify an expression for determining what values are used to order the collection of entries identified by the URI.

### [\\$top \[page 47\]](#)

This topic shows examples of the `$top` system query option used in a URI.

### [\\$skip \[page 48\]](#)

This topic shows examples of the `$skip` system query option used in a URI.

### [\\$filter \[page 49\]](#)

This topic shows examples of the logical operators, arithmetic operators, grouping operators, and customized operators including customized string functions that can be used with the `$filter` system query option.

### [\\$expand \[page 55\]](#)

This topic shows examples of the `$expand` system query option used in a URI.

### [\\$format \[page 56\]](#)

This topic shows examples of the `$format` system query option used in a URI and the Response received for each one.

### [\\$select \[page 58\]](#)

You can use the `$select` system query option in a URI to specify a subset of properties to be returned by the OData service.

### [\\$count \[page 59\]](#)

Use the `$count` system query option to identify the number of entries represented by the resource the URI.

### [\\$inlinecount \[page 61\]](#)

Use the `$inlinecount` system query option to include a count number of the entries identified by the resource path section of the URI.

### [Examples of Queries with Multiple Keywords \[page 62\]](#)

Examples of queries with multiple keywords.

**Parent topic:** [Query Operations \[page 43\]](#)

## Related Information

[Query with Key Predicate \[page 44\]](#)

[Pagination \[page 67\]](#)

[Querying Effective-Dated Entities \[page 76\]](#)

## 5.4.2.1 \$orderby

Use the `$orderby` system query option to specify an expression for determining what values are used to order the collection of entries identified by the URI.

The `$orderby` system query option only works when multiple entries are returned.

### ❖ Example

Query all users in the system and sort the results by the `username` property:

```
https://<API-Server>/odata/v2/User?$orderby=username
```

### ❖ Example

Query all users in the system and sort the results by the `username` property in ascending order:

```
https://<API-Server>/odata/v2/User?$orderby=username asc
```

### ❖ Example

Query all users in the system and sort the results by the `username` property in descending order, and subsequently sort the results by the `username` property of the user's HR:

```
https://<API-Server>/odata/v2/User?$orderby=username,hr/username desc
```

### i Note

The system query option sorts the top level entity only; child entities aren't sorted. When using `$orderby` on a 1: many navigation properties, the order isn't predicable.

## Sorting of String-Type Numeric Characters

Numbers of a string-type property are sorted in lexical order (lexicographic sorting) rather than by their numeric values. For example, if you sort a collection of entries by a string-type ID field in ascending order, ID "123" comes before ID "45".

### ❖ Example

Query `TodoEntryV2` and sort the results by the `categoryId` property in **ascending** order:

```
{
  "d": {
    "results": [
      {
        "__metadata": {
          "uri": "https://<API-Server>/odata/v2/TodoEntryV2(185114M)",
          "type": "SFOData.TodoEntryV2"
        },
        "categoryId": "44"
      },
      {

```

```

        "__metadata": {
          "uri": "https://<API-Server>/odata/v2/ToDoEntryV2(185260M)",
          "type": "SFOData.ToDoEntryV2"
        },
        "categoryId": "5"
      },
    {
      "__metadata": {
        "uri": "https://<API-Server>/odata/v2/ToDoEntryV2(162942M)",
        "type": "SFOData.ToDoEntryV2"
      },
      "categoryId": "57"
    },
    {
      "__metadata": {
        "uri": "https://<API-Server>/odata/v2/ToDoEntryV2(185684M)",
        "type": "SFOData.ToDoEntryV2"
      },
      "categoryId": "59"
    }
  ],
}

```

Parent topic: [System Query Options \[page 45\]](#)

## Related Information

- [\\$top \[page 47\]](#)
- [\\$skip \[page 48\]](#)
- [\\$filter \[page 49\]](#)
- [\\$expand \[page 55\]](#)
- [\\$format \[page 56\]](#)
- [\\$select \[page 58\]](#)
- [\\$count \[page 59\]](#)
- [\\$inlinecount \[page 61\]](#)
- [Examples of Queries with Multiple Keywords \[page 62\]](#)

### 5.4.2.2 \$top

This topic shows examples of the `$top` system query option used in a URI.

To learn about the OData System query options used in the example URIs in this section, visit [www.odata.org](http://www.odata.org), and search for "OData Version 2.0 Uri Conventions".

EXAMPLE URI	DESCRIPTION
<code>https://&lt;API-Server&gt;/odata/v2/User?\$top=5</code>	The first 5 <code>User</code> entries are returned and the entries are sorted using a schema determined by the OData service.

EXAMPLE URI	DESCRIPTION
<code>https://&lt;API-Server&gt;/odata/v2/User?\$top=5&amp;\$orderby=username desc</code>	The first 5 User entries are returned in descending order and sorted by the username property.

Parent topic: [System Query Options \[page 45\]](#)

## Related Information

[\\$orderby \[page 46\]](#)

[\\$skip \[page 48\]](#)

[\\$filter \[page 49\]](#)

[\\$expand \[page 55\]](#)

[\\$format \[page 56\]](#)

[\\$select \[page 58\]](#)

[\\$count \[page 59\]](#)

[\\$inlinecount \[page 61\]](#)

[Examples of Queries with Multiple Keywords \[page 62\]](#)

### 5.4.2.3 \$skip

This topic shows examples of the `$skip` system query option used in a URI.

To learn more about the OData System query options used in the example URIs in this section, visit [www.odata.org](http://www.odata.org), and search for "OData Version 2.0 Uri Conventions".

Example	Description
<code>/odata/v2/User?\$skip=2</code>	The set of User entries starting with the third user.
<code>/odata/v2/User?\$skip=2&amp;\$top=2&amp;\$orderby=username</code>	The third and fourth User entries from the collection of all User entities when the collection is sorted by username in ascending order.

Parent topic: [System Query Options \[page 45\]](#)

## Related Information

[\\$orderby \[page 46\]](#)

[\\$top \[page 47\]](#)

[\\$filter \[page 49\]](#)



[\\$expand \[page 55\]](#)  
[\\$format \[page 56\]](#)  
[\\$select \[page 58\]](#)  
[\\$count \[page 59\]](#)  
[\\$inlinecount \[page 61\]](#)  
[Examples of Queries with Multiple Keywords \[page 62\]](#)

## 5.4.2.4 \$filter

This topic shows examples of the logical operators, arithmetic operators, grouping operators, and customized operators including customized string functions that can be used with the `$filter` system query option.

You can use the `$filter` system query option in the URI to limit the number of results returned in an OData API query. For example, the following URI filters the status field in the User entity to return all active users:

```
https://<API-Server>/odata/v2/User?$filter=status eq 't'
```

### Note

- You can only filter top-level entities and child entities that have 1:1 relationships with their parents. When you apply `$filter` on an entity that has one-to-many (1:M) relationship with its children, the child entities are not filtered. The query returns all parents with full list of child entities even though only one child meets the condition in the filter.
- When a key predicate is present in the URI, the `$filter` option is disregarded.

To learn more about the OData System query options used in the example URIs in this section, visit [www.odata.org](http://www.odata.org), and search for "OData Version 2.0 Uri Conventions".

## Logical Operators

LOGICAL OPERATOR	DESCRIPTION	EXAMPLE
eq	Equal	Finds a User entity whose hr/username is <b>cgrant</b> (hr is the navigation property): <code>https://&lt;API-Server&gt;/odata/v2/User?\$filter=hr/username eq 'cgrant'</code> . Finds a User Entity whose username property is <b>cgrant</b> : <code>https://&lt;API-Server&gt;/odata/v2/User?\$filter=username eq 'cgrant'</code> .
ne	Not equal	<code>https://&lt;API-Server&gt;/odata/v2/User?\$filter=hr/username ne 'London'</code>
gt	Greater than	Finds a PickListLabel Entity whose id is greater than 2000: <code>https://&lt;API-Server&gt;/odata/v2/PicklistLabel?\$filter=id gt 20000</code>

LOGICAL OPERATOR	DESCRIPTION	EXAMPLE
ge	Greater than or equal	<code>https://&lt;API-Server&gt;/odata/v2/PicklistLabel?\$filter=id ge 10</code>
lt	Less than	<code>https://&lt;API-Server&gt;/odata/v2/PicklistLabel?\$filter=id lt 20</code>
le	Less than or equal	<code>https://&lt;API-Server&gt;/odata/v2/PicklistLabel?\$filter=id le 100</code>
and	Logical and	<code>https://&lt;API-Server&gt;/odata/v2/PicklistLabel?\$filter=id le 1005 and id gt 1000</code> to find PicklistLabel whose id is within range [1000,1005]
or	Logical or	<code>https://&lt;API-Server&gt;/odata/v2/PicklistLabel?\$filter=id le 3.5 or id gt 200</code>
not	Logical negation	<code>https://&lt;API-Server&gt;/odata/v2/User?\$filter=not endswith(username, 'grant')</code>

### Note

There's a limit to the number of expressions you can use in a single query. This number varies from server to server. When the limit is reached, an error occurs. To solve this problem, reduce the number of expressions in your query.

## Arithmetic Operators

ARITHMETIC OPERATOR	DESCRIPTION	EXAMPLE
add	Addition	<code>https://&lt;API-Server&gt;/odata/v2/PicklistLabel?\$filter=id add 5 gt 10</code>
sub	Subtraction	<code>https://&lt;API-Server&gt;/odata/v2/PicklistLabel?\$filter=id sub 5 gt 10</code>
mul	Multiplication	<code>https://&lt;API-Server&gt;/odata/v2/PicklistLabel?\$filter=id mul 2 gt 2000</code>
div	Division	<code>https://&lt;API-Server&gt;/odata/v2/PicklistLabel?\$filter=id div 2 gt 4</code>
mod	Modulo	<code>https://&lt;API-Server&gt;/odata/v2/PicklistLabel?\$filter=id mod 2 eq 0</code>

## Grouping Operator

GROUPING OPERATOR	DESCRIPTION	EXAMPLE
()	Precedence grouping.	<code>https://&lt;API-Server&gt;/odata/v2/PicklistLabel?\$filter=(id sub 5) gt 10</code>

## Customized Operators

CUSTOMIZED OPERATOR	DESCRIPTION	EXAMPLE
in	In clause	Identifies <code>User</code> entities whose <code>userId</code> is equal to one specified by an <code>in</code> clause: <code>https://&lt;API-Server&gt;/odata/v2/User?\$filter=userId in 'ctsel','mhuang1','flynch1'&amp;\$select=username,userId</code>

**i Note**

Due to the limit of number of expressions (1000) in a list, OData API doesn't accept an `IN` clause exceeding 1000 expressions in `$filter`.

CUSTOMIZED OPERATOR	DESCRIPTION	EXAMPLE
like	Like clause	<p>Identifies User entities whose <code>userId</code> property <b>cgrant</b> may be retrieved with a <code>like</code> clause:</p> <pre>https://&lt;API-Server&gt;/odata/v2/User? \$filter=userId like 'cgrant'</pre> <p>Identifies User entities whose <code>userId</code> property is equivalent to <b>cgrant</b>:</p> <pre>\$filter=userId eq 'cgrant'</pre> <p>Identifies User entities whose <code>userId</code> property <b>cgrant</b> may be retrieved with a <code>like</code> clause:</p> <pre>https://&lt;API-Server&gt;/odata/v2/User? \$filter=userId like 'cgrant%'</pre> <p>Identifies User entities whose <code>userId</code> property starts with <b>cgrant</b>:</p> <pre>\$filter=startswith(userId, 'cgrant')</pre> <p>Identifies User entities whose <code>userId</code> property ends with <b>cgrant</b>:</p> <pre>https://&lt;API-Server&gt;/odata/v2/User? \$filter=userId like '%cgrant'</pre> <p>This is equivalent to</p> <pre>\$filter=endswith(userId, 'cgrant')</pre> <p>Identifies User entities whose <code>userId</code> property contains string <b>cgrant</b>.</p> <pre>https://&lt;API-Server&gt;/odata/v2/User? \$filter=userId like '%cgrant%'</pre> <p>Identifies User entities whose <code>userId</code> property contains string <b>cgrant</b> and is case insensitive.</p> <pre>https://&lt;API-Server&gt;/odata/v2/User? \$filter=tolower(userId) like ' %cgrant%'</pre> <pre>https://&lt;API-Server&gt;/odata/v2/User? \$filter=toupper(userId) like ' %cgrant%'</pre> <p>Identifies User entities whose <code>userId</code> is like <b>cgrant</b> and unions with <code>username</code> like <b>cgrant</b>.</p>

CUSTOMIZED OPERATOR	DESCRIPTION	EXAMPLE
		<pre>https://&lt;API-Server&gt;/odata/v2/User? \$filter=toupper(userId) like ' %cgrant%' or tolower(username) like '%cgrant%'</pre> <p>Identifies User entities whose userId is like <b>cgrant</b> and intersects with username like <b>cgrant</b>.</p> <pre>https://&lt;API-Server&gt;/odata/v2/User? \$filter=toupper(userId) like ' %cgrant%' and tolower(username) like '%cgrant%'</pre>

## String Functions

In addition to operators, a set of functions is also defined for use with the `$filter` query option. The following table lists the available functions.

### Note

ISNULL or COALESCE operators are not defined. Instead, there's a null literal that can be used in comparisons. If your query contains a single quote, it must be escaped by another one if it appears with single-quoted string.

STRING FUNCTION	EXAMPLE
<code>bool endswith(string p0, string p1)</code>	<p>Finds all usernames that end with <b>Futterkiste</b>:</p> <pre>https://&lt;API-Server&gt;/odata/v2/User? \$filter=endswith(username, 'Futterkiste')</pre>
<code>bool startswith(string p0, string p1)</code>	<p>Finds all usernames that start with <b>Alfr</b>:</p> <pre>https://&lt;API-Server&gt;/odata/v2/User? \$filter=startswith(username, 'Alfr')</pre>
<code>bool substringof(string p0, string p1)</code>	<p>Case-insensitive search for username field:</p> <pre>https://&lt;API-Server&gt;/odata/v2/User? \$filter=substringof('futter', tolower( username)) eq true</pre>

## STRING FUNCTION

## EXAMPLE

`string tolower(string p0)`

Compare username in lower case to the literal value **futterkiste**:

`https://<API-Server>/odata/v2/User?  
$filter=tolower(username) eq 'futterkiste'`

`string toupper(string p0)`

Compare username in upper case to the literal value 'FUTTERKISTE':

`https://<API-Server>/odata/v2/User?  
$filter=toupper(username) eq 'FUTTERKISTE'`

`string trim(string p0)`

Trim leading and trailing whitespaces before comparing values:

`https://<API-Server>/odata/v2/User?  
$filter=trim(username) eq 'Futterkiste'`

The `$filter` option can also be used in a query using a `datetime` parameter:

## QUERY EXAMPLE

`https://<API-Server>/odata/v2/User?  
$format=json&$select=lastModified&  
$filter=lastModified%20ge%20datetime  
%272003-12-18T17:19:28%27&$top=200`

The query returns last modified date in Java data format (the milliseconds since January 1, 1970, 00:00:00 GMT).

Parent topic: [System Query Options \[page 45\]](#)

## Related Information

[\\$orderby \[page 46\]](#)

[\\$top \[page 47\]](#)

[\\$skip \[page 48\]](#)

[\\$expand \[page 55\]](#)

[\\$format \[page 56\]](#)

[\\$select \[page 58\]](#)

[\\$count \[page 59\]](#)

[\\$inlinecount \[page 61\]](#)

[Examples of Queries with Multiple Keywords \[page 62\]](#)

## 5.4.2.5 \$expand

This topic shows examples of the `$expand` system query option used in a URI.

To learn about the OData System query options used in the example URIs in this section, visit [www.odata.org](http://www.odata.org), and search for "OData Version 2.0 Uri Conventions".

EXAMPLE	DESCRIPTION
<code>https://&lt;API-Server&gt;/odata/v2/User?\$expand=proxy</code>	<ul style="list-style-type: none"><li>Identifies the collection of <code>User</code> entities as well as each of the <code>proxy</code> associated with each <code>User</code>.</li><li>Described by the Entity Set named "User" and the "proxy" navigation property on the <code>User</code> Entity Type in the service metadata document.</li></ul>
<code>https://&lt;API-Server&gt;/odata/v2/User?\$expand=hr/matrixManager</code>	<ul style="list-style-type: none"><li>Identifies the collection of <code>User</code> entities as well as each of the <code>hr</code> associated with each <code>User</code>. In addition, the URI also identifies the <code>matrixManager</code> navigation property associated with each <code>hr</code>.</li><li>Described by the Entity Set named "User", the "hr" navigation property on the "User" Entity Type, and the <code>matrixManager</code> navigation property on the "hr" entity type in the service metadata document.</li></ul>
<code>https://&lt;API-Server&gt;/odata/v2/User?\$expand=hr,proxy</code>	<ul style="list-style-type: none"><li>Identifies the collection of <code>User</code> entities as well as the <code>hr</code> and <code>proxy</code> associated with each <code>User</code>.</li><li>Described by the Entity Set named "User" as well as the "hr" and "proxy" navigation properties on the "User" Entity Type in the service metadata document.</li></ul>

Parent topic: [System Query Options \[page 45\]](#)

### Related Information

[\\$orderby \[page 46\]](#)

[\\$top \[page 47\]](#)

[\\$skip \[page 48\]](#)

[\\$filter \[page 49\]](#)

[\\$format \[page 56\]](#)

[\\$select \[page 58\]](#)

[\\$count \[page 59\]](#)

[\\$inlinecount \[page 61\]](#)

[Examples of Queries with Multiple Keywords \[page 62\]](#)

## 5.4.2.6 \$format

This topic shows examples of the `$format` system query option used in a URI and the Response received for each one.

To learn about the OData System query options that will be used in the example URIs in this section, visit [www.odata.org](http://www.odata.org), and search for "OData Version 2.0 Uri Conventions".

\$FORMAT VALUE	RESPONSE MEDIA TYPE
atom	application/atom+xml
json	application/json

The following table shows examples of the `$format` keyword:



**\$FORMAT**

```
https://<API-Server>/odata/v2/User?  
$format=atom
```

**EXAMPLE RESPONSE**

```
<?xml version='1.0' encoding='utf-8'?>  
<feed xmlns="https://www.w3.org/2005/  
Atom" xmlns:m="https://  
schemas.microsoft.com/ado/2007/08/  
dataservices/metadata" xmlns:d="https://  
schemas.microsoft.com/ado/2007/08/  
dataservices" xml:base="https://<API-  
Server>/odata/v2/">  
  <title type="text">Picklist</title>  
  <id>https://<API-Server>/odata/v2/  
Picklist</id>  
  <updated>2013-07-29T07:10:44Z</  
updated>  
  <link rel="self" title="Picklist"  
href="Picklist" />  
  <entry>  
    <id>https://<API-Server>/  
odata/v2/  
Picklist('CandidateStatus')</id>  
    <title type="text" />  
    <updated>2013-07-29T07:10:44Z</  
updated>  
    <author>  
      <name />  
    </author>  
    <link rel="edit"  
title="Picklist"  
href="Picklist('CandidateStatus')"/>  
    <link rel="https://  
schemas.microsoft.com/ado/2007/08/  
dataservices/related/picklistOptions"  
type="application/atom+xml;type=entry"  
title="picklistOptions"  
href="Picklist('CandidateStatus')/  
picklistOptions" />  
    <category  
term="SFOData.Picklist" scheme="https://  
schemas.microsoft.com/ado/2007/08/  
dataservices/scheme" />  
    <content type="application/xml">  
      <m:properties>  
  
<d:picklistId>CandidateStatus</  
d:picklistId>  
      </m:properties>  
    </content>  
  </entry>  
</feed>
```

## \$FORMAT

`https://<API-Server>/odata/v2/User?  
$format=json`

## EXAMPLE RESPONSE

```
d": {  
  "results": [{  
    "__metadata": {  
      "uri": "https://<API-  
Server>/odata/v2/  
Picklist('CandidateStatus')",  
      "type":  
        "SFOData.Picklist"  
    },  
    "picklistId":  
      "CandidateStatus",  
    "picklistOptions": {  
      "__deferred": {  
        "uri":  
          "Picklist('CandidateStatus')/  
picklistOptions"  
      }  
    }  
  }]  
}
```

Parent topic: [System Query Options \[page 45\]](#)

## Related Information

[\\$orderby \[page 46\]](#)

[\\$top \[page 47\]](#)

[\\$skip \[page 48\]](#)

[\\$filter \[page 49\]](#)

[\\$expand \[page 55\]](#)

[\\$select \[page 58\]](#)

[\\$count \[page 59\]](#)

[\\$inlinecount \[page 61\]](#)

[Examples of Queries with Multiple Keywords \[page 62\]](#)

### 5.4.2.7 \$select

You can use the `$select` system query option in a URI to specify a subset of properties to be returned by the OData service.

The value of the `$select` query option is a list of comma-separated properties or navigation properties. By limiting the number of fields returned, this query option can reduce the query data size and improve performance.

To learn about the OData System query options used in the example URIs in this section, visit [www.odata.org](http://www.odata.org), and search for "OData Version 2.0 Uri Conventions".

The following set of examples uses the data model available at [https://<API-Server>/odata/v2/\\$metadata](https://<API-Server>/odata/v2/$metadata) to describe the semantics for a base set of URIs using the `$select` system query option. From these base cases, the semantics of longer URIs are defined by composing the rules below:

EXAMPLE URI	DESCRIPTION
<a href="https://&lt;API-Server&gt;/odata/v2/User?\$select=username,userId">https://&lt;API-Server&gt;/odata/v2/User?\$select=username,userId</a>	<ul style="list-style-type: none"> <li>• <code>username</code> and <code>userId</code> property values are returned for each <code>User</code> entry.</li> <li>• If the <code>\$select</code> query option had listed a property that identified a <code>Complex Type</code>, then all properties defined on the <code>Complex Type</code> must be returned.</li> </ul>
<a href="https://&lt;API-Server&gt;/odata/v2/User?\$select=username,proxy">https://&lt;API-Server&gt;/odata/v2/User?\$select=username,proxy</a>	<code>username</code> property value and a link to the related proxy Entry is returned for each <code>User</code> .
<a href="https://&lt;API-Server&gt;/odata/v2/User?\$select=username,proxy&amp;\$expand=proxy/hr">https://&lt;API-Server&gt;/odata/v2/User?\$select=username,proxy&amp;\$expand=proxy/hr</a>	All the properties of the Entries identified by the <code>proxy</code> and <code>hr</code> navigation properties are returned.
<a href="https://&lt;API-Server&gt;/odata/v2/User?\$select=username,proxy/username&amp;\$expand=proxy">https://&lt;API-Server&gt;/odata/v2/User?\$select=username,proxy/username&amp;\$expand=proxy</a>	In a response from an OData service, the <code>username</code> property is included and <code>proxy</code> Entries with a <code>username</code> property is included. But, instead of including the fully expanded proxy entries, each <code>proxy</code> contains a <code>user</code> property.

Parent topic: [System Query Options \[page 45\]](#)

## Related Information

[\\$orderby \[page 46\]](#)

[\\$top \[page 47\]](#)

[\\$skip \[page 48\]](#)

[\\$filter \[page 49\]](#)

[\\$expand \[page 55\]](#)

[\\$format \[page 56\]](#)

[\\$count \[page 59\]](#)

[\\$inlinecount \[page 61\]](#)

[Examples of Queries with Multiple Keywords \[page 62\]](#)

### 5.4.2.8 \$count

Use the `$count` system query option to identify the number of entries represented by the resource the URI.

The `$count` query option can be used standalone or with the `$filter` query option. When it's used to count filtered entries, it must be placed before the `$filter` option.

Here are some examples:

- The following query returns the total number of users in the system:

```
https://<API-Server>/odata/v2/User/$count
```

- If you add a `$filter` to the same query, the `$count` option returns the number of filtered results:

```
https://<API-Server>/odata/v2/User/$count?$filter=status eq 'F'
```

## How Is the Count Calculated?

In SAP SuccessFactors OData API, the `$count` option calculates the number of identified entries after whatever business logic is implemented in the entity. Here are two examples to help you understand the design.

### ❖ Example

The following query returns the number of all active users in the system by default:

```
https://<API-Server>/odata/v2/User/$count
```

If you want to count both active and inactive users, apply a filter option and call out the status explicitly in the query:

```
https://<API-Server>/odata/v2/User/$count?$filter=status in 't', 'T', 'f', 'F'
```

### ❖ Example

The following query returns position entries that are active on the present date by default:

```
https://<API-Server>/odata/v2/Position/$count
```

If you want the number of all position entries including historical and future positions, you need to apply a time parameter to the query:

```
https://<API-Server>/odata/v2/Position/$count?toDate=9999-12-31
```

Parent topic: [System Query Options \[page 45\]](#)

## Related Information

[\\$orderby \[page 46\]](#)

[\\$top \[page 47\]](#)

[\\$skip \[page 48\]](#)

[\\$filter \[page 49\]](#)

[\\$expand \[page 55\]](#)

[\\$format \[page 56\]](#)

[\\$select \[page 58\]](#)

[\\$inlinecount \[page 61\]](#)

[Examples of Queries with Multiple Keywords \[page 62\]](#)

[\\$filter \[page 49\]](#)

## 5.4.2.9 \$inlinecount

Use the `$inlinecount` system query option to include a count number of the entries identified by the resource path section of the URI.

The `$inlinecount` calculates all entries identified by the resource path after applying any `$filter` query options. The difference between `$inlinecount` and `$count` is that `$count` only returns a count number in the response while `$inlinecount` returns both entries and the count number.

### Parameter Values

<code>\$inlinecount</code> Value	Description
<b>none</b>	The response doesn't include a count. This is equivalent to a URI that doesn't include an <code>\$inlinecount</code> option.
<b>allpages</b>	The response includes a count of the number of entities in the collection identified by the URI after applying any filter query option in the URI.

### Use Cases

The following query identifies a list of countries/regions with a count number at the end of the response:

```
https://<API-Server>/odata/v2/Country?$inlinecount=allpages
```

Here's a snippet of the response that contains the count number:

```
{
  "d": {
    "results": [
      ... ..
    ],
    "__count": "251"
  }
}
```

The following query identifies all users with last name **Smith** and the count number of the entries:

```
https://<API-Server>/odata/v2/User?$filter=lastName eq 'Smith'&$inlinecount=allpages
```

Sample response:

#### Sample Code

```
{
  "d": {
    "results": [
      ...
    ],
    "__count": "8"
  }
}
```

**Parent topic:** [System Query Options \[page 45\]](#)

## Related Information

[\\$orderby \[page 46\]](#)

[\\$top \[page 47\]](#)

[\\$skip \[page 48\]](#)

[\\$filter \[page 49\]](#)

[\\$expand \[page 55\]](#)

[\\$format \[page 56\]](#)

[\\$select \[page 58\]](#)

[\\$count \[page 59\]](#)

[Examples of Queries with Multiple Keywords \[page 62\]](#)

## 5.4.2.10 Examples of Queries with Multiple Keywords

Examples of queries with multiple keywords.

**Parent topic:** [System Query Options \[page 45\]](#)

## Related Information

[\\$orderby \[page 46\]](#)

[\\$top \[page 47\]](#)

[\\$skip \[page 48\]](#)

[\\$filter \[page 49\]](#)

[\\$expand \[page 55\]](#)

[\\$format \[page 56\]](#)

[\\$select \[page 58\]](#)

[\\$count \[page 59\]](#)

[\\$inlinecount \[page 61\]](#)

### 5.4.2.10.1 Query with nested-navigation properties with \$select

This topic explains the use of \$select in a query.

The following URIs are constructed differently but provide the same response:

```
http://<hostname>/odata/v2/User?$select=userId,username,hr&$format=json&$top=1
```

```
http://<hostname>/odata/v2/User?$select=userId,username,hr/manager/proxy&$format=json&$top=1
```

Each URI identifies all `User` Entities. Instead of having the response contain all properties for each Entity, the \$select option ensures the response contains only the three properties specified in the URI. . You can specify a simple property, a navigation property, or a nested-navigation property. For a simple property, the response contains results. For a navigation property, the response is a deferred link , whatever the depth of the property.

Sample response:

```
{ "__metadata" : { "uri" : "http://localhost:8080/odata/v2/User('mhuang1')", "type" :
"SFOData.User" }, "userId" : "mhuang1", "username" : "mhuang", "manager" :
{ "__deferred" : { "uri" : "http://localhost:8080/odata/v2/User('mhuang1')/
manager" } } }
```

### 5.4.2.10.2 Query with \$select and \$expand

The following example shows the \$select and \$expand options used together in a URI:

```
http://<hostname>/odata/v2/User?$select=userId,manager/hr/manager&$format=json&
$expand=manager/hr
```

The URI identifies all entities of a `User` Entity Set. The response is similar to the response from a URI that uses nested-navigation properties with the \$select option, but the \$expand option can make a difference in the results that appear in the response. For example:

- If a navigation property gets expanded, the property shows as a flat object.
- If a navigation property is not expanded, it shows as a deferred link.
- If the nested-navigation property `manager/hr/manager` gets expanded, it shows a manager object.
- Because the last entry in a nested navigation doesn't get expanded, the manager object is given a nested `hr` property. The `hr` property will have a deferred link named [manager](#) which refers to the actual object. Unselected properties of expanded objects will be trimmed..

Sample Response:

```
{
  "__metadata" : {
```

```

"uri" : "http://localhost:8080/odata/v2/User('asлива_upsert_JAM_200')", "type" :
"SFOData.User"
}, "userId" : "asлива_upsert_JAM_200", "manager" : {
  "__metadata" : {
    "uri" : "http://localhost:8080/odata/v2/User('asлива_upsert_JAM_201')",
    "type" : "SFOData.User"
  },
  "hr" : {
    "__metadata" : {
      "uri" : "http://localhost:8080/odata/v2/
User('asлива_upsert_b11_112_inline_manager11')", "type" : "SFOData.User"
    },
    "manager" : {
      "__deferred" : {
        "uri" : "http://localhost:8080/odata/v2/
User('asлива_upsert_b11_112_inline_manager11')/manager"
      }
    }
  }
}
}

```

### 5.4.2.10.3 Query links with \$select and \$expand

The following example shows \$select with \$expand:

```

http://<hostname>/odata/v2/User('asлива_upsert_JAM_200')/proxy?
$select=userId,manager/hr/manager&$format=json&$expand=manager/hr

```

In the response, the \$expand option expands the selected navigation property. Unselected data columns do not get expanded. Set entities of User/proxy are returned.

Sample Response:

```

[
  {
    "__metadata" : {
      "uri" : "http://localhost:8080/odata/v2/User('asлива_upsert_JAM_197')", "type" :
      "SFOData.User"
    }, "userId" : "asлива_upsert_JAM_197", "manager" : null
  },
  {
    "__metadata" : {
      "uri" : "http://localhost:8080/odata/v2/User('asлива_upsert_JAM_198')", "type" :
      "SFOData.User"
    }, "userId" : "asлива_upsert_JAM_198", "manager" : null
  },
  {
    "__metadata" : {
      "uri" : "http://localhost:8080/odata/v2/
User('asлива_upsert_b11_112_inline_proxy')", "type" : "SFOData.User"
    }, "userId" : "asлива_upsert_b11_112_inline_proxy", "manager" : null
  },
  {
    "__metadata" : {
      "uri" : "http://localhost:8080/odata/v2/
User('asлива_upsert_b11_112_inline_proxy_1')", "type" : "SFOData.User"
    }, "userId" : "asлива_upsert_b11_112_inline_proxy_1", "manager" : null
  }
]

```



## 5.4.2.10.4 Query with \$select, \$expand, and \$filter

The following example shows a query with multiple options:

```
https://<hostname>/odata/v2/User?$format=JSON&$select=userId,username,proxy/
userId,proxy/username&$expand=proxy&$filter=proxy/userId eq 'admin'
```

The response identifies a set of users who has a `proxy` with `userId` value `admin`. After the users are identified, the response expands the `proxy` navigation property and retrieves the columns as specified in the `$select` option.

### Note

In a 1:M navigation relation, `$filter` only works on the base object. Expanded data will not be filtered. In this case, `$filter=proxy/userId eq 'admin'` serves as a search criteria to identify user 103004. In the response, both proxies of the user are retrieved.

Sample Response:

```
{
  "d": {
    "results": [
      {
        "_metadata": {
          "uri": "https://<hostname>/odata/v2/User('103004')",
          "type": "SFODData.User"
        },
        "userId": "103004",
        "username": "vwagner",
        "proxy": {
          "results": [
            {
              "_metadata": {
                "uri": "https://<hostname>/odata/v2/User('admin')",
                "type": "SFODData.User"
              },
              "userId": "admin",
              "username": "admin"
            },
            {
              "_metadata": {
                "uri": "https://<hostname>/odata/v2/User('sfadmin')",
                "type": "SFODData.User"
              },
              "userId": "sfadmin",
              "username": "sfadmin"
            }
          ]
        }
      }
    ]
  }
}
```

## Related Information

[\\$filter \[page 49\]](#)

[\\$expand \[page 55\]](#)

### 5.4.2.10.5 Query with \$top and \$skip

Using the \$top and \$skip options together in a request URI creates a pagination query. The response is a subset of the whole result, from \$skip to \$top. \$skip indicates the starting row for the query. \$top limits the size of the query.

For example, if the URI uses \$top=50, \$skip=20, the response is a subset from numbers 21 to number 70 of the whole result set.

### 5.4.2.10.6 Query with \$expand and \$orderby

When you use keywords \$expand and \$orderby in the same query, only the parent entities are sorted. Entities expanded through 1:n navigation properties aren't sorted. For example, in the following query tries to expand and sort the proxy users:

```
https://<hostname>/odata/v2/User?$format=JSON&$select=userId,username,proxy/
userId,proxy/username&$expand=proxy&$s=proxy/userId eq 'admin'&$orderby=proxy/
userId desc
```

Notice in the expanded results, the proxy users are not sorted in descending order by userId:

```
{
  "d": {
    "results": [
      {
        "_metadata": {
          "uri": "https://<hostname>/odata/v2/User('103004')",
          "type": "SFOData.User"
        },
        "userId": "103004",
        "username": "vwagner",
        "proxy": {
          "results": [
            {
              "_metadata": {
                "uri": "https://<hostname>/odata/v2/User('admin')",
                "type": "SFOData.User"
              },
              "userId": "admin",
              "username": "admin"
            },
            {
              "_metadata": {
                "uri": "https://<hostname>/odata/v2/User('sfadmin')",
                "type": "SFOData.User"
              },
              "userId": "sfadmin",
              "username": "sfadmin"
            }
          ]
        }
      }
    ]
  }
}
```

```
}  
}
```

## Related Information

[\\$orderby \[page 46\]](#)

[\\$expand \[page 55\]](#)

## 5.4.3 Pagination

The OData API provides several pagination options for query results.

A single OData HTTP `GET` request can return at most 1,000 records. This is reasonable because most HTTP clients have a timeout limit of 2 - 5 minutes. They do not allow transactions to wait more than that. A request running longer than the limit gets an HTTP timeout error. Therefore, it is often required to tune complex queries and lower the data size for them to complete within the timeout restriction.

Pagination limits the maximum size of a query response to 1,000 records. The OData standard provides a ' `__next` ' link in your query response if there are more results in the database.

SAP SuccessFactors HXM Suite OData API provides several pagination options for you to choose from. In this document, you'll find out how each pagination mechanism works, what are their advantages and disadvantages, and where to use which.

### ! Restriction

Pagination only works for entity sets. Function imports do not support pagination.

#### [Client-Side Pagination \[page 68\]](#)

Client-side pagination uses query options on the client side to create an offset that restricts the amount of data returned from the server.

#### [Server-Side Pagination \[page 70\]](#)

Unlike client-side pagination, where the pagination is controlled by client-specified parameters, server-side pagination is controlled on the server side.

#### [How Do I Choose? \[page 75\]](#)

#### [Custom Page Size \[page 76\]](#)

The default and maximum page size for a paginated query is 1000 items. You can customize the page size by setting a different number with parameter `customPageSize`.

**Parent topic:** [Query Operations \[page 43\]](#)

## Related Information

[Query with Key Predicate \[page 44\]](#)

[System Query Options \[page 45\]](#)

[Querying Effective-Dated Entities \[page 76\]](#)

### 5.4.3.1 Client-Side Pagination

Client-side pagination uses query options on the client side to create an offset that restricts the amount of data returned from the server.

A client-side pagination query can be made using the following URI parameters:

- The `$top` parameter indicates the number of records to return in the batch. The default and maximum number is 1,000 records.
- The `$skip` parameter indicates the number of records in the full data set to skip before fetching data.

For example, a query with `$skip=2000&$top=500` returns the fifth page of data where the page size is 500.

#### **i** Note

Client-side pagination doesn't guarantee that every page returns the exact number of records as you specified in the `$top` parameter because there might be dirty data and post-business rule validations might apply. This doesn't mean that the data is lost. The total number of records in the entire data set is not impacted.

Here's a summary of the advantages and disadvantages of client-side pagination:

Advantages	Disadvantages
<ul style="list-style-type: none"><li>• Faster initial page load</li><li>• It works well with SAP UI5 list and table controls to fetch records on demand. UI5 automatically changes the <code>\$skip</code> value in order to scroll through the data set.</li><li>• It allows reverse scrolling by decreasing <code>\$skip</code> values. This allows UI controls to avoid buffering all data as the user scrolls up and down.</li></ul>	<ul style="list-style-type: none"><li>• Query rerun for each page For each page, the query is re-executed and then moves forward through the data set to reach the specified <code>\$skip</code> value. Not only is this inefficient but also it can lead to unstable results. The data set could be changing due to simultaneous creating, updating, or deleting of the data.</li><li>• Potential duplicates If a record is inserted on a prior page after it has been read, it shifts the data set to the right, resulting the last record from the previous page to be pushed into the current page. This causes the page to be read again because the <code>\$skip</code> value of the current page is one record too small, resulting in duplicates in the total data set. You can avoid this issue using <code>\$orderBy=createdDateTime</code> to push newly created records to the end of the data set.</li><li>• Potential data loss If a record is deleted from a prior page after it has been read, it shifts the data set to the left. The first record of the current page is lost because the <code>\$skip</code> value of the current page is one record too large. There's no good work-around when dealing with concurrent hard deletes from the data set. Note these potential data loss issues make "last modified since" queries especially unreliable. If a record is lost in a query, it can only be picked up by a later attempt after it's modified.</li></ul>

Parent topic: [Pagination \[page 67\]](#)

## Related Information

[Server-Side Pagination \[page 70\]](#)

[How Do I Choose? \[page 75\]](#)

[Custom Page Size \[page 76\]](#)

## 5.4.3.2 Server-Side Pagination

Unlike client-side pagination, where the pagination is controlled by client-specified parameters, server-side pagination is controlled on the server side.

SAP SuccessFactors offer two types of server pagination: cursor-based pagination and snapshot-based pagination. You can specify the pagination method in the query by adding the parameter `&paging=cursor` or `&paging=snapshot`.

Both types of pagination return a `__next` parameter at the end of each query response that contains a `$skiptoken` value indicating the next page of data. Here are two examples of the `__next` link:

- Sample response in Atom format:

```
<link rel="next" href="https://<hostname>/odata/v2/User?
$skiptoken=eyJzdGFydFJvdYI6MTAwMCwiZW5kUm93IjoyMDAwfQ%3D%3D"></link>
```

### ⚠ Caution

Do not change the URL in the `__next` link. A modified URL can lead to unexpected results.

In the Atom format query responses, the ampersand "&" is automatically encoded as "%26". If you choose Atom as the format of your query response and the `__next` URL contains an ampersand, the link doesn't work because the server doesn't recognize "%26". You must decode the link before opening it. Alternatively, you can choose JSON as the response format.

- Sample response in JSON format:

```
"__next" : "https://<hostname>/odata/v2/User?$format=JSON&
$skiptoken=eyJzdGFydFJvdYI6MTAwMSwiZW5kUm93IjoyMDAwfQ%3D%3D"
```

[Cursor-based Pagination \[page 70\]](#)

[Snapshot-Based Pagination \[page 71\]](#)

Snapshot-based pagination is a server-side pagination that works by keeping a list of all business keys of the data set on the server.

Parent topic: [Pagination \[page 67\]](#)

## Related Information

[Client-Side Pagination \[page 68\]](#)

[How Do I Choose? \[page 75\]](#)

[Custom Page Size \[page 76\]](#)

### 5.4.3.2.1 Cursor-based Pagination

Cursor-based pagination maintains a database "cursor" on the server throughout pagination HTTP requests. The cursor represents a pointer to the start of the next page in the full data set. This is enabled using the

`&paging=cursor` parameter. The server uses a `$skiptoken` value to continue pagination at the data set location indicated by the cursor. The query is processed by the database for the first request. Requests for subsequent pages reuse the query by accessing data using the cursor.

### i Note

Currently, cursor-based pagination is only supported in Employee Central EmpJob entity, User entity, MDF Generic Objects, and most EC Foundation Objects.

Below is a summary of the advantages and disadvantages of cursor-based pagination:

Advantages	Disadvantages
<ul style="list-style-type: none"><li>Cursor-based pagination is faster than client offset pagination because the database query is only executed when the first page is requested. Subsequent pages are fetched from the location indicated by the <code>\$skiptoken</code> value.</li><li>It will not suffer data loss and duplication issues exhibited by client offset pagination.</li></ul>	<ul style="list-style-type: none"><li>Sorting is not supported. The <code>\$orderby</code> parameter cannot be used. An error will occur if it is attempted.</li><li>Filter is not supported for MDF-based entities. This includes Position, custom Generic Objects, and entities for MDF-based features such as Time, Benefits, CRM, etc. It can only be used for full extracts and does not support "last modified since" delta queries.</li><li>It does not support going backwards in the data set, making it not the ideal choice for UI consumption where UI5 control binding may require scrolling backwards for data not buffered in the control.</li></ul>

Parent topic: [Server-Side Pagination \[page 70\]](#)

## Related Information

[Snapshot-Based Pagination \[page 71\]](#)

### 5.4.3.2.2 Snapshot-Based Pagination

Snapshot-based pagination is a server-side pagination that works by keeping a list of all business keys of the data set on the server.

During the first page request, a "snapshot" or list of all the business keys is created after executing the query on the base entity. Query option `$filter` is processed to filter the full data set and `$orderby` is executed to sort the full data set. Subsequent pages are fetched by selecting the set of business keys and fetching each record based on these keys.

You can enable snapshot-based pagination by adding parameter `&paging=snapshot` in a query. Here is an example:

```
/odata/v2/PerPerson?paging=snapshot&$filter=lastModifiedDateTime ge
datetime'2001-07-01T17:19:28'&$orderby=personId
```

Below is a summary of the advantages and disadvantages of snapshot-based pagination:

Advantages	Disadvantages
<ul style="list-style-type: none"><li>• Snapshot-based pagination can improve the performance of complex queries with large data volumes. For simple queries, consider other pagination options.</li><li>• It doesn't suffer data loss and duplication issues exhibited by client offset pagination.</li><li>• It fully supports filtering with <code>\$filter</code> and sorting with <code>\$orderby</code>.</li><li>• It supports "last modified since" queries for all entity types.</li></ul>	<ul style="list-style-type: none"><li>• Potential first-page timeouts for complex queries. When you execute a query, a snapshot is created during the first page by executing the full query. Therefore, the request of the first page takes longer than subsequent pages. It can lead to potential timeout errors for large data sets with complex filtering, sorting, and expanding options.</li><li>• Snapshot-based pagination isn't the ideal choice for UI consumption because it doesn't support going backwards in the data set. UI control binding may require scrolling backwards for data not buffered in the control.</li><li>• Snapshot-based pagination is inefficient for UI consumption. The snapshot is created for the entire data set. If you only view the first few pages, most of the snapshot is wasted.</li></ul>

## Important Notes

Before you use snapshot-based pagination in your query, read the following important notes:

- Not all entities support snapshot-based pagination. If you attempt snapshot-based pagination on an entity that doesn't support the feature, the server automatically forces client offset pagination on the query. You can find out which type of pagination is used in the following custom headers of the query response:
  - If snapshot-based pagination is successful: `X-SF-Paging: snapshot`
  - If client offset pagination is successful as a fallback: `X-SF-Paging: offset`

### i Note

The custom header information is only available for successful snapshot-based pagination and forced client offset pagination when snapshot-based pagination isn't supported.

- Enable session reuse if you choose snapshot-based pagination. It helps avoid potential errors caused by switching servers during load balancing. For more information, see [Enabling Session Reuse](#).
- Querying entities that have multiple business keys can be slow if they aren't optimized for snapshot-based pagination.
- Do not use `&paging=snapshot` together with `$top` or `$skip`.
- A snapshot is deleted after 15 minutes if the next page isn't accessed. In rare cases, a snapshot may be lost on the cache server before the 15-minute limit is reached. In either situation, you need to request the entire snapshot again.



## Snapshot-Based Pagination in MDF OData API

We've optimized all MDF entity types for snapshot-based pagination. To be able to run snapshot-based pagination queries on MDF entities, you must have the [Admin access to MDF OData API](#) permission under [Metadata Framework](#) in RBP.

Although the snapshot-based pagination in MDF OData API is built based on the OData framework, it has its own specific logic and behaviors:

- Filter (`$filter`) and sort (`$orderby`) can only be applied to the first query. Queries to fetch subsequent pages can't be filtered or sorted.
- You can use multiple filter conditions on `lastModifiedDateTime` of the base entity and expanded entities. This applies to all types of navigation in MDF OData entities (excluding MDF Foundation Objects), such as picklist fields and entities with [Valid When](#) and [Composite](#) associations. For example:

```
/odata/v2/Position?paging=snapshot&$filter=parentPosition/lastModifiedDateTime le datetime'2001-07-01T17:19:28' or companyNav/lastModifiedDateTime ge datetime'2001-07-01T17:19:28'
```

## Snapshot-Based Pagination in Employee Central OData API

As of 1H 2020 Release, the following Employee Central entities have been optimized for snapshot-based pagination:

- PerPersonal
- EmpJob
- EmpEmployment
- FOLocation
- FOPayGrade
- MDF Foundation Objects

## Service Limits for Snapshot-Based Pagination

The following table summarizes the limits we currently have for snapshot-based pagination:

Operation	Default Limit
Maximum number of records expanded in a single query	300,000
Maximum number of consecutive "bad requests" within 30 minutes	10 (Company user level) 5 (Entity level)

### Expand Limit

The expand limit is intended to prevent excessive resource consumption and ensure better availability and performance for all customers. The default limit for the number of expanded records returned in a snapshot-based pagination query is 300,000. If this limit is reached, an error occurs.

### ❖ Example

This example query tries to expand a list of users' manager's direct reports:

```
/odata/v2/User?paging=snapshot&$filter=status eq 'T'&$expand=manager/  
directReports&$select=manager/directReports/userId
```

Under the expand limit, the total number of managers or direct reports expanded by **\$expand=manager/directReports** must not exceed 300,000.

### i Note

If you wish to increase the size limit, contact Product Support.

## Number of Consecutive "Bad Requests"

Snapshot-based pagination is intended for data sync and integration purposes where you pull data from the first page to the last. However, it often occurs that a user repeatedly requests for the first page of a paginated dataset and never reaches the subsequent pages. These requests are considered as bad requests because they waste server resources and could lead to performance issues.

To protect servers against intended or unintended bad requests, we've introduced a protection mechanism to regulate the usage of snapshot-based pagination. A quota is imposed on the company user level and the entity level to restrict the number of bad requests that can be made within a given period of time:

- The company user-level limit counts the number of consecutive bad requests made on all entities.
- The entity level limit counts the number of consecutive bad requests made on a single entity.

If the limit is reached, the user is blocked for 30 minutes during which the server rejects all subsequent requests and returns an error message in response.

The following table summarizes how this quota works:

Quota	Number of consecutive bad requests allowed within 30 minutes	Block time if the limit is reached
Company user level	10	30 minutes
Entity level	5	30 minutes

### i Note

- Multiple bad requests on a single entity also count as one bad request on the company user level.
- Before the limit is reached, using `$skiptoken` in the `__next` parameter to request the subsequent pages resets the count.
- If a user is blocked on the entity level, they still can access other entities.
- If a user is blocked on the company user level, they can't access any entities in this company.

### ❖ Example

You have made three bad requests on three different entities a, b, and c, and another two bad requests on entity d, the counts of consecutive bad requests are:

- Company user level: 4

- Entity level:
  - Entity a: 1
  - Entity b: 1
  - Entity c: 1
  - Entity d: 2

If you request for the second page of entity d now, the counts change to:

- Company user level: 0
- Entity level:
  - Entity a: 1
  - Entity b: 1
  - Entity c: 1
  - Entity d: 0

Parent topic: [Server-Side Pagination \[page 70\]](#)

## Related Information

[Cursor-based Pagination \[page 70\]](#)

### 5.4.3.3 How Do I Choose?

As a developer, you can choose among the different types of pagination based on their advantages and disadvantages. Here are our recommendations:

- Always use client offset pagination for UI consumptions. This is the only supported method on UI.
- Cursor-based pagination and snapshot-based pagination are recommended for integration use cases.
- Only use snapshot-based pagination for complex queries with large data volume, for example, a query that has complex sorting, filtering, and expanding options.
- We recommend that you tune your queries when you use snapshot-based pagination. If you have problems with first page performance for extremely large data sets, switch to cursor-based pagination if applicable.

Parent topic: [Pagination \[page 67\]](#)

## Related Information

[Client-Side Pagination \[page 68\]](#)

[Server-Side Pagination \[page 70\]](#)

[Custom Page Size \[page 76\]](#)

## 5.4.3.4 Custom Page Size

The default and maximum page size for a paginated query is 1000 items. You can customize the page size by setting a different number with parameter `customPageSize`.

You can use `customPageSize` for both client and server paginations. Here are two examples:

Setting the page size to 50 in a client pagination query:

```
https://<hostname>/odata/v2/Photo?$format=JSON&$top=500&$skip=200&customPageSize=50
```

Setting the page size to 50 in a snapshot-based server pagination query:

```
https://<hostname>/odata/v2/Photo?$format=JSON&paging=snapshot&customPageSize=50
```

### i Note

The custom page size must be smaller than 1000. Page size larger than 1000 is ignored.

Parent topic: [Pagination \[page 67\]](#)

## Related Information

[Client-Side Pagination \[page 68\]](#)

[Server-Side Pagination \[page 70\]](#)

[How Do I Choose? \[page 75\]](#)

## 5.4.4 Querying Effective-Dated Entities

You can use the following additional keywords for effective-dated entities (such as most Employee central entities and those MDF entities that are created effective-dated) :

Keyword	API Call
asOfDate	<code>https://&lt;API-Server&gt;/odata/v2/EmpJob&amp;\$format=json&amp;asOfDate=2014-01-01</code>
fromDate/toDate	<code>https://&lt;API-Server&gt;/odata/v2/EmpJob&amp;\$format=json&amp;fromDate=2014-01-01&amp;toDate=2014-12-31</code>

## How effective-dating is handled in an OData API query

The following rules are followed when an effective dated entity is queried:

1. If both the base entity and the navigation entity are effective-dated, then when expanding the navigation entity, the effectiveStartDate of the base entity is used as the asOfDate of the navigation entity.
2. If the base entity is none effective-dated, TODAY is used as asOfDate when expanding the navigation entity.
3. For a Picklist field, the navigation entity is PicklistValue. When expanding a Picklist field, the effectiveStartDate/ effectiveEndDate of the Picklist entity are used because PicklistValue itself is not effective-dated.
4. If you specify asOfDate as a query option, all entities in all levels of the query use this date as asOfDate.
5. If you specify fromDate and toDate as query options, then only the top level entity is filtered by the fromDate and toDate, all lower level entities follow rules 1, 2, and 3.
6. You can use either AsOfDate or fromDate/endDate in a query request, not both.

**Parent topic:** [Query Operations \[page 43\]](#)

## Related Information

[Query with Key Predicate \[page 44\]](#)

[System Query Options \[page 45\]](#)

[Pagination \[page 67\]](#)

## 5.5 Edit Operations

Learn how edit operations work in SAP SuccessFactors HXM Suite OData API.

Edit operations include insert, update (replace/merge), upsert (insert/update), and delete.

[Insert \[page 78\]](#)

You can use the insert operation to create an entry.

[Merge \[page 79\]](#)

[Replace \[page 80\]](#)

[Upsert \[page 81\]](#)

[Delete \[page 90\]](#)

[Edit Parameters \[page 91\]](#)

A list of parameters and examples for edit operations.

## 5.5.1 Insert

You can use the insert operation to create an entry.

The insert operation uses HTTP method `POST` to create an entry. In an insert request, you must specify the name of the API in the URI.

Here is an example of an insert request and response.

Request:

Operation	Insert
HTTP Request	POST
URI	https://<API-Server>/odata/v2/User
Payload	<pre>{   "__metadata": {     "uri": "User('paulchris')"   },   "username": "paulchris",   "password": "pwd",   "hireDate": "/Date(978307200000)/",   "gender": "M",   "status": "active",   "userId": "paulchris",   "firstName": "Paul",   "lastName": "Chris",   "email": "paulchris@bestrun.com",   "timeZone": "PST",   "manager": {     "__metadata": {       "uri": "User('kevinsmith')"     },     "username": "kevinsmith",     "password": "pwd",     "hireDate": "/Date(978307200000)/",     "gender": "M",     "status": "active",     "userId": "kevinsmith",     "firstName": "Kevin",     "lastName": "Smith",     "email": "kevinsmith@bestrun.com",     "department": "Retail Banking",     "timeZone": "PST"   } }</pre>

Response:

```
{
  "d": {
    "__metadata": {
      "uri": "https://<api-server>/odata/v2/User('paulchris')",
      "type": "SFOData.User"
    },
    "username": "paulchris",
    "password": "pwd",
    "hireDate": "/Date(978307200000)/",
    "gender": "M",
```

```

    "status": "active",
    "userId": "paulchris",
    "firstName": "Paul",
    "lastName": "Chris",
    "email": "paulchris@bestrun.com",
    "timeZone": "PST",
    "manager": {
      "_metadata": {
        "uri": "https://<api-server>/odata/v2/User('kevinsmith')",
        "type": "SFODData.User"
      },
      "username": "kevinsmith",
      "password": "pwd",
      "hireDate": "/Date(978307200000)/",
      "gender": "M",
      "status": "active",
      "userId": "kevinsmith",
      "firstName": "Kevin",
      "lastName": "Smith",
      "email": "kevinsmith@bestrun.com",
      "department": "Retail Banking",
      "timeZone": "PST"
    }
  }
}

```

To learn more about OData operations, visit [www.odata.org](http://www.odata.org), and search for "OData Version 2.0 Operations".

**Parent topic:** [Edit Operations \[page 77\]](#)

## Related Information

[Merge \[page 79\]](#)

[Replace \[page 80\]](#)

[Upsert \[page 81\]](#)

[Delete \[page 90\]](#)

[Edit Parameters \[page 91\]](#)

## 5.5.2 Merge

In certain cases, you might want to do an incremental update without replacing all the content of a data entity. To avoid overloading the `PUT` HTTP method, OData offers the merge operation to update a record. A merge request updates only the properties provided in the request body, and leaves the data not mentioned in the request body in its current state.

A merge operation request is delivered using the `POST` HTTP method with special header `x-http-method`, as shown in the example below. In the request payload, you only include the fields you want to update.

HTTP Method

`POST`

URI	<code>https://&lt;API-Server&gt;/odata/v2/ User('ExistingUser')</code>
Headers	<code>x-http-method: MERGE</code>
Payload	<pre>{   "email": "newmail@bestrun.com" }</pre>

A successful operation returns status 200 OK with no payload.

For merging data that includes navigation properties, see the examples in [Updating Links \[page 101\]](#).

**Parent topic:** [Edit Operations \[page 77\]](#)

## Related Information

[Insert \[page 78\]](#)

[Replace \[page 80\]](#)

[Upsert \[page 81\]](#)

[Delete \[page 90\]](#)

[Edit Parameters \[page 91\]](#)

## 5.5.3 Replace

The replace operation updates an existing record with the fields specified in the request payload and resets all other fields to their default value. In a replace request, you use the `PUT` HTTP method and identify the record to be updated by passing the key predicate in the request URI. Unlike the merge operation, you must provide all required fields in the request payload.

The replace request returns status 204 (No Content) to indicate success. No response body is returned.

The following table shows a sample replace request:

Operation	Replace
HTTP Method	<code>PUT</code>
URI	<code>https://&lt;API-Server&gt;/odata/v2/ User('ExistUser')</code>



Payload

```
{
  "__metadata": {
    "uri": "User('ExistUser')"
  },
  "username": "ExistUser",
  "password": "pwd",
  "hireDate": "/Date(978307200000)/",
  "gender": "M",
  "status": "active",
  "userId": "ExistUser",
  "firstName": "Paul",
  "lastName": "Chris",
  "email": "user@bestrun.com",
  "department": "Retail Banking",
  "timeZone": "PST",
  "hr": {
    "__metadata": {
      "uri": "User('HRUser')"
    }
  },
  "manager": {
    "__metadata": {
      "uri": "User('OldManager')"
    }
  }
}
```

Parent topic: [Edit Operations \[page 77\]](#)

## Related Information

[Insert \[page 78\]](#)

[Merge \[page 79\]](#)

[Upsert \[page 81\]](#)

[Delete \[page 90\]](#)

[Edit Parameters \[page 91\]](#)

## 5.5.4 Upsert

The `upsert` operation is a custom function import to update or insert records. If the record specified in payload doesn't exist, the server inserts (creates) a new record; if a record exists, the upsert operation updates the record. You can use upsert to batch insert/update and inline-edit records. You can invoke the upsert operation using the HTTP method `POST`.

### → Remember

Any invalid property value leads to the failure of upserting a record.

The failure in upserting one record doesn't affect other records.

If business keys are provided in the request, they also appear in failed response.

For effective-dated entities, the failure of upserting one time slice causes all other time slices of the same external code to fail. See example 2 for more information.

[Responses and HTTP Codes \[page 82\]](#)

[Example 1: Create and Update Users in One Upsert Request \[page 84\]](#)

[Example 2: Upsert Multiple Records of an Effective Dated Entity \[page 87\]](#)

[Example 3: Get a Faster Upsert Response by Specifying Entity Names in URI \[page 89\]](#)

**Parent topic:** [Edit Operations \[page 77\]](#)

## Related Information

[Insert \[page 78\]](#)

[Merge \[page 79\]](#)

[Replace \[page 80\]](#)

[Delete \[page 90\]](#)

[Edit Parameters \[page 91\]](#)

### 5.5.4.1 Responses and HTTP Codes

Depending on the request, the upsert operation returns the following values in the `httpCode` field of the response:

HTTP Code	Result	Sample Code
200	Successful.	<pre>{   "key": "newuser",   "status": "OK",   "editStatus":     "UPSERTED",   "message": null,   "index": 0,   "httpCode": 200,   "inlineResults":     null }</pre>

HTTP Code	Result	Sample Code
201	Successful. New record inserted.	<pre> {     "key": "newuser",     "status": "OK",     "editStatus": "INSERTED",     "message": null,     "index": 0,     "httpCode": 201,     "inlineResults": null }</pre>
204	Successful. An existing record has been updated.	<div> <div>Sample Code</div> <pre> {     "key": "cgrant",     "status": "OK",     "editStatus": "UPDATED",     "message": null,     "index": 1,     "httpCode": 204,      "inlineResults": null }</pre> </div>
400	Failed. The request couldn't be understood by the server due to malformed syntax. Check the message field for detailed error information.	<pre> {     "key": null,     "status": "ERROR",     "editStatus": null,     "message": "bad valueString ['AnotherManager'] as part of keyString ['AnotherManager']",     "index": 1,     "httpCode": 400,     "inlineResults": null }</pre>
500	Failed. The request couldn't be processed due to unexpected server error.	<pre> {     "key": null,     "status": "ERROR",     "editStatus": null,     "message": "Internal server error",     "index": 0,     "httpCode": 500,     "inlineResults": null }</pre>

Parent topic: [Upsert \[page 81\]](#)

## Related Information

[Example 1: Create and Update Users in One Upsert Request \[page 84\]](#)

[Example 2: Upsert Multiple Records of an Effective Dated Entity \[page 87\]](#)

[Example 3: Get a Faster Upsert Response by Specifying Entity Names in URI \[page 89\]](#)

[Errors \[page 202\]](#)

### 5.5.4.2 Example 1: Create and Update Users in One Upsert Request

The following code sample shows an upsert request that creates a new user, updates an existing user, and updates the manager of the existing user inline.

#### Request

Operation	Upsert
HTTP Method	POST
URI	<code>https://&lt;API-Server&gt;/odata/v2/upsert</code>

## Payload

```
[
  {
    "__metadata": {
      "uri": "User('NewUser')"
    },
    "username": "NewUser",
    "password": "pwd",
    "hireDate": "/"
Date(978307200000)/",
    "gender": "M",
    "status": "active",
    "userId": "NewUser",
    "firstName": "Paul",
    "lastName": "Chris",
    "email": "user@bestrun.com",
    "department": "Retail Banking",
    "timeZone": "PST",
    "hr": {
      "__metadata": {
        "uri": "User('HRUser')"
      }
    },
    "manager": {
      "__metadata": {
        "uri":
"User('OldManager')"
      }
    }
  },
  {
    "__metadata": {
      "uri": "User('ExistUser')"
    },
    "username": "ExistUser",
    "password": "pwd",
    "hireDate": "/"
Date(978307200000)/",
    "gender": "M",
    "status": "active",
    "userId": "ExistUser",
    "firstName": "Chris",
    "lastName": "Paul",
    "email": "user@bestrun.com",
    "department": "Retail Banking",
    "timeZone": "PST",
    "hr": {
      "__metadata": {
        "uri": "User('HRUser')"
      }
    },
    "manager": {
      "__metadata": {
        "uri":
"User('OldManager')"
      },
      "username": "OldManager",
      "password": "pwd",
      "hireDate": "/"
Date(978307200000)/",
      "gender": "F",
      "status": "active",
      "userId": "OldManager",
      "firstName": "Paul",
      "lastName": "Chris",
      "email": "user@bestrun.com",
```

```

        "department": "Retail
Banking",
        "timeZone": "PST",
        "hr": {
            "__metadata": {
                "uri":
"User('HRUser')"
            }
        },
        "manager": {
            "__metadata": {
                "uri":
"User('AnotherManager')"
            }
        }
    }
}
]

```

## Response

The operation returns a response body indicating all upsert results:

```

{
  "d": [
    {
      "key": "NewUser",
      "status": "OK",
      "editStatus": "INSERTED",
      "message": null,
      "index": "0",
      "httpCode": 201,
      "inlineResults": null
    },
    {
      "key": "ExistUser",
      "status": "OK",
      "editStatus": "UPDATED",
      "message": null,
      "index": "1",
      "inlineResults": [
        {
          "results": [
            {
              "key": "OldManager",
              "status": "OK",
              "editStatus": "UPDATED",
              "message": null,
              "index": "0",
              "httpCode": 204,
              "inlineResults": null
            }
          ],
          "inlineProperty": "manager"
        }
      ]
    }
  ]
}

```

Parent topic: [Upsert \[page 81\]](#)

## Related Information

[Responses and HTTP Codes \[page 82\]](#)

[Example 2: Upsert Multiple Records of an Effective Dated Entity \[page 87\]](#)

[Example 3: Get a Faster Upsert Response by Specifying Entity Names in URI \[page 89\]](#)

### 5.5.4.3 Example 2: Upsert Multiple Records of an Effective Dated Entity

You can create historical information of an effective-dated record by upserting multiple time slices of the same external code in one request. Note that if one of these time slices fails, all other time slices of the same external code fail.

## Request

In this example, the date format in the first time slice is incorrect.

Operation	Upsert
HTTP Method	POST
URI	<code>https://&lt;API-Server&gt;/odata/v2/upsert</code>

## Payload

```
[{
  "__metadata":{
    "uri":"cust_TestObject(effectiveStartDate=datetime'2019-06-01T00:00:00',externalCode='SO-004') "
  },
  "externalCode":"SO-005",
  "effectiveStartDate":"/Date(1559347ABC)/",
  "cust_Category":"Cat A"
},
{
  "__metadata":{
    "uri":"cust_TestObject(effectiveStartDate=datetime'2019-07-01T00:00:00',externalCode='SO-004') "
  },
  "externalCode":"SO-005",
  "effectiveStartDate":"/Date(1561939200000) /",
  "cust_Category":"Cat B"
},
{
  "__metadata":{
    "uri":"cust_TestObject(effectiveStartDate=datetime'2019-08-01T00:00:00',externalCode='SO-004') "
  },
  "externalCode":"SO-005",
  "effectiveStartDate":"/Date(1564617600000) /",
  "cust_Category":"Cat C"
}]
```

## Response

The response shows ERROR status for the first time slice and ACCEPTED for the other two. All three time slices have failed.

```
{
  "d": [
    {
      "key": null,
      "status": "ERROR",
      "editStatus": null,
      "message": "Illegal DateTime value in JSON format: /Date(1559347ABC)/. Please follow the standard format: \" /Date(<ticks>) /\", <ticks> = number of milliseconds since midnight Jan 1, 1970. Example: \" /Date(1495746637000) /\". For more information, see https://help.sap.com/viewer/d599f15995d348a1b45ba5603e2aba9b/latest/en-US/971f5829b7aa4c9fab04c12f06838683.html.",
      "index": 0,
      "httpCode": 400,
      "inlineResults": null
    },
    {
      "key": "cust_TestObject/effectiveStartDate=2019-07-01T00:00:00.000Z,cust_TestObject/externalCode=SO-004",
      "status": "ACCEPTED",
      "editStatus": null,
      "message": null,
      "index": 1,
      "httpCode": 202,
      "inlineResults": null
    }
  ]
}
```



```

    },
    {
      "key": "cust_TestObject/
effectiveStartDate=2019-08-01T00:00:00.000Z,cust_TestObject/externalCode=SO-004",
      "status": "ACCEPTED",
      "editStatus": null,
      "message": null,
      "index": 2,
      "httpCode": 202,
      "inlineResults": null
    }
  ]
}

```

Parent topic: [Upsert \[page 81\]](#)

## Related Information

[Responses and HTTP Codes \[page 82\]](#)

[Example 1: Create and Update Users in One Upsert Request \[page 84\]](#)

[Example 3: Get a Faster Upsert Response by Specifying Entity Names in URI \[page 89\]](#)

### 5.5.4.4 Example 3: Get a Faster Upsert Response by Specifying Entity Names in URI

When you start to make API calls, your first upsert operation tries to download the full set of OData metadata from the server and stores it in cache. Subsequent API calls use the cached metadata until it's refreshed. Due to its large size, downloading the full metadata can be time consuming. To get a faster response, you can explicitly include the entities to be upserted in the URI so that only the metadata of these entities are downloaded.

To do this, you include the entity names in the URI, separated by comma.

#### **i** Note

If you want to inline edit an entity, you also need to include it in the URI.

This example shows how to create an activity for a user. In the request URI, only the metadata of the necessary entities are specified.

Request URI:

```
https://<API-Server>/odata/v2/User,Activity,ActivityStatus/upsert
```

Request Payload:

```

{
  "_metadata": {
    "uri": "Activity"
  },
  "activityName": "Test Activity",
  "activityStatusNav": {

```

```

      "_metadata":{
        "uri":"ActivityStatus"
      },
      "activityStatusId":"high",
      "colorRGBCode":"#a71927"
    },
    "subjectUserIdNav":{
      "_metadata":{
        "uri":"User('cgrant') "
      }
    }
  }
}

```

Parent topic: [Upsert \[page 81\]](#)

## Related Information

[Responses and HTTP Codes \[page 82\]](#)

[Example 1: Create and Update Users in One Upsert Request \[page 84\]](#)

[Example 2: Upsert Multiple Records of an Effective Dated Entity \[page 87\]](#)

## 5.5.5 Delete

The delete operation deletes an entry. In a delete request, you use the `DELETE` HTTP method and specify the entry to be deleted by passing the key predicate in the request URI.

### Note

There are two types of delete operations: hard delete and soft delete. A hard delete permanently removes an entry from the database while a soft delete marks the entry as deleted. Soft deleted entries can be accessed with certain tools, for example, an audit report.

In Employee Central, soft deleted entries can even be queried if the instance is configured as such. For more information, see [Expanded Entities: Handling Deleted Objects](#).

Here's an example of a delete operation:

Operation	Delete
HTTP Method	<code>DELETE</code>
URI	<code>https://&lt;API-Server&gt;/odata/v2/ TodoEntryV2(18234M)</code>

A successful delete operation returns status 200 OK with no response payload.

Parent topic: [Edit Operations \[page 77\]](#)

## Related Information

[Insert \[page 78\]](#)

[Merge \[page 79\]](#)

[Replace \[page 80\]](#)

[Upsert \[page 81\]](#)

[Edit Parameters \[page 91\]](#)

## 5.5.6 Edit Parameters

A list of parameters and examples for edit operations.

The parameters in this section apply to all SAP SuccessFactors HXM Suite OData API entities unless otherwise specified in the topic. For a list of MDF-specific parameters, see [Query Parameters \[page 137\]](#) and [Insert and Upsert Parameters](#).

**Parent topic:** [Edit Operations \[page 77\]](#)

## Related Information

[Insert \[page 78\]](#)

[Merge \[page 79\]](#)

[Replace \[page 80\]](#)

[Upsert \[page 81\]](#)

[Delete \[page 90\]](#)

## 5.5.6.1 apiOptionProfileID

The `apiOptionProfileID` parameter allows you to specify an API option profile in your API call. With an API option profile, additional processing parameters can be specified to control extra business logic when you perform edit operations with the User entity.

### Parameter Values

Value	Description
<code>User defined value</code>	The parameter uses the API option profile ID that you define in the <i>Manage API Option Profile</i> tool in <i>API Center</i> . For more information, see <a href="#">Creating and Editing Users Using an API Option Profile [page 189]</a> .

### Use Case

For more information about using the `apiOptionProfileID`, see [Creating and Editing Users Using an API Option Profile \[page 189\]](#).

## 5.5.6.2 processInactiveEmployees

`processInactiveEmployees` is an exclusive parameter for the User entity. You can use this parameter when edit an inactive user.

### Parameter Values

Value	Description
<code>false</code> (default)	You can edit only active users.
<code>true</code>	You can edit both active and inactive users.

### Use Cases

By default, you can't edit users in statuses `inactive (f)` and `inactive_external (F)`. To edit inactive users, you need to either use the `processInactiveEmployees` parameter or an API option profile that allows editing of inactive users.

For detailed use cases, see [Processing Inactive Users](#).

### 5.5.6.3 purgeType

`purgeType` is an optional parameter that determines whether an incremental or full update will be performed on an entity.

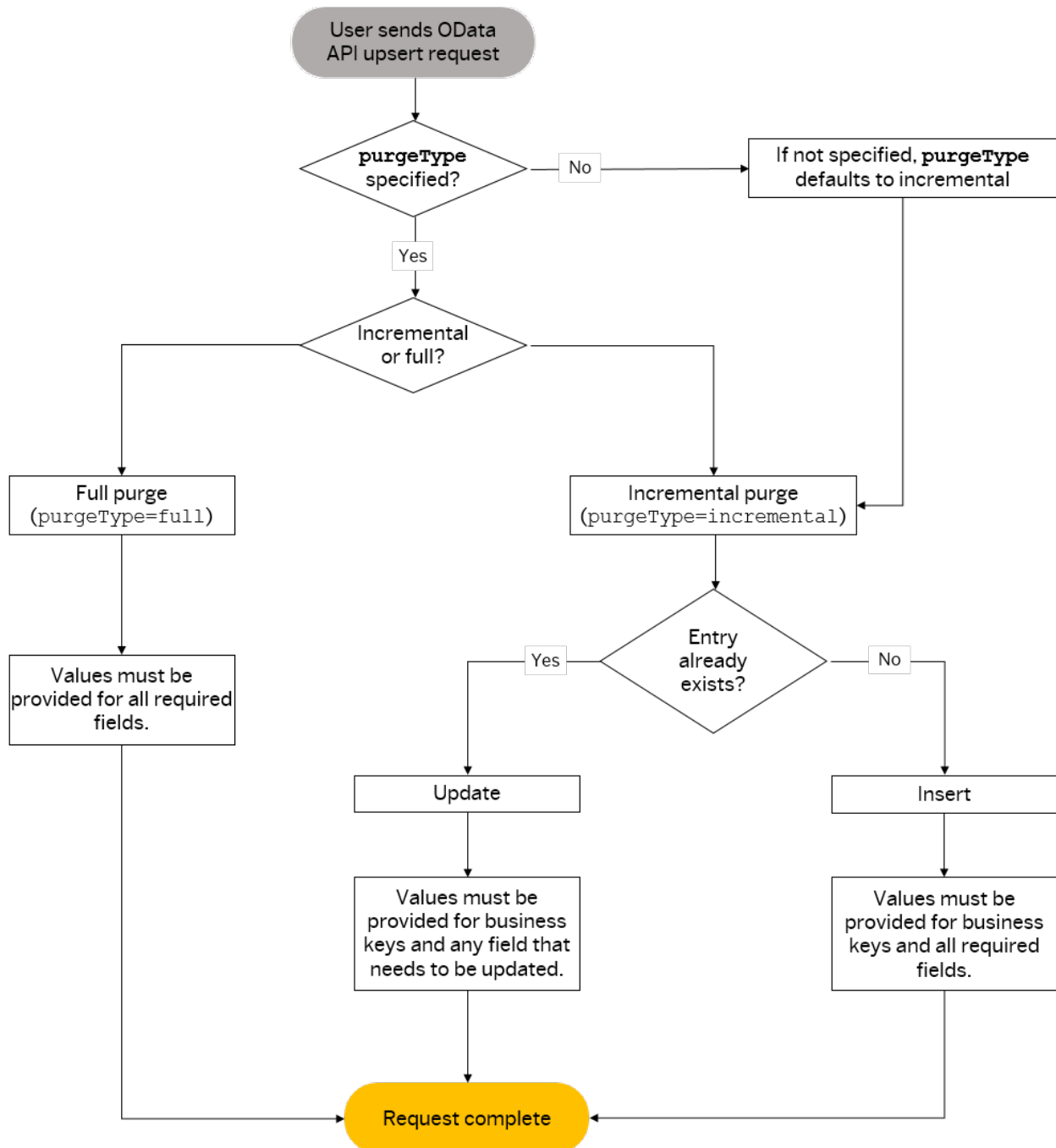
#### Parameter Values

Value	Description
<code>incremental</code> (default)	<p>The upsert operation purges and replaces only the data specified in the request payload.</p> <p>The default upsert behavior is incremental purge unless you specify <code>purgeType=full</code> in the request.</p>
<code>full</code> or <code>fullPurge</code>	<p>The upsert operation purges all existing data of the entry and creates new data specified in the request payload.</p> <div><p><b>⚠ Caution</b></p><p>If there are multiple time slices for the given record, the upsert operation with <code>purgeType=full</code> deletes all the time slices and creates new ones specified in the request. If you want to keep a particular time slice, make sure to include all of its data in the request payload. If some time slices are missing, then import engine will delete them.</p></div>
<code>record</code>	<p>Same behavior as incremental purge.</p> <p>This type of incremental purge is available only for MDF entities and the <code>User</code> entity.</p>

#### Mandatory Field Governance During Upsert

Since a full purge erases all existing data of an entry and creates new data, it follows the mandatory fields governance of the entity. Whether an entry exists or not in the database, you must provide all required fields in your

payload to be able to successfully upsert the entry. The following diagram helps you understand the purge behaviors of OData API:



## 5.5.6.4 strictTransactionIsolate

`strictTransactionIsolate` is an upsert parameter for determining if partial upsert applies during multiple-record upserts.

### Parameter Values

#### ⚠ Caution

The `strictTransactionIsolate` parameter is used to ensure data integrity during upsert. Using the parameter could result in performance degradation, the extent of which depends on the single record processing time and the batch size.

Value	Description
<code>false</code> (default)	Standard upsert behavior: one failed record during upsert causes all records to fail.
<code>true</code>	<ul style="list-style-type: none"><li>When you upsert multiple records, the upsert status of one record doesn't affect the others.</li><li>When you upsert multiple records in a <code>\$batch</code> request, if one record fails, the entire changeset rolls back.</li></ul>

### Use Case

The value of `strictTransactionIsolate` can be applied in the following use cases:

- Multiple record upserts
- Upserts of entities that don't support full purge

If you upsert a single record, this parameter is ignored.

## 5.5.6.5 enableUpsertResponseExtensionInChangeset

Use the `enableUpsertResponseExtensionInChangeset` parameter to control whether all upsert error messages are returned in a `ChangeSet`.

### Parameter Values

Value	Description
<code>false</code> (default)	When upsert errors occur in a batch <code>ChangeSet</code> , only the first error is returned.
<code>true</code>	When upsert errors occur in a batch <code>ChangeSet</code> , all errors are returned.

### Use Case

In this example, the batch request contains an upsert operation that tries to create two users. In the upsert payload, neither user ID matches the key, so the upsert operation fails.

#### Request Payload

```
--batch_202010071357
Content-Type: multipart/mixed; boundary=changeset_202010071357
Content-Length: 100
--changeset_202010071357
Content-Transfer-Encoding: binary
Content-Type: application/http
POST upsert HTTP/1.1
Content-Type: application/json; charset=utf-8
accept: application/json
[
  {
    "__metadata": {
      "uri": "User('user1')"
    },
    "username": "content1",
    "password": "content1",
    "hireDate": "/Date(978307200000)/",
    "status": "active",
    "userId": "content1",
    "firstName": "content1",
    "lastName": "example1@abc.com",
    "department": "test",
    "timeZone": "PST"
  },
  {
    "__metadata": {
      "uri": "User('user2')"
    },
    "username": "content2",
    "password": "content2",
    "hireDate": "/Date(978307200000)/",
```



```

    "status": "active",
    "userId": "content2",
    "firstName": "content2",
    "lastName": "example2@abc.com",
    "department": "test",
    "timeZone": "PST"
  }
]
--changeset_202010071357--

--batch_202010071357--

```

Compare the following two responses when the parameter is set to different values.

Example request through URI /odata/v2/\$batch or /odata/v2/\$batch?

enableUpsertResponseExtensionInChangeset=false:

```

--batch_a89da8c8-82e1-4171-ac21-e69e474666e1
Content-Type: application/http
Content-Transfer-Encoding: binary
HTTP/1.1 500 Internal Server Error
Content-Type: application/json; charset=utf-8
DataServiceVersion: 1.0
X-SF-Record-Count-Recursive: 0
Content-Length: 265
{
  "error": {
    "code": "COE_GENERAL_SERVER_FAILURE",
    "message": {
      "lang": "en-US",
      "value": "ChangeSet index 1 - Key property (User/userId) doesn't match
the key in the __metadata uri; Key property (User/userId) doesn't match the key in
the __metadata uri"
    }
  }
}
--batch_a89da8c8-82e1-4171-ac21-e69e474666e1--

```

Example request through URI /odata/v2/\$batch?enableUpsertResponseExtensionInChangeset=true:

```

--batch_fe845a0f-d7be-42dd-8cdb-bc4096abef2f
Content-Type: application/http
Content-Transfer-Encoding: binary
HTTP/1.1 500 Internal Server Error
Content-Type: application/json; charset=utf-8
DataServiceVersion: 1.0
X-SF-Record-Count-Recursive: 0
Content-Length: 446
{
  "error": {
    "code": "COE_GENERAL_SERVER_FAILURE",
    "message": {
      "lang": "en-US",
      "value": "ChangeSet index 1 - Upsert error"
    },
    "innererror": {
      "errordetails": [
        {
          "code": "User/userId=user1",
          "message": "Key property (User/userId) doesn't match the key in
the __metadata uri",
          "severity": "",
          "target": ""
        },
        {
          "code": "User/userId=user2",

```

```

    "message": "Key property (User/userId) doesn't match the key in
the __metadata uri",
    "severity": "",
    "target": ""
  }
]
}
}
}
--batch_fe845a0f-d7be-42dd-8cdb-bc4096abef2f--

```

## Related Information

[Batch Operations \[page 108\]](#)

## 5.6 Working with Links

You can use the `$links` system option to query, create, edit, and delete associations between OData entities.

The `$links` operation is a system-defined OData option that can be used to address associations between OData entities. An association is a relationship between one entity and another. The basic rules for addressing associations are shown in the following figure:



### Note

Key predicate of the navigation property can be required, optional, or empty depending on the type of request and association.

For example, the URI below can be used to query a users manager:

```
https://<API-Server>/odata/v2/User('cgrant')/$links/manager
```

- `https://<API-Server>/odata/v2/User('cgrant')` identifies user with user ID **cgrant**.
- The `manager` navigation property describes the relationship of this user

### [Querying Links \[page 99\]](#)

You can query links using the `$links` system option.

### [Creating Links \[page 100\]](#)

You can associate one entity to another by creating links.

### [Updating Links \[page 101\]](#)

You can update existing links by either replacing or merging them.

### [Deleting Links \[page 103\]](#)

You can remove entity associations by deleting the links.

### [Working with Links in Effective-Dated Entities \[page 104\]](#)

Special rules follow when you create and edit links for effective-dated entities, such as MDF entities.

## 5.6.1 Querying Links

You can query links using the `$links` system option.

To query the relationships between entities using `$links`:

```
https://<API-Server>/odata/v2/User('cgrant')/$links/matrixManager
```

The response returns the URI of the navigation property **manager**, not the entry itself:

```
<?xml version="1.0" encoding="utf-8"?>
<link xmlns="http://schemas.microsoft.com/ado/2007/08/dataservices">
  <uri>https://<API-Server>/odata/v2/User('manager1')</uri>
  <uri>https://<API-Server>/odata/v2/User('manager2')</uri>
</link>
```

You can also use `$format` to specify the response format:

```
https://<API-Server>/odata/v2/User('cgrant')/$links/matrixManager
```

Example response:

```
{
  "d": {
    "results": [
      {
        "uri": "https://<API-Server>/odata/v2/User('manager1') "
      },
      {
        "uri": "https://<API-Server>/odata/v2/User('manager2') "
      }
    ]
  }
}
```

Parent topic: [Working with Links \[page 98\]](#)

## Related Information

[Creating Links \[page 100\]](#)

[Updating Links \[page 101\]](#)

[Deleting Links \[page 103\]](#)

[Working with Links in Effective-Dated Entities \[page 104\]](#)

## 5.6.2 Creating Links

You can associate one entity to another by creating links.

You can create associations between entities by referencing one entity during the creation or modification of another, or by explicitly issuing a `POST` request against the URL of the `$links` option. The request must have the new URI in the request body following the appropriate format for stand-alone `$links` options in XML or JSON.

The example below adds a new matrix manager by creating a link between the user and the manager through navigation property `matrixManager`:

HTTP Method	POST
URI	<code>https://&lt;API-Server&gt;/odata/v2/ User('cgrant')/\$links/matrixManager</code>
Payload	<div>ATOM:<div><pre>&lt;uri xmlns="http://schemas.microsoft.com/ado/2007/08/dataservices"&gt; https://&lt;api-server&gt;/v2/User('EYtest02') &lt;/uri&gt;</pre></div><div>JSON:<div><pre>{   "uri": "User('manager2')" }</pre></div></div></div>

Different rules apply when you create links for MDF entities and non-MDF entities.

### Rules for Creating Links for Non-MDF Entities

The following rules apply when you create links between non-MDF entities:

- A new link is always inserted for a 1:n association. If the same link already exists, the `POST` operation returns `204 No Content` without data change.
- The server always performs a replace operation for a 1:1 relationship. For example, if you create a manager link for a user who already has a manager, the new manager replaces the old manager.

### Rules for Creating Links for MDF Entities

The following rules apply when you create links between MDF entities (including MDF Foundation Objects):

- You can't create links for MDF Generic Objects with composite type associations.
- A new link is always inserted for a 1:n association. If the same link already exists, the `POST` operation returns `204 No Content` without data change.

- For 1:1 relationship, if a link already exists, the POST operation fails.

Parent topic: [Working with Links \[page 98\]](#)

## Related Information

- [Querying Links \[page 99\]](#)
- [Updating Links \[page 101\]](#)
- [Deleting Links \[page 103\]](#)
- [Working with Links in Effective-Dated Entities \[page 104\]](#)

### 5.6.3 Updating Links

You can update existing links by either replacing or merging them.

You use HTTP method PUT to update links between entities.

#### Replacing Links

You can replace links for 1:1 or 1:n associations using the PUT HTTP method.

- For 1:n associations, you must provide the key in the URI and replace it with the new entry in the request payload:

HTTP Method	PUT
URI	<code>https://&lt;API-server&gt;/odata/v2/ User('cgrant')/\$links/ matrixManager('manager1')</code>
Payload	<pre>{   "uri": "User('manager3')" }</pre>

This request replaces **manager1** of user cgrant with **manager3**. Note that the user's other matrix manager links aren't impacted.

- For 1:1 associations, you don't provide any key in the URI. You provide only the new link in the request payload. This entry replaces the existing link.

HTTP Method	PUT
-------------	-----

URI	<code>https://&lt;API-server&gt;/odata/v2/ User('cgrant')/\$links/manager</code>
Payload	<pre>{   "uri": "User('manager1') " }</pre>

### **i Note**

For non-MDF entities, you can also use POST to replace a 1:1 link. It gets you the same result as PUT. For MDF entities, POST can only be used when there's no existing link.

## Merging Links

You can use POST requests to add a new link to a 1:n relation. When you add a new link, it's merged with the existing links.

The example below adds a new matrix manager **manager3** to user **cgrant**.

HTTP Method	POST
URI	<code>https://&lt;API-server&gt;/odata/v2/ User('cgrant')/\$links/matrixManager</code>
Payload	<pre>{   "uri": "User('manager3') " }</pre>

**Parent topic:** [Working with Links \[page 98\]](#)

## Related Information

[Querying Links \[page 99\]](#)

[Creating Links \[page 100\]](#)

[Deleting Links \[page 103\]](#)

[Working with Links in Effective-Dated Entities \[page 104\]](#)

## 5.6.4 Deleting Links

You can remove entity associations by deleting the links.

Using a `DELETE` request with the `$links` option, you can remove the association between two entries. The business key of the source and target entry must be provided in the URI. Upon success, the operation returns status `204 No Content`. No response body is returned.

For example, the following request deletes the link between user `cgrant` and its matrix manager:

HTTP Method	DELETE
URI	<code>https://&lt;API-Server&gt;/odata/v2/ User('cgrant')/\$links/ matrixManager('manager1')</code>

### Rules for Deleting Links of Non-MDF Entities

The following rules apply when you delete links of non-MDF entities:

- Business key is optional if you delete a link in a 1:1 association.
- If you want to delete a single link in a 1:n association, the key must be provided.
- If no key is provided in URI, all links are deleted.

### Rules for Deleting Links of MDF Entities (Including MDF Foundation Objects)

The following rules apply when you delete links of MDF entities, including MDF Foundation Objects:

- Regardless of the association type, a business key must be provided.
- You can't delete multiple links in one request.

**Parent topic:** [Working with Links \[page 98\]](#)

### Related Information

[Querying Links \[page 99\]](#)

[Creating Links \[page 100\]](#)

[Updating Links \[page 101\]](#)

[Working with Links in Effective-Dated Entities \[page 104\]](#)

## 5.6.5 Working with Links in Effective-Dated Entities

Special rules follow when you create and edit links for effective-dated entities, such as MDF entities.

### Creating Links Between Two Effective-Dated Records

You can use `$links` to create references:

- Between effective-dated records, such as *Valid When* associations, a Picklist field, and Generic Object (GO) field.
- From an effective-dated record to a noneffective-dated record with a *Valid When* association
- From a noneffective-dated record to an effective-dated record with a *Valid When* association

When you create a link between two effective-dated records using `$links`, the following rules apply:

- The `effectiveStartDate` of the source timeslice must be later than the `effectiveStartDate` of the earliest timeslice in the target record. Otherwise, an error occurs. (See Example 1)
- You can only create links for source timeslices that are subsets of the target record. The target timeslice doesn't necessarily have to match the source timeslice. The link is created between the source timeslice and matching target timeslices. (See examples 2 and 3)

#### Example 1

In this example, you try to link a source entry to a target whose earliest timeslice has an `effectiveStartDate` later than the source record. It leads to an error.

HTTP Method	POST
URI	<code>https://&lt;API-Server&gt;/odata/v2/ cust_SourceObject(effectiveStartDate=datetime'2019-06-01T00:00:00',externalCode='SO-001')/\$links/cust_TargetNav</code>
Payload	<pre>&lt;uri xmlns="http://schemas.microsoft.com/ado/2007/08/dataservices"&gt; https://&lt;API-Server&gt;/odata/v2/ cust_TargetObject(effectiveStartDate=datetime'2019-07-01T00:00:00',externalCode='TO-001') &lt;/uri&gt;</pre>





Source Record

Target Record

Link created

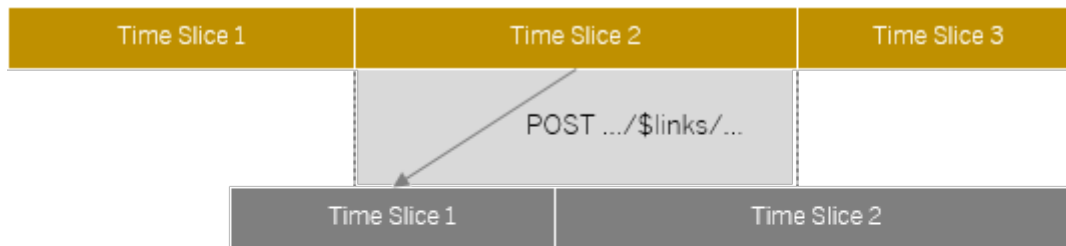
Sample response

```
{
  "error": {
    "code": "COE_GENERAL_BAD_REQUEST",
    "message": {
      "lang": "en-US",
      "value": "[COE0018]Invalid reference '(effectiveStartDate=2019-07-01, externalCode=TO-001)' for valid when association 'cust_TargetNav'."
    }
  }
}
```

## Example 2

For the same objects, if you choose a different timeslice in the source entry that has a later effectiveStartDate, the link is created.

HTTP Method	POST
URI	https://<API-Server>/odata/v2/ cust_SourceObject(effectiveStartDate=datetime'2019-07-01T00:00:00',externalCode='SO-001')/\$links/cust_TargetNav
Payload	<pre>&lt;uri xmlns="http://schemas.microsoft.com/ado/2007/08/dataservices"&gt; https://&lt;API-Server&gt;/odata/v2/ cust_TargetObject(effectiveStartDate=datetime'2019-06-15T00:00:00',externalCode='TO-001') &lt;/uri&gt;</pre>



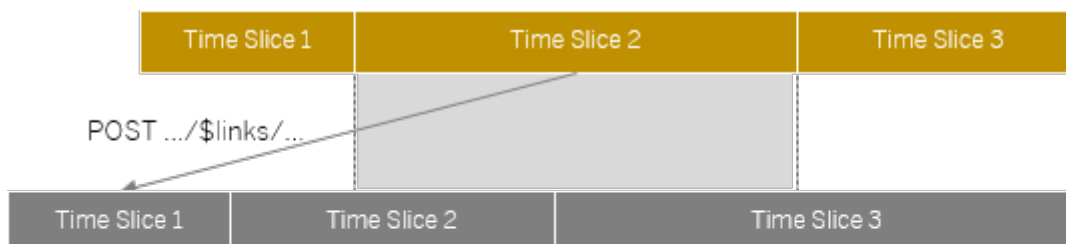
- Source Record
- Target Record
- Link created

A successful response returns status code [204](#) with no content.

### Example 3

If the source timeslice is a subset of the target record, you can create links to any timeslice in the target. The link is created between the source timeslice and the matching target timeslices.

HTTP Method	POST
URI	https://<API-Server>/odata/v2/ cust_SourceObject(effectiveStartDate=datetime'2019-07-01T00:00:00',externalCode='SO-001')/\$links/cust_TargetNav
Payload	<pre>&lt;uri xmlns="http://schemas.microsoft.com/ado/2007/08/dataservices"&gt; https://&lt;API-Server&gt;/odata/v2/ cust_TargetObject(effectiveStartDate=datetime'2019-05-01T00:00:00',externalCode='TO-001') &lt;/uri&gt;</pre>



- Source Record
- Target Record
- Link created

## Deleting a Link

You can delete links between effective-dated records with `$link`. To delete a link, specify the target key predicate in the URI:

HTTP Method	DELETE
URI	<code>https://&lt;API-Server&gt;/odata/v2/ cust_SourceObject(effectiveStartDate=datetime'<b>2019-07-01T00:00:00</b>',externalCode='SO-001')/\$links/ cust_TargetNav(effectiveStartDate=datetime'<b>2019-05-01T00:00:00</b>',externalCode='TO-001')</code>

Note that you're not allowed to delete a link if:

- The field is required
- It's a required 1:1 association.
- The link you're trying to delete is the last existing link in a required 1:many association.

If you try to delete such a link, an error occurs:

```
{
  "error": {
    "code": "COE_GENERAL_BAD_REQUEST",
    "message": {
      "lang": "en-US",
      "value": "[COE0018]Required field 'cust_TargetObject' cannot be
deleted."
    }
  }
}
```

## Creating Links Between Effective-Dated and Non-Effective-Dated Records

You can create links between an effective-dated record and a non-effective-dated record, and vice versa. There are no constraints on effective dates when you create such links.

Parent topic: [Working with Links \[page 98\]](#)

## Related Information

[Querying Links \[page 99\]](#)

[Creating Links \[page 100\]](#)

[Updating Links \[page 101\]](#)

[Deleting Links \[page 103\]](#)

## 5.7 Batch Operations

The OData protocol has introduced the batch operation that can combine multiple requests into a single request. An OData batch request is represented as a multipart MIME v1.0 message, a standard format allowing the representation of multiple parts, each of which may have a different content type, within a single request.

### Batch Request Header

Batch requests are submitted as a single POST request to the `/odata/v2/$batch` URI. The request must include a content-type header specifying a content type of “multipart/mixed” and a boundary specification. This is an example for a batch request header:

#### Sample Code

```
POST /odata/v2/$batch
HOST: localhost
Content-Type: multipart/mixed; boundary=batch_36522ad7-
fc75-4b56-8c71-56071383e77b
```

The batch boundary must follow the pattern `[a-zA-Z0-9_\\-\\. '\\\\+]{1,70}` and it must not contain any special character in `( ) < > @ , ; : / " [ ] ? =`.

#### Note

The maximum number of requests allowed in a batch request is 180. An error occurs if this number is exceeded.

### Batch Request Body

In a batch request, each ChangeSet and query request is represented as a distinct MIME part and is separated by the boundary maker defined in Content-Type. This is an example of a batch request body that includes these operations:

- Query user "admin"
- ChangeSet: Insert a new user
- Replace user "carla"
- Query the attachment.

#### Sample Code

```
--batch_36522ad7-fc75-4b56-8c71-56071383e77b
Content-Type: application/http
Content-Transfer-Encoding:binary
GET User('admin') HTTP/1.1
--batch_36522ad7-fc75-4b56-8c71-56071383e77b
Content-Type: multipart/mixed; boundary=changeset_77162fcd-b8da-41ac-
a9f8-9357efbbd621
```

```
--changeset_77162fcd-b8da-41ac-a9f8-9357efbbd621
Content-Type: application/http
Content-Transfer-Encoding: binary
POST User HTTP/1.1
Content-Type: application/atom+xml;type=entry
<AtomPub representation of a new User>
--changeset_77162fcd-b8da-41ac-a9f8-9357efbbd621
Content-Type: application/http
Content-Transfer-Encoding: binary
PUT User('carla') HTTP/1.1
Content-Type: application/json
<JSON representation of User carla>
--changeset_77162fcd-b8da-41ac-a9f8-9357efbbd621--
--batch_36522ad7-fc75-4b56-8c71-56071383e77b
Content-Type: application/http
Content-Transfer-Encoding: binary
GET Attachment("invalidAttachmentId") HTTP/1.1
--batch_36522ad7-fc75-4b56-8c71-56071383e77b--
```

## Batch Processing

Request parts in a batch are processed sequentially and independently of each other. A request part can be a query request or a ChangeSet. The OData batch process follows these rules:

- All operations in a single ChangeSet unit follow the all-or-nothing rule. Failure of a single unit causes the entire ChangeSet to fail. No partial changes are applied.
- Upsert operations in a ChangeSet also follow the all-or-nothing rule. If multiple records are processed in an upsert operation within a ChangeSet, the failure of one single record causes all other records to fail.
- Each ChangeSet is processed independently. The failure of one ChangeSet doesn't affect others.

## Referencing Requests in a Change Set

If there's an insert request in the ChangeSet and the MIME part representing that request includes a Content-ID header, then the new entity that is created by the insert operation can be referenced by subsequent requests within that ChangeSet by referring to the Content-ID value prefixed with a "\$" character:

### Sample Code

```
--batch_36522ad7-fc75-4b56-8c71-56071383e77b
Content-Type: multipart/mixed; boundary=changeset_77162fcd-b8da-41ac-a9f8-9357efbbd621
--changeset_77162fcd-b8da-41ac-a9f8-9357efbbd621
Content-Type: application/http
Content-Transfer-Encoding: binary
POST User HTTP/1.1
Content-Type: application/atom+xml;type=entry
Content-ID: 1
<AtomPub representation of a new User>
--changeset_77162fcd-b8da-41ac-a9f8-9357efbbd621
Content-Type: application/http
Content-Transfer-Encoding: binary
POST $1 /Manager HTTP/1.1
Content-Type: application/atom+xml;type=entry
<AtomPub representation of a new user manager>
```

```
--changeset_77162fcd-b8da-41ac-a9f8-9357efbbd621--  
--batch_36522ad7-fc75-4b56-8c71-56071383e77b--
```

This batch request:

1. Inserts a new User
2. Insert the User's Manager

## Batch Response Header

The body of a batch response contains a response for each Query and ChangeSet request that makes up the batch request. The order of responses matches the order of requests. Each response includes a Content-Type header with a value of "application/http", and a Content-Transfer-Encoding MIME header with a value of "binary".

### Sample Code

```
--batch_36522ad7-fc75-4b56-8c71-56071383e77b  
Content-Type: application/http  
Content-Transfer-Encoding: binary  
HTTP/1.1 200 Ok  
Content-Type: application/atom+xml;type=entry  
Content-Length: ###  
<AtomPub representation of the User entity with EntityKey admin>  
--batch_36522ad7-fc75-4b56-8c71-56071383e77b  
Content-Type: multipart/mixed; boundary=changeset_77162fcd-b8da-41ac-a9f8-9357efbbd621  
Content-Length: ###  
--changeset_77162fcd-b8da-41ac-a9f8-9357efbbd621  
Content-Type: application/http  
Content-Transfer-Encoding: binary  
HTTP/1.1 201 Created  
Content-Type: application/atom+xml;type=entry  
Location: http://localhost/odata/V2/User('abc')  
Content-Length: ###  
<AtomPub representation of a new User entity>  
--changeset_77162fcd-b8da-41ac-a9f8-9357efbbd621  
Content-Type: application/http  
Content-Transfer-Encoding: binary  
HTTP/1.1 204 No Content  
--changeset_77162fcd-b8da-41ac-a9f8-9357efbbd621--  
--batch_36522ad7-fc75-4b56-8c71-56071383e77b  
Content-Type: application/http  
Content-Transfer-Encoding: binary  
HTTP/1.1 404 Not Found  
Content-Type: application/xml  
Content-Length: ###  
  
<Error message>  
--batch_36522ad7-fc75-4b56-8c71-56071383e77b--
```

### Note

For a successful ChangeSet request, the response lists all the request responses included in the ChangeSet. For example, if a ChangeSet has three requests, then the response contains three responses in the ChangeSet body. However, if the ChangeSet fails, then only one response entry, which is the error in the problematic request is displayed. The ChangeSet response is then a non-ChangeSet response because it doesn't have the ChangeSet body.

[Upsert Behavior in a \\$batch ChangeSet \[page 111\]](#)

Learn how upsert works in a \$batch ChangeSet.

[Examples of \\$batch Request Bodies \[page 113\]](#)

Examples of \$batch requests are listed. Please strictly follow the format in the examples including the new lines and spaces.

## Related Information

<http://www.odata.org/documentation/odata-version-3-0/batch-processing/> ➡

### 5.7.1 Upsert Behavior in a \$batch ChangeSet

Learn how upsert works in a \$batch ChangeSet.

Upsert operations in a \$batch ChangeSet follow the "all-or-nothing" rule. That means, if one record fails to upsert, all other changes will roll back.

#### ⚠ Caution

Upsert parameter `strictTransactionIsolate` is ignored in a ChangeSet.

## Adding Upsert Parameters

You can add upsert parameters in a \$batch ChangeSet. Here's an example:

```
--batch_202010071357
Content-Type: multipart/mixed; boundary=changeset_202010071357
Content-Length: 100
--changeset_202010071357
Content-Transfer-Encoding: binary
Content-Type: application/http
POST upsert?purgeType=full HTTP/1.1
Content-Type: application/json; charset=utf-8
accept: application/json
<JSON/Atom representation of an upsert operation>
--changeset_202010071357--

--batch_202010071357--
```

## Returning All Upsert Errors in a ChangeSet

By default, if errors occur to an upsert operation in a \$batch ChangeSet, only the first error is returned no matter how many entities are in the upsert payload. However, you can use parameter

`enableUpsertResponseExtensionInChangeset` to control whether all error messages of an upsert operation in a `ChangeSet` are returned. To return all upsert error messages, set the parameter to true in the request.

This is an example of the complete upsert errors returned in batch operation:

```
{
  "error": {
    "code": "COE_GENERAL_SERVER_FAILURE",
    "message": {
      "lang": "en-US",
      "value": "ChangeSet index 1 - Upsert error"
    },
    "innererror": {
      "errordetails": [
        {
          "code": "cust_MDF35137/
effectiveStartDate=2000-01-01T00:00:00.000Z,cust_MDF35137/externalCode=P1",
          "message": "Invalid value 'DummyValue' for the Generic Object
field 'cust_gof'.;Invalid Picklist value: 'DummyValue'.;Invalid User ID: 'abc123'.",
          "severity": "error",
          "target": ""
        },
        {
          "code": "cust_MDF35137/
effectiveStartDate=2000-01-01T00:00:00.000Z,cust_MDF35137/externalCode=P1",
          "message": "Invalid value 'DummyValue' for the Generic Object
field 'cust_gof'.;Invalid Picklist value: 'DummyValue'.;Invalid User ID: 'abc123'.",
          "severity": "error",
          "target": ""
        },
        {
          "code": "cust_MDF35137/
effectiveStartDate=2001-01-01T00:00:00.000Z,cust_MDF35137/externalCode=P1",
          "message": "Invalid User ID: 'abc123'.",
          "severity": "error",
          "target": ""
        }
      ]
    }
  }
}
```

Parent topic: [Batch Operations \[page 108\]](#)

## Related Information

[Examples of \\$batch Request Bodies \[page 113\]](#)

[enableUpsertResponseExtensionInChangeset \[page 96\]](#)

[enableUpsertResponseExtensionInChangeset \[page 96\]](#)



## 5.7.2 Examples of \$batch Request Bodies

Examples of \$batch requests are listed. Please strictly follow the format in the examples including the new lines and spaces.

### Example of an Edit + Upsert

An upsert request is a special kind of the update request. It should be included in a ChangeSet.

```
--batch_36522ad7-fc75-4b56-8c71-56071383e77b
Content-Length: 100
Content-Type: multipart/mixed; boundary=changeset_77162fcd-b8da-41ac-a9f8-9357efbbd621
--changeset_77162fcd-b8da-41ac-a9f8-9357efbbd621
Content-Transfer-Encoding: binary
Content-Type: application/http
POST upsert HTTP/1.1
Content-Type: application/json;charset=utf-8
accept:application/json
{
  "__metadata": {
    "uri": "User('content1')"
  },
  "username": "content1",
  "password": "content1",
  "hireDate": "/Date(978307200000)/",
  "status": "active",
  "userId": "content1",
  "firstName": "content1",
  "lastName": "helloterry@abc.com",
  "department": "test",
  "timeZone": "PST",
  "hr": {
    "__metadata": {
      "uri": "User('admin')"
    }
  },
  "manager": {
    "__metadata": {
      "uri": "User('admin')"
    }
  }
}
--changeset_77162fcd-b8da-41ac-a9f8-9357efbbd621
Content-Transfer-Encoding: binary
Content-Type: application/http
POST upsert HTTP/1.1
Content-Type: application/json;charset=utf-8
accept:application/json
{
  "__metadata": {
    "uri": "User('content3')"
  },
  "lastModified": "/Date(3923749827424)/"
}
--changeset_77162fcd-b8da-41ac-a9f8-9357efbbd621--

--batch_36522ad7-fc75-4b56-8c71-56071383e77b--
```

## Example of a Query + Edit + Upsert

```
--batch_36522ad7-fc75-4b56-8c71-56071383e77b
Content-Type: application/http
Content-Transfer-Encoding: binary
GET User('admin')?$format=json HTTP/1.1
Content-Type: application/atom+xml;type=entry

--batch_36522ad7-fc75-4b56-8c71-56071383e77b
Content-Length: 100
Content-Type: multipart/mixed; boundary=changeset_77162fcd-b8da-41ac-a9f8-9357efbbd621
--changeset_77162fcd-b8da-41ac-a9f8-9357efbbd621
Content-Transfer-Encoding: binary
Content-Type: application/http
POST upsert HTTP/1.1
Content-Type: application/json;charset=utf-8
accept:application/json
{
  "__metadata": {
    "uri": "User('content1')"
  },
  "username": "content1",
  "password": "content1",
  "hireDate": "/Date(978307200000)/",
  "status": "active",
  "userId": "content1",
  "firstName": "content1",
  "lastName": "helloterry@abc.com",
  "department": "test",
  "timeZone": "PST",
  "hr": {
    "__metadata": {
      "uri": "User('admin')"
    }
  },
  "manager": {
    "__metadata": {
      "uri": "User('admin')"
    }
  }
}
--changeset_77162fcd-b8da-41ac-a9f8-9357efbbd621
Content-Transfer-Encoding: binary
Content-Type: application/http
POST User('content1') HTTP/1.1
Content-Type: application/json;charset=utf-8
X-HTTP-METHOD: MERGE
{
  "manager": {
    "__metadata": {
      "uri": "User('admin')"
    }
  }
}
--changeset_77162fcd-b8da-41ac-a9f8-9357efbbd621--

--batch_36522ad7-fc75-4b56-8c71-56071383e77b--
```

## Example of Upsert with a Parameter

```
--batch_202010071357
Content-Type: multipart/mixed; boundary=changeset_202010071357
Content-Length: 100
--changeset_202010071357
Content-Transfer-Encoding: binary
Content-Type: application/http
POST upsert?purgeType=full HTTP/1.1
Content-Type: application/json; charset=utf-8
accept: application/json
[
{
  "__metadata": {
    "uri": "User('content1')"
  },
  "username": "content1",
  "password": "content1",
  "hireDate": "/Date(978307200000)/",
  "status": "active",
  "userId": "content1",
  "lastName": "eyc@abc.com"
},
{
  "__metadata": {
    "uri": "User('content2')"
  },
  "username": "content2",
  "password": "content2",
  "hireDate": "/Date(978307200000)/",
  "status": "active",
  "userId": "content2",
  "firstName": "content2",
  "lastName": "example2@abc.com",
  "department": "test",
  "timeZone": "PST"
}
]
--changeset_202010071357--
--batch_202010071357--
```

Parent topic: [Batch Operations \[page 108\]](#)

## Related Information

[Upsert Behavior in a \\$batch ChangeSet \[page 111\]](#)

## 6 MDF OData API

This section provides information about MDF OData API entities including MDF Generic Object entities and MDF Foundation Object entities.

When you create a new MDF generic object (GO), you can choose whether you want to expose it to OData API. Once exposed, you can access the object through OData API calls. In this section, we'll introduce what you need to do to expose an MDF object, how to query and edit data through API calls, and what rules you need to follow when accessing MDF OData APIs.

In addition to GO entities, you'll also learn how MDF Foundation Object entities work. MDF Foundation Objects are Employee Central Foundation Objects which have been migrated to MDF and exposed to OData as OData API entities. Many MDF Foundation Objects adopt the same querying and editing behaviors as MDF Generic Objects. For more information, see [MDF Foundation Objects](#).

Before you proceed, make sure you have the basic knowledge of OData API and you're already familiar with configuring an MDF object definition:

- [OData Operations](#)
- [Configuring the Object Definition](#)

### 6.1 Configuring API Visibility for MDF Generic Objects

You can expose an MDF generic object to OData API by setting the [API Visibility](#) field when you create or edit the object definition.

#### Prerequisites

MDF is enabled for your company and you have the corresponding permissions to configure object definitions. For more information, see the [Enabling MDF](#) section of the [Implementing the Metadata Framework \(MDF\)](#) guide.

#### Context

The OData API exposes the object definition, field definition, rules, and conditions of an MDF object as OData metadata. This makes it easier for other modules to consume the data.

MDF entities that are associated with more than one parent will be exposed to OData API if one of the parents is exposed. For example, the Job Profile Builder entity makes its child objects exposed as API entities, such as job codes and competencies.

You can use the [API Visibility](#) field to control whether an MDF object is to be exposed to OData API or not, and whether it's editable or read only through API.

## Procedure

1. Go to the [Configure Object Definitions](#) page and choose one of the following options:
  - To create a new object definition, select [Object Definition](#) in the [Create New](#) dropdown list.
  - To edit an existing object definition, select [Object Definition](#) in the [Search](#) dropdown list and type the name of the object definition to search. Click and open the object definition from the result list, and choose [Take Action](#) > [Make Correction](#).
2. On the [Object Definition](#) screen, you have the following options in the [API Visibility](#) field. Choose either [Editable](#) or [Read Only](#) to allow OData API access to the object:
  - [Editable](#) - This option allows you to create and edit object instances through OData API.
  - [Read Only](#) - This option only allows you to query object instances through OData API.
  - [Not Visible](#) - The object isn't exposed to OData API.
3. In the [API Sub Version](#) field, choose the subversion for your object. For more information, see [API Subversioning \[page 117\]](#).
4. Proceed with the rest of the configuration tasks and choose [Save](#).

## Results

The MDF object is now exposed to OData. The OData metadata will be automatically refreshed shortly after and you're able to access the object through the corresponding OData API.

### i Note

Depending on the frequency of metadata refreshing job scheduled for your instance, the change may not take effect immediately. You can manually refresh the metadata using one of the following approaches:

- Go to [Admin Center](#) > [OData Metadata Refresh and Export](#) and choose [Refresh](#) to refresh the metadata.
- Send an API request to refresh the metadata:

```
GET https://<API-endpoint-URL>/odata/v2/refreshMetadata
```

## 6.1.1 API Subversioning

You can control whether you want to expose system fields of an MDF object to the OData API with API subversioning.

An MDF object contains a list of technical system fields. When exposed to OData API, all of them may not be needed. With the API subversioning feature, you can choose to hide these fields in the MDF OData API entity.

Depending on the [MDF system field in API Mode](#) Provisioning settings for your company, the system fields may be:

- Not exposed for all MDF entities regardless of the [API Sub Version](#) field setting in the MDF object definition

- Exposed for all MDF entities regardless of the *API Sub Version* field setting in the MDF object definition
- (By default) Dependent on the *API Sub Version* field setting of the MDF object definition

If you're not sure which provisioning setting was made for your company, contact your SAP implementation partner or Product Support.

In *Configure Object Definitions*, you have the following options in the *API Sub Version* field:

- V1.0: System fields following the rules of the subversion are exposed.
- V1.1: System fields following the rules of the subversion are hidden.
- No Selection: The same as using V1.1.

### Note

The *API Sub Version* setting takes effect only when the provisioning setting of MDF system field for API is set to be dependent on it. Otherwise, this field is ignored.

Selecting or changing to an API subversion can affect the metadata of the OData API. As a result, your API request may fail because a nonexistent property. We recommend that you check the metadata of the MDF OData API entity and compare it against your API request as a necessary troubleshooting step.

## List of MDF System Fields

The following table lists all MDF system fields and whether they can be removed:

Field Name	Following the rule of subversions	Notes
mdfSystemCreatedBy	Yes	Equivalent to <code>createdBy</code> property in OData
mdfSystemCreatedDate	Yes	Equivalent to <code>createdDateTime</code> property in OData
mdfSystemLastModifiedby	Yes	Equivalent to <code>lastModifiedBy</code> property in OData
mdfSystemLastModifiedDate	Yes	Equivalent to <code>lastModifiedDateTime</code> property in OData
mdfSystemLastModifiedDateWithTZ	Yes	Equivalent to <code>lastModifiedDateTime</code> property in OData
LastModifiedDateWithTZ	Yes	Equivalent to <code>lastModifiedDateTime</code> property in OData
mdfSystemEntityId	Depends	Exposed when the field name doesn't start with <code>mdfSystem</code>

Field Name	Following the rule of subversions	Notes
mdfSystemRecordId	Depends	<b>i Note</b> Users can overwrite the system field name during object definition. If the user-defined name doesn't start with <code>mdfSystem</code> , this field is exposed.
mdfSystemObjectType	Yes	
mdfSystemStatus	Depends	Exposed when the field visibility is set to be visible
mdfSystemStatusNav	Depends	Exposed when the field visibility is set to be visible
mdfSystemRecordStatus	No	This field indicates pending data and won't be removed in OData API
mdfSystemVersionId	Yes	This field indicates a pending history record. Pending history records aren't supported in OData API.
mdfSystemEffectiveStartDate	Yes	This field can be removed when the MDF object isn't effective dated.
mdfSystemEffectiveEndDate	Yes	This field can be removed when the MDF object isn't effective dated.
mdfSystemTransactionSequence	Yes	This field can be removed when the effective dating of the MDF object isn't Multiple Changes Per Day.

## 6.2 MDF Foundation Object Entities

Employee Central Foundation Objects migrated to the Metadata Framework (MDF).

MDF Foundation Object entities are Employee Central Foundation Objects that have been migrated to the Metadata Framework (MDF) and exposed as OData API entities. MDF Foundation Object entities follow the general rules of MDF OData operations with a few exceptions.

Here's a list of MDF Foundation Object entities:

OData API Entity	MDF Foundation Object
FOBusinessUnit	Business Unit
FOCompany	Legal Entity
FOCostCenter	Cost Center
FODepartment	Department
FODivision	Division

OData API Entity	MDF Foundation Object
FOJobCode	Job Classification
FOJobClassLocal<Country/Region>	Job Classification <Country/Region>
FOJobFunction	Job Function
FOLegalEntityLocal<Country/Region>	Legal Entity <Country/Region>
FOPayGroup	Pay Group

For more information, see [MDF Foundation Objects](#).

## 6.3 Data Mapping Between MDF and OData API

This topic contains information about the mapping of MDF objects to the OData API entities.

Each MDF object can be mapped to one OData entity only. Mapping an MDF object to multiple OData entities is not allowed. When an MDF object is exposed to OData API, the code of the MDF object is used as the name of the OData API entity type. In the following sections, you'll find detailed mapping relationships between an MDF object and its corresponding OData API entity.

### i Note

Composite objects with multiple parents will not be exposed to OData API.

If a parent object is not exposed to OData API, none of its composite children will be exposed regardless of their API visibility settings.

## Mapping of General Terms

MDF Object Definition	OData API
Object	Entity Set or Entity Type
Field	Property/Navigation Property
Association	Navigation Property

## Mapping of Attributes

The following table shows the mapping relationship between the MDF field/association attributes and OData entity set attributes.



Property Attribute of an OData Entity Type	Attribute of the Corresponding MDF Field/Association	Mapping Rule
Property name	Field/association name	<p>The following rules apply:</p> <ul style="list-style-type: none"> <li>• In general, the name of OData property attribute is the same as the attribute name of the MDF object.</li> <li>• If an MDF field is linked to another object, or if it's an association, the name appears in OData with the 'Nav' suffix. See the <a href="#">Mapping of Data Types [page 123]</a> section for more details.</li> <li>• The following MDF field names are hardcoded and appear "as-is" in OData: <ul style="list-style-type: none"> <li>◦ createdBy</li> <li>◦ createdDateTime</li> <li>◦ lastModifiedBy</li> <li>◦ lastModifiedDateTime</li> </ul> </li> </ul>
Property type	Field data type	For detailed mapping of data types, see <a href="#">Mapping of Data Types [page 123]</a> .
Nullable	There is no equivalent attribute in MDF.	The <code>sap:nullable</code> property is always true.
Required	Required and Visibility	<p>When the API Visibility of the MDF object is set to Read Only, the values of <code>sap:required</code>, <code>sap:creatable</code>, <code>sap:updatable</code>, and <code>sap:upsertable</code> attributes are always false regardless of the field required and visibility settings.</p> <p>When the API Visibility of the MDF object is set to Editable, these field values correspond to the <a href="#">Required</a> and <a href="#">Visibility</a> attributes of the MDF fields.</p> <p>For a detailed mapping of these attributes, see the table below in <a href="#">Mapping of Edit Attributes</a>.</p>
Creatable	Visibility	
Updatable	Visibility	
Upsertable	Visibility	
Visible	There is no equivalent attribute in MDF.	The <code>sap:visible</code> attribute is always true.
Sortable	There is no equivalent attribute in MDF.	The <code>sap:sortable</code> attribute is always true except for the CLOB-type field and localized property of a translatable field.
Filterable	There is no equivalent attribute in MDF.	The <code>sap:filterable</code> attribute is always true except for the CLOB-type field and localized property of a translatable field.

## Mapping of Edit Attributes

The following table explains how the visibility and edit attributes of an MDF field works when exposed to OData API.

### Note

When a MDF field maps to both property and navigation property in OData, the `sap:required` attribute of these properties may take different values. These exception are documented in the notes below.

MDF Field Attribute		OData Property Attribute			
Required	Visibility	sap:required	sap:creatable	sap:updatable	sap:upsertable
Yes	Editable	true	true	true	true
		<b>i Note</b> Exception: for properties mapped from Generic Objects, User, and Picklist, and navigation properties mapped from Enum type, the value is <code>false</code> .			
Yes	Read Only	false	false	false	false
		<b>i Note</b> Exception: this value is <code>true</code> for navigation properties that are mapped from associations and from these types: Attachment, Generic Object, Picklist, and User.			
Yes	Not Visible	false	false	false	false
		<b>i Note</b> Exception: this value is <code>true</code> for navigation properties that are mapped from associations and from these types: Attachment, Generic Object, Picklist, and User.			
No	Editable	false	false	false	false
No	Read Only	false	false	false	false
No	Not Visible	false	false	false	false

For transient fields and optimistic lock UUID, the following rules apply:

	sap:required	sap:createable	sap:updateable	sap:upsertable	sap:sortable	sap:filterable
Transient fields ( <a href="#">What is a transient field?</a> )	false	false	false	false	false	false
Optimistic Lock UUID	false	false	true	true	false	false

## Mapping of Data Types

The following table shows the mapping relationship between the data types of MDF fields and OData properties.

MDF Field Type	OData Property Type	Naming Rule	Max Length	Multiplicity (For Navigation Properties)
Attachment	Navigation property to the Attachment entity.	Add Nav suffix to MDF field name. Example: cust_MyAttachmentNav	N/A	*:1
<div> <div>i Note</div> <div> <p>Inline editing of the attachment navigation property is not allowed. To add an attachment for an MDF entity, you first create the attachment with the Attachment OData API, and then add the attachment to the MDF entity. For more information, see <a href="#">Uploading Attachments to an MDF Entity</a> [page 173].</p> </div> </div>				
Auto Complete	String	Same name as MDF field.	N/A	N/A
Auto Number	Int64	Same name as MDF field.	N/A	N/A

MDF Field Type	OData Property Type	Naming Rule	Max Length	Multiplicity (For Navigation Properties)
Boolean	Boolean	Same name as MDF field.	N/A	N/A
CLOB	String  You can use this data type to store a block of text. The advantage of this data type is that access is fast because an application program can access any portion of a CLOB object. Additionally, this data type provides large storage for a user-defined data type.	Same name as MDF field.	N/A	N/A
Date	DateTime	Same name as MDF field.	N/A	N/A
DateTime	DateTimeOffset	Same name with the following exceptions: <ul style="list-style-type: none"> <li>MDF field <code>createdDate</code> maps to OData property type <code>createdDateTime</code></li> <li>MDF field <code>lastModifiedDate</code> maps to OData property type <code>lastModifiedDateTime</code></li> </ul>	N/A	N/A
Decimal	Decimal	Same name as MDF field.	N/A	N/A
Enum	One string property.  One navigation property to the <code>MDFEnumValue</code> entity.	Property: same name as MDF field name.  Navigation property: add Nav suffix to MDF field name. Example: <code>cust_MyEnumNav</code>	Same max length as defined in MDF field. Default value is 255.	*:1

MDF Field Type	OData Property Type	Naming Rule	Max Length	Multiplicity (For Navigation Properties)
Foundation Object	One string property.  One navigation property to the Foundation Object entity.	Property: same name as MDF field.  Navigation property: add Nav suffix to MDF field name. Example: cust_MyObjectNav	32	*:1
Generic Object	One string property.  One navigation property to the Generic Object entity.	Property: same name as MDF field.  Navigation property: add Nav suffix to MDF field name. Example: cust_MyObjectNav	128	If the source object is not effective dated and destination object is effective dated, or if the destination object is PickListValue the multiplicity is *:*.  Otherwise, if the field is required, the multiplicity is *:1; if it's optional, the multiplicity is *:0..1.
Number	Int64	Same name as MDF field.	N/A	N/A
Picklist	One string property.  One navigation property to the destination picklist entity.	Property: same name as MDF field name.  Navigation property: add Nav suffix to MDF field name. Example: cust_MyPicklistNav	N/A	If the source object is effective dated, the multiplicity is *:*.  Otherwise, if the field is required, the multiplicity is *:1; if it's optional, the multiplicity is *:0..1.
String	String	Same name as MDF field.	N/A	N/A
Time	Time  On UI, times are entered and displayed in <b>HH:MM:SS</b> format, for example: <b>10:30:59</b> .  In OData API, times are entered and displayed in duration format, for example: <b>PT10H30M59S</b>	Same name as MDF field.	N/A	N/A

MDF Field Type	OData Property Type	Naming Rule	Max Length	Multiplicity (For Navigation Properties)
Translatable	<p>One string property for default value, one string property for localized label, and a set of string properties representing labels in each locale with locales names added as suffixes.</p> <p>One navigation property to the MDFLocalizedValue entity.</p>	<p>Default value property: add <code>_defaultValue</code> suffix to MDF field name. Example, <code>cust_myLabel_defaultValue</code>.</p> <p>Localized value property: add <code>_localized</code> suffix to MDF field name. Example, <code>cust_myLabel_localized</code>.</p> <p>Locale properties: add locale suffixes to MDF field names. Examples: <code>cust_myLabel_en_US</code>, <code>cust_myLabel_zh_CN</code>.</p> <p>Navigation property: add <code>_TranslationTextNav</code> suffix to the MDF field name. Example: <code>cust_myLabel_TranslationTextNav</code>.</p>	Same max length as defined in MDF field. Default value is 255.	1:*

MDF Field Type	OData Property Type	Naming Rule	Max Length	Multiplicity (For Navigation Properties)
User	One string property.  One navigation property to <code>User</code> entity.	Property: same name as MDF field name.  Navigation property: add <code>Nav</code> suffix to MDF field name. Example: <code>cust_MyUserNav</code>	100	1:1 if the field is required; 1:0..1 if it's optional.

**i Note**

The user-type audit fields `mdfSystemCreatedBy` and `mdfSystemLastModifiedBy` are mapped to the following read-only properties in OData API by default:

- Properties: `createdBy`, `lastModifiedBy`
- Navigation Properties: `createdByNav`, `lastModifiedByNav`

Reverse navigation is not allowed for the navigation properties.

MDF Field Type	OData Property Type	Naming Rule	Max Length	Multiplicity (For Navigation Properties)
-	wfRequestNav	When workflow-enabled MDF objects are exposed to OData, a wfRequestNav navigation property is added automatically to the OData API entity. You can use this navigation property to retrieve the related workflow information for pending data.  For more information, see <a href="#">Data Mapping Between MDF and OData API [page 120]</a> .	N/A	1:0..*

## Mapping of Associations

The associations in an MDF object are mapped to navigation properties in OData API with the same names. The associations are mapped as shown in the following table.

MDF Association	Mapped to OData API	Multiplicity
Composite	A navigation property to the <b>child</b> MDF entity in the association.	If it's a 1:M association, or if the source object is not effective dated and destination object is effective dated, the multiplicity is 1:*.  Otherwise, if the association is required, the multiplicity is 1:1; if it's optional, the multiplicity is 1:0..1.
Valid When	A navigation property to the <b>referred</b> MDF entity in the association.	If it's a 1:M association, or if the source object is not effective dated and destination object is effective dated, the multiplicity is *:*.  Otherwise, if the association is required, the multiplicity is *:1; if it's optional, the multiplicity is *:0..1.



MDF Association	Mapped to OData API	Multiplicity
Join By Column	A navigation property to the <b>referred</b> destination entity.	<p>If it's a 1:M association, or if the source object is not effective dated and destination object is effective dated, the multiplicity is *:*. </p> <p>Otherwise, if the association is required, the multiplicity is *:1; if it's optional, the multiplicity is *:0..1.</p>

## Defining Key Properties for OData Entity

The key properties of an MDF OData API entity depend on the effective dating setting for the MDF object. The following table explains which MDF fields are used as keys in OData.

### Note

The field names referred to in this table reflect the default names which correspond to the Database Field Name column. If you have changed the default names, the key property names should reflect these changes.

Effective Dating	Number of Keys	Key Properties
None	1	<code>externalCode</code>
Basic	2	<code>externalCode</code> <code>effectiveStartDate</code>
Multiple Changes Per Day	3	<code>externalCode</code> <code>effectiveStartDate</code> <code>transactionSequence</code>

Effective Dating	Number of Keys	Key Properties
From Parent	2 or more depending on the parent entity.	<p><code>externalCode</code></p> <p><code>&lt;ParentEntityName&gt;_externalCode</code></p> <p><code>&lt;ParentEntityName&gt;_effectiveStartDate</code> - Only available if the effective dating of parent entity is Basic or Multiple Changes Per Day</p> <p><code>&lt;ParentEntityName&gt;_transactionSequence</code> - Only available if the effective dating of parent entity is Multiple Changes Per Day</p>

**Note**

In a multiple-level parent-child relation, the child inherits the `<ParentEntityName>_externalCode` key properties from all its ancestors. That means if there's a 3-level relationship, the "youngest" child will have two `<ParentEntityName>_externalCode` key properties in addition to its own `externalCode`, one from its parent, and the other from the parent's parent. However, there will be only one `<ParentEntityName>_effectiveStartDate` as the effective start date is the same for all entities in a composite association.

## 6.4 Locale Handling in MDF Entities

How MDF handles translatable text fields in OData API.

MDF exposes translatable text fields as OData properties. For each translatable field, these standard properties and navigation properties are available in the entity metadata:

- Property `<fieldname_defaultValue>` represents the default property value. You can assign a value to the property when you create an entry. Otherwise, it takes the value of the locale-specific property.
- Property `<fieldname_localized>`: represents the localized property value in the current locale. The localized value calculated following the rules described in [How Is the Localized Value Determined? \[page 131\]](#).
- Property `<fieldname_locale code>`: represents the localized value in the locale indicated by the locale code.
- Navigation property `<fieldnameTranslationTextNav>`: represents the association with entity `MDFLocalizedValue`, where the localized values of the properties are stored.

For example, a description field in an MDF object has a default property `<description_defaultValue>`, a localized property `<description_localized>`, several locale-specific properties such as `<description_en_US>`, `<description_de_DE>`, `<description_zh_CN>`, etc., and a navigation property `<descriptionTranslationTextNav>`.

All these properties and navigation properties can be queried and edited.

## How Is the Localized Value Determined?

The system determines the value of the `<fieldname_localized>` field based on the following information:

- User locale. The user local corresponds to the language setting the global header bar under [Settings](#) [Languages](#).
- Company locale. The company locale depends on the language setting of your company in provisioning.
- Value of the `<fieldname_defaultValue>` property.

Here's how the `<fieldname_localized>` property value is determined:

1. If the user's locale is available, it takes the property value that corresponds to the user's locale.
2. If user's locale isn't available, it takes the property value that corresponds to company's locale.
3. If neither locale is available, it takes the property value of `<fieldname_defaultValue>`.

### Note

Both queries and edits follow the same rules. If you edit the value of `<fieldname_localized>`, the corresponding field of the locale is also changed. For example, the `<name_localized>` field of the `FOBusinessUnit` in your company comes from `en_US` field. If you modify this field with an API request, the new value is also reflected in the `<name_en_US>` field.

## Example 1: Query an Entry's Localized Label

This type of request can be useful especially for UI consumption where you need to retrieve a record with the labels of the default locale. Here's a sample request using the `FOBusinessUnit` entity as an example:

### Request

Operation	Query
HTTP Method	GET
URI	<code>https://&lt;API-Server&gt;/odata/v2/ FOBusinessUnit(externalCode='MC_AVIA',start tDate=datetime'1900-01-01T00:00:00')? \$format=JSON&amp; \$select=externalCode,name_localized</code>

### Response

```
{
  "d": {
    "__metadata": {
      "uri": "https://<API-Server>/odata/v2/  
FOBusinessUnit(externalCode='MC_AVIA',start  
tDate=datetime'1900-01-01T00:00:00')",
      "type": "SFOData.FOBusinessUnit"
    },
    "externalCode": "MC_AVIA",
    "name_localized": "MC Aviation"
  }
}
```

## Example 2: Query All Labels of an Entry

You often find this type of usage in integration scenarios where you want to export the labels in all languages.

### Request

Operation	Query
HTTP Method	GET
URI	<code>https://&lt;API-Server&gt;/odata/v2/ FOBusinessUnit(externalCode='SVCS',startDa te=datetime'1900-01-01T00:00:00')/ nameTranslationTextNav?\$format=JSON</code>

### Response

```
{
  "d": {
    "results": [
      {
        "__metadata": {
          "uri": "https://<API-Server>/odata/v2/  
MDFLocalizedValue('en_US')",
          "type": "SFOData.MDFLocalizedValue"
        },
        "locale": "en_US",
        "value": "Services"
      },
      {
        "__metadata": {
          "uri": "https://<API-Server>/odata/v2/  
MDFLocalizedValue('defaultValue')",
          "type": "SFOData.MDFLocalizedValue"
        },
        "locale": "defaultValue",
        "value": "Services"
      },
      {
        "__metadata": {
          "uri": "https://<API-Server>/odata/v2/  
MDFLocalizedValue('en_GB')",
          "type": "SFOData.MDFLocalizedValue"
        },
        "locale": "en_GB",
        "value": "Services"
      },
      {
        "__metadata": {
          "uri": "https://<API-Server>/odata/v2/  
MDFLocalizedValue('zh_CN')",
          "type": "SFOData.MDFLocalizedValue"
        },
        "locale": "zh_CN",
        "value": "服务"
      },
      {
        "__metadata": {
          "uri": "https://<API-Server>/odata/v2/  
MDFLocalizedValue('pt_BR')",
          "type": "SFOData.MDFLocalizedValue"
        },
        "locale": "pt_BR",
        "value": "Serviços"
      }
    ]
  }
}
```

```

    },
    {
      "__metadata": {
        "uri": "https://<API-Server>/odata/v2/
MDFLocalizedValue('ja_JP')",
        "type": "SFOData.MDFLocalizedValue"
      },
      "locale": "ja_JP",
      "value": "サービス統括本部"
    },
    {
      "__metadata": {
        "uri": "https://<API-Server>/odata/v2/
MDFLocalizedValue('ru_RU')",
        "type": "SFOData.MDFLocalizedValue"
      },
      "locale": "ru_RU",
      "value": "Сервисы"
    },
    {
      "__metadata": {
        "uri": "https://<API-Server>/odata/v2/
MDFLocalizedValue('es_ES')",
        "type": "SFOData.MDFLocalizedValue"
      },
      "locale": "es_ES",
      "value": "Servicios"
    },
    {
      "__metadata": {
        "uri": "https://<API-Server>/odata/v2/
MDFLocalizedValue('ko_KR')",
        "type": "SFOData.MDFLocalizedValue"
      },
      "locale": "ko_KR",
      "value": "서비스즈"
    }
  ]
}
}
}

```

### Example 3: Creating or Editing an Entry with Localized Texts

When you create or edit an entry, you can include the localized texts of translatable fields either through the local-specific properties or through inline-editing the `<fieldname>TranslationTextNav` navigation property. Here are some examples:

#### Request Info

Operation	Upsert
HTTP Method	POST
URI	https://<API-Server>/odata/v2/upsert

Payload example for upserting property values:

```

{
  "__metadata": {
    "uri": "FOBusinessUnit"
  },

```

```

    "externalCode": "SERVICE",
    "status": "A",
    "startDate": "/Date(1508375121000)/",
    "name_ko_KR": "서비스",
    "name_pt_BR": "Serviços",
    "name_de_DE": "Dienstleistungen",
    "name_zh_TW": "服務",
    "name_es_ES": "Servicios",
    "name_ru_RU": "Сервисы",
    "name_fr_FR": "Services",
    "name_ja_JP": "サービス統括本部",
    "name_en_US": "Services",
    "name_zh_CN": "服务"
  }

```

Payload example for inline-upserting the navigation property:

### Sample Code

```

{
  "__metadata": {
    "uri": "F0BusinessUnit"
  },
  "externalCode": "SERVICE",
  "status": "A",
  "startDate": "/Date(1508375121000)/",
  "nameTranslationTextNav": [
    {
      "locale": "ko_KR",
      "value": "서비스"
    },
    {
      "locale": "pt_BR",
      "value": "Serviços"
    },
    {
      "locale": "de_DE",
      "value": "Dienstleistungen"
    },
    {
      "locale": "zh_TW",
      "value": "服務"
    },
    {
      "locale": "es_ES",
      "value": "Servicios"
    },
    {
      "locale": "ru_RU",
      "value": "Сервисы"
    },
    {
      "locale": "fr_FR",
      "value": "Services"
    },
    {
      "locale": "ja_JP",
      "value": "サービス統括本部"
    },
    {
      "locale": "en_US",
      "value": "Services"
    },
    {
      "locale": "zh_CN",
      "value": "服务"
    }
  ]
}

```

```
]
}
```

## Response

```
{
  "d": [
    {
      "key": "F0BusinessUnit/externalCode=SERVICE,F0BusinessUnit/
startDate=2017-10-19T00:00:00.000-04:00",
      "status": "OK",
      "editStatus": "UPSERVED",
      "message": null,
      "index": 0,
      "httpCode": 200,
      "inlineResults": null
    }
  ]
}
```

## 6.5 MDF OData API Operations

Describes the MDF OData API behaviors with examples.

MDF OData API is based on SAP SuccessFactors HXM Suite OData API framework, currently on OData Version 2.0. While the majority of MDF OData API operations follow the same rules defined by the framework, there are patterns specific to MDF when you use MDF OData API entities. In this section, you'll find out details about the specific patterns and how they affect your OData API usage.

### Service Limit for Large MDF OData Requests

There's a service limit for large MDF OData requests. When you query or edit an MDF OData entity and the response returns a large number of records that exceed the system-defined limit, the request fails and an error occurs. You need to adjust the request parameters or payload so that the data size returned is within the limit. Depending on your request, the limit may vary.

#### → Remember

The service limit is designed to optimize MDF OData performance while ensuring all MDF OData APIs continue to work in all instances. It has very little impact on your current integrations and UIs.

If you meet such an error, here are a few steps that can help solve the problem:

- Use pagination in your query if you haven't already.
- If you're already paging your query, try tuning down the page size.
- Include fewer records in your request payload.
- A rule of thumb is to cut the page or record size to half.

If you wish to change the limit, contact Product Support.

## 6.5.1 Permissions

Permissions required for accessing MDF OData API entities.

The following table lists the required permissions for users to query and edit MDF OData API entities:

Permission	Description
<a href="#">▶ Administrator Permissions ▶ Manage Integration Tools ▶ Allow Admin to Access OData API through Basic Authentication ▶</a>	This permission allows users to access OData API entities through Basic Authentication. If you're using other authentication methods such as OAuth 2.0, this permission isn't required.
<a href="#">▶ Administrator Permissions ▶ Metadata Framework ▶ Admin access to MDF OData API ▶</a>	<p>This is the MDF OData admin permission that allows users to query and edit all MDF OData entities. It overrides all entity-level permissions.</p> <p>This permission is recommended only for integration scenarios. Permission roles granted with this permission can override other MDF data access permission configurations and save data without approvals.</p>
<a href="#">▶ Administrator Permissions ▶ Metadata Framework ▶ Access to non-secured objects ▶</a>	<p>This is the nonadmin MDF OData permission. This permission allows users to query and edit all nonsecured MDF OData entities.</p> <div><p><b>i Note</b></p><p>This permission isn't required for querying and editing secured MDF OData entities.</p></div>
Entity-level permissions for secured objects	<p>If an MDF object is secured, an entity-level permission item is available under the corresponding permission category specified in object definition. You can enable view, edit, and import/export permission for this entity. You can also maintain field-level permission overrides.</p> <p>For more information, see <a href="#">Adding Security</a> and <a href="#">Setting Up Security for Fields</a>.</p>

## 6.5.2 Query Operation

This section contains information about query operations on MDF OData entities, including query parameters and examples. Note that only MDF-specific information is documented here. For more information on general OData API operations, refer to the OData Operations section.

[Query Parameters \[page 137\]](#)

A list of query parameters MDF OData API and usage examples.



## 6.5.2.1 Query Parameters

A list of query parameters MDF OData API and usage examples.

Parent topic: [Query Operation \[page 136\]](#)

### 6.5.2.1.1 asOfDate

Use the `asOfDate` parameter to query a single record that is effective on a specific date.

#### Parameter Values

Value	Description
<code>&lt;yyyy-mm-dd&gt;</code>	A numeric date value, for example: 2019-12-25

#### i Note

Returns only one record that is effective on the specified date.

#### i Note

Depending on the parameters you use, a query may return zero, one, or multiple records.

If the effective dating type of the entity is Multiple Changes Per Day (MCPD), the record with the largest sequence number is returned as the effective record of that date.

If you don't specify any date or date period, the query uses today's date as the `asOfDate` value and fetch the current effective record.

#### Use Case

#### → Tip

When you query an effective-dated entity without any date parameters, the result returns a single record effective on the present date. It defaults to `&asOfDate=<today's date>`.

The `asOfDate` parameter retrieves the single records of an entity that is effective on the specified date.

Example:

```
https://<API-Server>/odata/v2/MDFEntity?$format=JSON&asOfDate=2018-11-07
```

## Related Information

[Using asOfDate with \\$expand \[page 143\]](#)

### 6.5.2.1.2 fromDate

Use the `fromDate` parameter to query records that are effective after a specific date.

## Parameter Values

Value	Description
<code>&lt;yyyy-mm-dd&gt;</code>	A numeric date value, for example: 2019-12-25

**i Note**  
Multiple records can be returned.

### i Note

Depending on the parameters you use, a query may return zero, one, or multiple records.

If the effective dating type of the entity is Multiple Changes Per Day (MCPD), the record with the largest sequence number is returned as the effective record of that date.

If you don't specify any date or date period, the query uses today's date as the `asOfDate` value and fetch the current effective record.

## Use Case

The `fromDate` parameter retrieves all effective records of the entity in the specified time period. If the `mdfSystemEffectiveEndDate` of a record is later than the specified date, the record is returned.

Example:

```
https://<API-Server>/odata/v2/MDFEntity?$format=JSON&fromDate=2018-11-07
```

## Related Information

[Using fromDate with \\$expand \[page 145\]](#)

## 6.5.2.1.3 toDate

Use the `toDate` parameter to query records that are effective before a specific date.

### Parameter Values

Value	Description
<code>&lt;yyyy-mm-dd&gt;</code>	A numeric date value, for example: 2019-12-25
<b>i Note</b> Multiple records can be returned in the result.	
<b>i Note</b> Depending on the parameters you use, a query may return zero, one, or multiple records. If the effective dating type of the entity is Multiple Changes Per Day (MCPD), the record with the largest sequence number is returned as the effective record of that date. If you don't specify any date or date period, the query uses today's date as the <code>asOfDate</code> value and fetch the current effective record.	

### Use Case

The `toDate` parameter retrieves all effective records of the entity before the specified date. If the `mdfSystemEffectiveStartDate` of a record is earlier than the specified date, the record is returned.

Example:

```
https://<API-Server>/odata/v2/MDFEntity?$format=JSON&toDate=2018-11-07
```

### Related Information

[Using toDate with \\$expand \[page 148\]](#)

## 6.5.2.1.4 filterParentDate

Use the `filterParentDate` parameter to query a child of an effective-dated entity with date parameters.

In a composite MDF association, a child entity inherits its effective dating attribute from its parent. When you query a composite child entity, date parameters such as `fromDate`, `toDate`, and `asOfDate` are ignored. In order for them to work, you must include parameter `filterParentDate` with value `true` in the query. This parameter allows you to filter the child entity based on the effective dates of its parent.

### Parameter Values

Value	Description
<code>true</code>	Filter child entity based on the effective dates of parent.
<code>false</code> (Default)	Date parameters are ignored on child entities.

### Use Case

See the topic in [Related Links](#) section for detailed use cases.

### Related Information

[Querying the Child of an Effective Dated Entity \[page 151\]](#)

## 6.5.2.1.5 recordStatus

Use the `recordStatus` parameter to query normal, pending, or pending history data of an MDF entity.

### Parameter Values

Value	Description
<code>normal</code> (default)	Parameter value to retrieve normal records.  If <code>recordStatus</code> isn't specified, the query returns normal data by default.

Value	Description
<b>pending</b>	Parameter value to retrieve records pending approval.
<b>pendinghistory</b>	A record in pending history status can be: <ul style="list-style-type: none"> <li>Declined by the approver</li> <li>Canceled by the initiator of the workflow</li> <li>Modified by the approver</li> </ul>

### ⚠ Caution

To query records with pending and pendinghistory status, you must have the [Admin access to MDF OData API](#) permission.

## Use Case

In MDF OData API entities, if workflow routing is configured for an object and the [Pending Data](#) option is enabled, data changes pending approval are stored as pending data. Once the data changes are approved, the records are saved as normal data. If a workflow is declined, canceled, or modified, a pending history is stored in the database. The following table explains how the parameter values map to the `mdfSystemRecordStatus` field of the MDF object:

Parameter Value	<code>mdfSystemRecordStatus</code>
normal	<b>N</b>
pending	<b>P</b>
pendinghistory	<b>PH</b>

You can use parameter `recordStatus` in an OData query to specify which type of data to retrieve.

- To retrieve normal records:

```
https://<API-Server>/odata/v2/cust_TestObject?$format=JSON&recordStatus=normal
```

- To retrieve pending records:

```
https://<API-Server>/odata/v2/cust_TestObject?$format=JSON&recordStatus=pending
```

- To retrieve pending history records:

```
https://<API-Server>/odata/v2/cust_TestObject?
$format=JSON&recordStatus=pendinghistory
```

### i Note

This parameter is required if you want to query the workflow information of an MDF entity by expanding `wfRequestNav`. For more information, see [Retrieving Workflow Information for Pending Data \[page 157\]](#).

## Error Codes

Error Code	Description
COE_UNSUPPORTED_FEATURE	<p>[COE0025]Unsupported feature: Expand workflow navigation from normal data is NOT supported.</p> <p>If you try to expand wfRequestNav on a normal entry, this error occurs. wfRequestNav can only be expanded if it's a pending entry.</p>
COE_GENERAL_FORBIDDEN	<p>[COE0020]Non admin user is NOT allowed to query pending data.</p> <p>Only users with the <a href="#">Admin access to MDF OData API</a> permission can query pending data.</p>
COE_GENERAL_FORBIDDEN	<p>[COE0025]Unsupported feature: Non admin user query pending history data is NOT supported.</p> <p>Only users with the <a href="#">Admin access to MDF OData API</a> permission can query pending data.</p>

### 6.5.2.1.6 versionId

Use the `versionId` parameter to query a specific version of a pending record.

## Parameter Values

Value	Description
<largest version ID> (default)	If versionId is not specified, the query returns the record with the largest version ID.
<user specified version ID>	The record with specified version ID is returned.

## Use Case

### i Note

If you use `versionId` in a query, it returns pending data by default.

A record pending approval can have multiple versions. You can use parameter `versionId` to query a specific version of the record. Example:

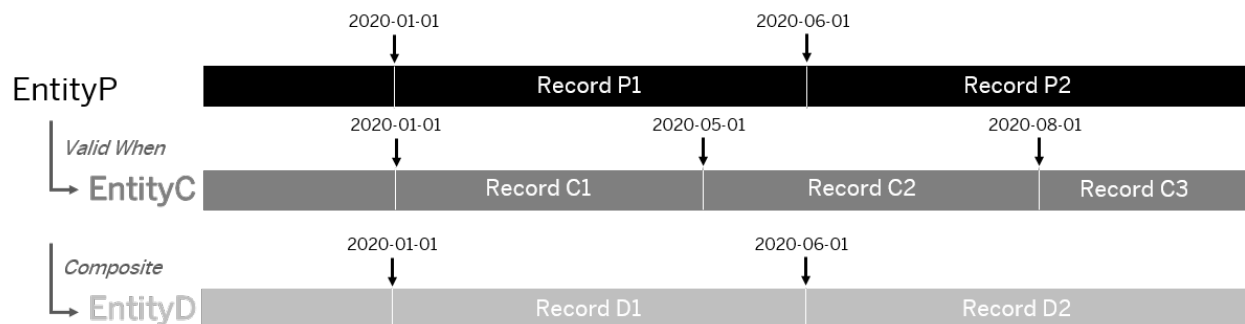
```
https://<API-Server>/odata/v2/TestObject('Key1')?$format=JSON&versionId=2
```

## 6.5.2.2 Sample Operations

### 6.5.2.2.1 Using `asOfDate` with `$expand`

Getting to know how `&asOfDate` impacts the query results when it's used in combination with `$expand` to query effective-dated entities and its associations.

#### Example Entities



#### Composite Association

When querying an effective-dated entity and expanding to a composite association with `&asOfDate`, the restrictive criteria of the parameter only works on the base object. However, as a composite association, EntityD has an `effectiveDate` type "from parent", the expanded data is also effective on this date.

#### Request

Operation	Query
HTTP Method	GET
URI	<code>https://&lt;API-server&gt;/odata/v2/EntityP?\$format=JSON&amp;\$expand=CompositeAssociation&amp;asOfDate=2020-08-15</code>

## Response

### Sample Code

```
{
  "d": {
    "results": [
      {
        "__metadata": {
          "uri": "https://<API-server>/odata/v2/EntityP(effectiveStartDate=datetime'2020-06-01T00:00:00',externalCode='P')",
          "type": "SFOData.cust_EntityP"
        },
        "externalCode": "P",
        "Record": "P2",
        "cust_ParentChildComposite": {
          "results": [
            {
              "__metadata": {
                "uri": "https://<API-server>/odata/v2/cust_EntityD(cust_EntityP_effectiveStartDate=datetime'2020-06-01T00:00:00',cust_EntityP_externalCode='P',externalCode='D')",
                "type": "SFOData.cust_EntityD"
              },
              "externalCode": "D",
              "Record": "D2"
            }
          ]
        }
      }
    ]
  }
}
```

## Valid-When Association

When querying an effective-dated entity and expanding to a valid-when association with the `&asofDate`, the restrictive criteria of the parameter works on both the base object and the expanded data.

### Request

Operation	Query
HTTP Method	GET
URI	<code>https://&lt;API-server&gt;/odata/v2/EntityP?\$format=JSON&amp;\$expand=ValidWhenAssociation&amp;asOfDate=2020-08-15</code>

## Response

### Sample Code

```
{
  "d": {
    "results": [
      {
        "__metadata": {
```





## Composite Association

When querying an effective-dated entity and expanding to a composite association with `&fromDate`, the restrictive criteria of the parameter only works on the base object. However, as a composite association, EntityD has an effectiveDate type "from parent", the expanded data is also effective on this date.

### Request

Operation	Query
HTTP Method	GET
URI	<code>https://&lt;API-server&gt;/odata/v2/EntityP?\$format=JSON&amp;\$expand=CompositeAssociation&amp;fromDate=2020-05-15</code>

### Response

#### Sample Code

```
{
  "d": {
    "results": [
      {
        "__metadata": {
          "uri": "https://<API-server>/odata/v2/EntityP(effectiveStartDate=datetime'2020-01-01T00:00:00',externalCode='P')",
          "type": "SFODData.cust_EntityP"
        },
        "externalCode": "P",
        "Record": "P1",
        "cust_ParentChildComposite": {
          "results": [
            {
              "__metadata": {
                "uri": "https://<API-server>/odata/v2/EntityD(cust_EntityP_effectiveStartDate=datetime'2020-01-01T00:00:00',cust_EntityP_externalCode='P',externalCode='D')",
                "type": "SFODData.cust_EntityD"
              },
              "externalCode": "D",
              "Record": "D1"
            }
          ]
        }
      },
      {
        "__metadata": {
          "uri": "https://<API-server>/odata/v2/EntityP(effectiveStartDate=datetime'2020-06-01T00:00:00',externalCode='P')",
          "type": "SFODData.cust_EntityP"
        },
        "externalCode": "P",
        "Record": "P2",
        "cust_ParentChildComposite": {
          "results": [
            {
              "__metadata": {
                "uri": "https://<API-server>/odata/v2/EntityD(cust_EntityP_effectiveStartDate=datetime'2020-06-01T00:00:00',cust_EntityP_externalCode='P',externalCode='D')",
                "type": "SFODData.cust_EntityD"
              },
              "externalCode": "D",
              "Record": "D2"
            }
          ]
        }
      }
    ]
  }
}
```

```

        "externalCode": "D",
        "Record": "D2"
      }
    ]
  }
}

```

## Valid-When Association

When querying an effective-dated entity and expanding to a valid-when association with the `&fromDate`, the restrictive criteria of the parameter works on both the base object and the expanded data.

### Request

Operation	Query
HTTP Method	GET
URI	https://<API-server>/odata/v2/EntityP?\$format=JSON&\$expand=ValidWhenAssociation&fromDate=2020-05-15

### Response

#### Sample Code

```

{
  "d": {
    "results": [
      {
        "__metadata": {
          "uri": "https://<API-server>/odata/v2/EntityP(effectiveStartDate=datetime'2020-01-01T00:00:00',externalCode='P')",
          "type": "SFODData.cust_EntityP"
        },
        "externalCode": "P",
        "Record": "P1",
        "cust_ParentChildValidWhen": {
          "results": [
            {
              "__metadata": {
                "uri": "https://<API-server>/odata/v2/EntityC(effectiveStartDate=datetime'2020-01-01T00:00:00',externalCode='C')",
                "type": "SFODData.cust_EntityC"
              },
              "externalCode": "C",
              "Record": "C1"
            }
          ]
        }
      },
      {
        "__metadata": {
          "uri": "https://<API-server>/odata/v2/EntityP(effectiveStartDate=datetime'2020-06-01T00:00:00',externalCode='P')",
          "type": "SFODData.cust_EntityP"
        }
      }
    ]
  }
}

```

```

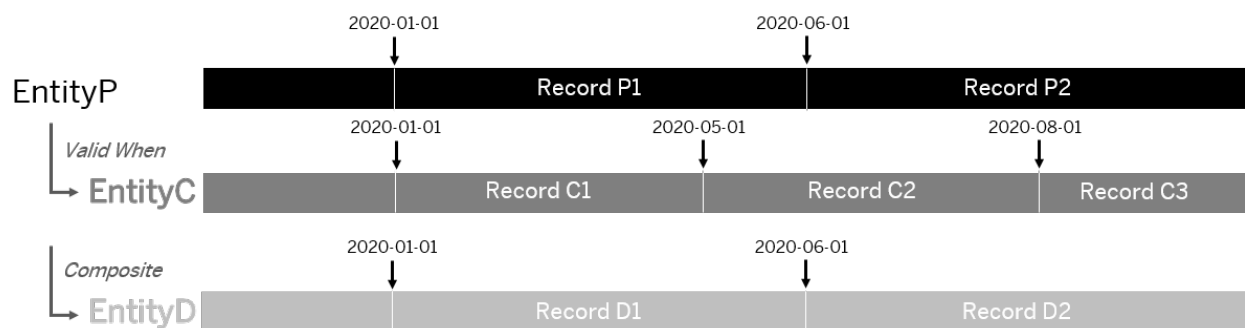
    },
    "externalCode": "P",
    "Record": "P2",
    "cust_ParentChildValidWhen": {
      "results": [
        {
          "__metadata": {
            "uri": "https://<API-server>/odata/v2/
EntityC(effectiveStartDate=datetime'2020-05-01T00:00:00',externalCode='C')",
            "type": "SFODData.cust_EntityC"
          },
          "externalCode": "C",
          "Record": "C2"
        }
      ]
    }
  ]
}

```

### 6.5.2.2.3 Using toDate with \$expand

Getting to know how &toDate impacts the query results when it's used in combination with \$expand to query effective-dated entities and its associations.

#### Example Entities



#### Composite Association

When querying an effective-dated entity and expanding to a composite association with &toDate, the restrictive criteria of the parameter only works on the base object. However, as a composite association, EntityD has an effectiveDate type "from parent", the expanded data is also effective on this date.

## Request

Operation	Query
HTTP Method	GET
URI	<code>https://&lt;API-server&gt;/odata/v2/EntityP?\$format=JSON&amp;\$expand=CompositeAssociation&amp;toDate=2020-07-01</code>

## Response

### Sample Code

```
{
  "d": {
    "results": [
      {
        "__metadata": {
          "uri": "https://<API-server>/odata/v2/EntityP(effectiveStartDate=datetime'2020-01-01T00:00:00',externalCode='P')",
          "type": "SFODData.cust_EntityP"
        },
        "externalCode": "P",
        "Record": "P1",
        "cust_ParentChildComposite": {
          "results": [
            {
              "__metadata": {
                "uri": "https://<API-server>/odata/v2/EntityD(EntityP_effectiveStartDate=datetime'2020-01-01T00:00:00',EntityP_externalCode='P',externalCode='D')",
                "type": "SFODData.cust_EntityD"
              },
              "externalCode": "D",
              "Record": "D1"
            }
          ]
        }
      },
      {
        "__metadata": {
          "uri": "https://<API-server>/odata/v2/EntityP(effectiveStartDate=datetime'2020-06-01T00:00:00',externalCode='P')",
          "type": "SFODData.cust_EntityP"
        },
        "externalCode": "P",
        "Record": "P2",
        "cust_ParentChildComposite": {
          "results": [
            {
              "__metadata": {
                "uri": "https://<API-server>/odata/v2/EntityD(EntityP_effectiveStartDate=datetime'2020-06-01T00:00:00',EntityP_externalCode='P',externalCode='D')",
                "type": "SFODData.cust_EntityD"
              },
              "externalCode": "D",
              "Record": "D2"
            }
          ]
        }
      }
    ]
  }
}
```

```
}
```

## Valid-When Association

When querying an effective-dated entity and expanding to a valid-when association with the `&toDate`, the restrictive criteria of the parameter works on both the base object and the expanded data.

### Request

Operation	Query
HTTP Method	GET
URI	<code>https://&lt;API-server&gt;/odata/v2/EntityP?\$format=JSON&amp;\$expand=ValidWhenAssociation&amp;toDate=2020-07-01</code>

### Response

#### Sample Code

```
{
  "d": {
    "results": [
      {
        "__metadata": {
          "uri": "https://<API-server>/odata/v2/EntityP(effectiveStartDate=datetime'2020-01-01T00:00:00',externalCode='P')",
          "type": "SFODData.cust_EntityP"
        },
        "externalCode": "P",
        "Record": "P1",
        "cust_ParentChildValidWhen": {
          "results": [
            {
              "__metadata": {
                "uri": "https://<API-server>/odata/v2/EntityC(effectiveStartDate=datetime'2020-01-01T00:00:00',externalCode='C')",
                "type": "SFODData.cust_EntityC"
              },
              "externalCode": "C",
              "Record": "C1"
            }
          ]
        }
      },
      {
        "__metadata": {
          "uri": "https://<API-server>/odata/v2/EntityP(effectiveStartDate=datetime'2020-06-01T00:00:00',externalCode='P')",
          "type": "SFODData.cust_EntityP"
        },
        "externalCode": "P",
        "Record": "P2",
        "cust_ParentChildValidWhen": {
          "results": [
            {
              "__metadata": {
                "uri": "https://<API-server>/odata/v2/EntityC(effectiveStartDate=datetime'2020-05-01T00:00:00',externalCode='C')",
```

```

        "type": "SFODData.cust_EntityC"
      },
      "externalCode": "C",
      "Record": "C2"
    }
  ]
}

```

#### 6.5.2.2.4 Querying the Child of an Effective Dated Entity

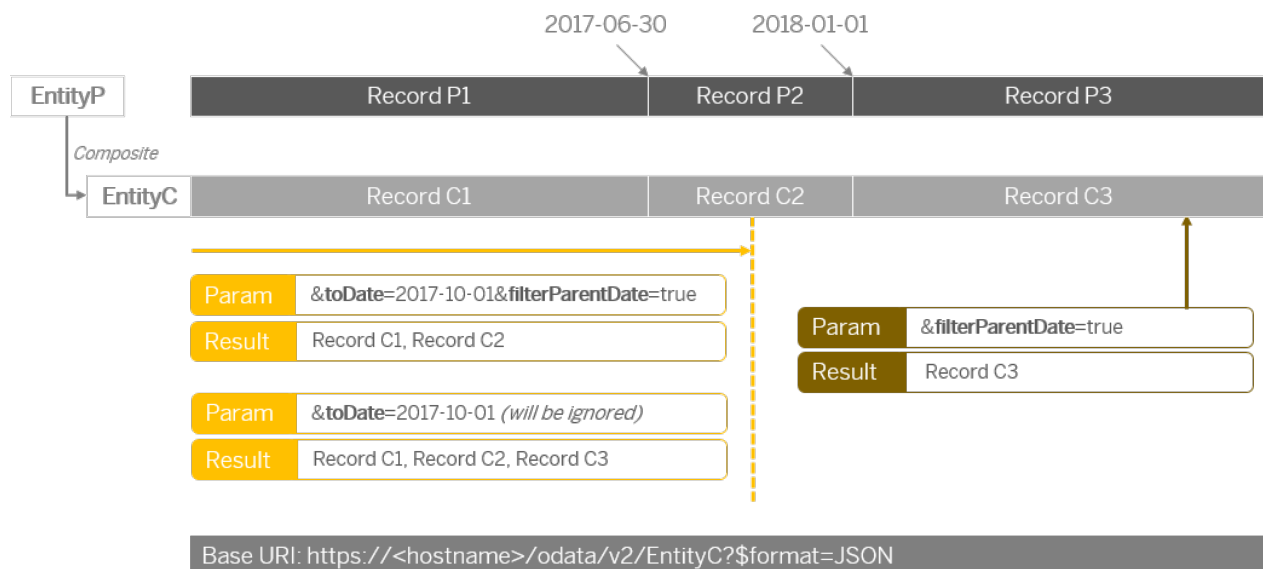
You can use parameter `filterParentDate` to query a child entity of an effective dated entity with date parameters.

In a composite MDF association, a child entity inherits its effective dating attribute from its parent. When you query a composite child entity, date parameters such as `fromDate`, `toDate`, and `asOfDate` will be ignored. In order for them to work, you must include parameter `filterParentDate` with value `true` in the query. This parameter allows you to filter the child entity based on the effective dates of its parent.

##### Note

`filterParentDate` is not a standard OData parameter.

In the example below, `EntityP` has a composite child `EntityC` which has three entries. The different results of the three queries explain how the `filterParentDate` parameter work:



In the first example, the query contains a date parameter and the `filterParentDate` parameter. The date parameter works as intended and the query returns Record C1 and Record C2, both of which are effective in the specified date range (to 2017-10-01):

```
https://<hostname>/odata/v2/EntityC?
$filterParentDate=2017-10-01&filterParentDate=true
```

The next example query contains only a date parameter. Without the `filterParentDate` parameter, the date parameter fails to work and is ignored. The result returns all available entries of the child entity:

```
https://<hostname>/odata/v2/EntityC?$format=JSON&toDate=2017-10-01
```

The last example query contains only the `filterParentDate` parameter. This will be treated as if the parent entity is filtered with the default date parameter, which equals to `&filterParentDate=true&asOfDate=<today's date>`. Therefore, the current effective entry Record C3 is returned.

```
https://<hostname>/odata/v2/EntityC?$format=JSON&filterParentDate=true
```

## 6.5.2.2.5 Querying Links of Effective Dated Entities With Composite Associations

You can use the `$links` option to query links of composite associations of an effective dated entity.

We use `PickListV2` and `PickListValueV2` as example.

### Request

Operation	Query
HTTP Method	GET
URI	<code>https://&lt;API-server&gt;/odata/v2/ PickListV2(effectiveStartDate=datetime'2020-07-17T00:00:00', id='ExamplePicklist')/\$links/values</code>

### Response

#### Sample Code

```
{
  "d": {
    "results": [
      {
        "uri": "https://<API-server>/odata/v2/  
PickListValueV2(PickListV2_effectiveStartDate=datetime'2020-07-17T00:00:00',PickL  
istV2_id='ExamplePicklist',externalCode='ExampleValue1') "
      },
      {
        "uri": "https://<API-server>/odata/v2/  
PickListValueV2(PickListV2_effectiveStartDate=datetime'2020-07-17T00:00:00',PickL  
istV2_id='ExamplePicklist',externalCode='ExampleValue2') "
      }
    ]
  }
}
```



```

        "uri": "https://<API-server>/odata/v2/
PickListValueV2(PickListV2_effectiveStartDate=datetime'2020-07-17T00:00:00',PickL
istV2_id='ExamplePicklist',externalCode='ExampleValue3') "
    },
    {
        "uri": "https://<API-server>/odata/v2/
PickListValueV2(PickListV2_effectiveStartDate=datetime'2020-07-17T00:00:00',PickL
istV2_id='ExamplePicklist',externalCode='ExampleValue4') "
    }
]
}
}

```

## 6.5.2.2.6 Filtering 1:M Composite Association

This topic shows examples of filtering parent and child entities whose association type is *Composite* and multiplicity is *One To Many*.

You can filter records of a parent entity based on properties of its child entities. When the parent and child entities have a one-to-many (1:M) relationship with its children, the query returns all parent records with full list of child entity records even though only one record of child entity meets the condition in the filter.

We use PickListV2 and PickListValueV2 as example entities. You can search for a picklist that contains a certain picklist value or you can search for a picklist whose parent picklist contains a certain picklist value.

### Example 1: Search for a Picklist That Contains a Certain Picklist Value

#### Request

Operation	Query
HTTP Method	GET
URI	https://<API-server>/odata/v2/PickList?\$format=JSON&\$filter=values/externalCode eq 'ABRAND'&\$expand=values

#### Response

The query returns the picklist that has the said picklist value and a full list of child entity records if you choose to expand the association property.

#### Sample Code

```

{
  "d": {
    "results": [
      {
        "__metadata": {
          "uri": "https://<api-server>/odata/v2/
PickListV2(effectiveStartDate=datetime'2020-07-17T00:00:00',id='CARBRAND') ",
          "type": "SFOData.PickListV2"
        },

```

```

        "id": "CARBRAND",
        "status": "A",
        "values": {
            "results": [
                {
                    "__metadata": {
                        "uri": "https://<api-server>/odata/v2/
PickListValueV2(PickListV2_effectiveStartDate=datetime'2020-07-17T00:00:00',PickL
istV2_id='CARBRAND',externalCode='ABRAND')",
                        "type": "SFOData.PickListValueV2"
                    },
                    "externalCode": "ABRAND",
                    "status": "I"
                },
                {
                    "__metadata": {
                        "uri": "https://<api-server>/odata/v2/
PickListValueV2(PickListV2_effectiveStartDate=datetime'2020-07-17T00:00:00',PickL
istV2_id='CARBRAND',externalCode='B BRAND')",
                        "type": "SFOData.PickListValueV2"
                    },
                    "externalCode": "B BRAND",
                    "status": "I"
                },
                {
                    "__metadata": {
                        "uri": "https://<api-server>/odata/v2/
PickListValueV2(PickListV2_effectiveStartDate=datetime'2020-07-17T00:00:00',PickL
istV2_id='CARBRAND',externalCode='C BRAND')",
                        "type": "SFOData.PickListValueV2"
                    },
                    "externalCode": "C BRAND",
                    "status": "I"
                },
                {
                    "__metadata": {
                        "uri": "https://<api-server>/odata/v2/
PickListValueV2(PickListV2_effectiveStartDate=datetime'2020-07-17T00:00:00',PickL
istV2_id='CARBRAND',externalCode='D BRAND')",
                        "type": "SFOData.PickListValueV2"
                    },
                    "externalCode": "D BRAND",
                    "status": "I"
                },
                {
                    "__metadata": {
                        "uri": "https://<api-server>/odata/v2/
PickListValueV2(PickListV2_effectiveStartDate=datetime'2020-07-17T00:00:00',PickL
istV2_id='CARBRAND',externalCode='E BRAND')",
                        "type": "SFOData.PickListValueV2"
                    },
                    "externalCode": "E BRAND",
                    "status": "I"
                },
                {
                    "__metadata": {
                        "uri": "https://<api-server>/odata/v2/
PickListValueV2(PickListV2_effectiveStartDate=datetime'2020-07-17T00:00:00',PickL
istV2_id='CARBRAND',externalCode='F BRAND')",
                        "type": "SFOData.PickListValueV2"
                    },
                    "externalCode": "F BRAND",
                    "status": "A"
                }
            ]
        }
    }
]

```

```

    }
}

```

### Example 2: Search for a Picklist Whose Parent Picklist Contains a Certain Picklist Value

### Request

Operation	Query
HTTP Method	GET
URI	https://<API-server>/odata/v2/PickList?\$format=JSON&\$filter=parentPickListNav/values/externalCode eq 'ABRAND'&\$expand=values,parentPickListNav,parentPickListNav/values

**Response**

The query returns the picklist whose parent picklist has the said picklist value and a full list of child entity records if you choose to expand association properties.

≡ Sample Code

```
{
  "d": {
    "results": [
      {
        "__metadata": {
          "uri": "https://<api-server>/odata/v2/
PickListV2(effectiveStartDate=datetime'2020-07-01T00:00:00',id='CARCOLOR')",
          "type": "SFOData.PickListV2"
        },
        "id": "CARCOLOR",
        "parentPickList": "CARBRAND",
        "status": "A",
        "values": {
          "results": [
            {
              "__metadata": {
                "uri": "https://<api-server>/odata/v2/
PickListValueV2(PickListV2_effectiveStartDate=datetime'2020-07-01T00:00:00',PickL
istV2_id='CARCOLOR',externalCode='Black')",
                "type": "SFOData.PickListValueV2"
              },
              "externalCode": "Black",
              "status": "A"
            },
            {
              "__metadata": {
                "uri": "https://<api-server>/odata/v2/
PickListValueV2(PickListV2_effectiveStartDate=datetime'2020-07-01T00:00:00',PickL
istV2_id='CARCOLOR',externalCode='Yellow')",
                "type": "SFOData.PickListValueV2"
              },
              "externalCode": "Yellow",
              "status": "A"
            }
          ]
        }
      }
    ]
  }
}
```

```

        "__metadata": {
          "uri": "https://<api-server>/odata/v2/
PickListValueV2(PickListV2_effectiveStartDate=datetime'2020-07-01T00:00:00',PickL
istV2_id='CARCOLOR',externalCode='Orange')",
          "type": "SFOData.PickListValueV2"
        },
        "externalCode": "Orange",
        "status": "A"
      }
    ],
    "parentPickListNav": {
      "__metadata": {
        "uri": "https://<api-server>/odata/v2/
PickListV2(effectiveStartDate=datetime'2020-07-17T00:00:00',id='CARBRAND')",
        "type": "SFOData.PickListV2"
      },
      "id": "CARBRAND",
      "values": {
        "results": [
          {
            "__metadata": {
              "uri": "https://<api-server>/odata/v2/
PickListValueV2(PickListV2_effectiveStartDate=datetime'2020-07-17T00:00:00',PickL
istV2_id='CARBRAND',externalCode='ABRAND')",
              "type": "SFOData.PickListValueV2"
            },
            "externalCode": "ABRAND"
          },
          {
            "__metadata": {
              "uri": "https://<api-server>/odata/v2/
PickListValueV2(PickListV2_effectiveStartDate=datetime'2020-07-17T00:00:00',PickL
istV2_id='CARBRAND',externalCode='B BRAND')",
              "type": "SFOData.PickListValueV2"
            },
            "externalCode": "B BRAND"
          },
          {
            "__metadata": {
              "uri": "https://<api-server>/odata/v2/
PickListValueV2(PickListV2_effectiveStartDate=datetime'2020-07-17T00:00:00',PickL
istV2_id='CARBRAND',externalCode='C BRAND')",
              "type": "SFOData.PickListValueV2"
            },
            "externalCode": "C BRAND"
          },
          {
            "__metadata": {
              "uri": "https://<api-server>/odata/v2/
PickListValueV2(PickListV2_effectiveStartDate=datetime'2020-07-17T00:00:00',PickL
istV2_id='CARBRAND',externalCode='D BRAND')",
              "type": "SFOData.PickListValueV2"
            },
            "externalCode": "D BRAND"
          },
          {
            "__metadata": {
              "uri": "https://<api-server>/odata/v2/
PickListValueV2(PickListV2_effectiveStartDate=datetime'2020-07-17T00:00:00',PickL
istV2_id='CARBRAND',externalCode='E BRAND')",
              "type": "SFOData.PickListValueV2"
            },
            "externalCode": "E BRAND"
          },
          {
            "__metadata": {

```

```

        "uri": "https://<api-server>/odata/v2/  

PickListValueV2(PickListV2_effectiveStartDate=datetime'2020-07-17T00:00:00',PickL  

istV2_id='CARBRAND',externalCode='FBRAND')",  

        "type": "SFOData.PickListValueV2"  

    },  

    "externalCode": "FBRAND"  

}  

]  

}  

}  

]  

}  

}

```

## Related Information

\$filter

### 6.5.2.2.7 Retrieving Workflow Information for Pending Data

You can use the `wfRequestNav` navigation property and the `recordStatus` parameter to retrieve workflow information for pending data of MDF entities.

When an MDF Generic Object (GO) with workflow routing configured is exposed to OData, a navigation property named `wfRequestNav` is automatically added to the OData API entity. This navigation property allows you to retrieve the corresponding workflow information for pending data. Although `wfRequestNav` is available by default for all MDF OData API entities with a workflow routing, you can only expand it if the *Pending Data* option is enabled in the object definition.

i Note

To enable workflow and pending data, associate the MDF generic object with a workflow and set the *Pending Data* field to **Yes** in *Configure Object Definitions*.

Child entities in a composite association don't have the `wfRequestNav` property.

Since MDF OData API only supports workflow information on pending data, you must add the `recordStatus` parameter to specifically filter the pending records, and then expand the workflow information. The following examples show how to do this.

You can use `&recordStatus=pending` to filter all pending records of the entity, and then expand the workflow information:

```
https://<API-endpoint-URL>/odata/v2/cust_MDFEntity?  
$format=JSON&recordStatus=pending&$expand=wfRequestNav
```

If you use a key predicate to query a single record, even if it's a pending record, you still need `&recordStatus=pending` to be able to expand the workflow information:

```
https://<API-endpoint-URL>/odata/v2/cust_MDFEntity('CC01')?
$format=JSON&recordStatus=pending&$expand=wfRequestNav
```

## Error Codes

Error Code	Description
COE_UNSUPPORTED_FEATURE	<p>[COE0025]Unsupported feature: Expand workflow navigation from normal data is NOT supported.</p> <p>If you try to expand wfRequestNav on a normal entry, this error occurs. wfRequestNav can only be expanded if it's a pending entry.</p>

## Related Information

[recordStatus](#) [page 140]

### 6.5.2.2.8 Querying Effective-Dated Entities Using lastModifiedDateTime

You can use the lastModifiedDateTime field in a \$filter option to query the records of an effective-dated entity.

The value of lastModifiedDateTime in a request URI is in date/time format without timezone information:

**datetime**'yyyy-mm-ddThh:mm:ss'

For example: `$filter=lastModifiedDateTime gt datetime'2019-09-01T12:00:00'`.

You can use lastModifiedDateTime together with other time parameters such as fromDate, toDate, and asOfDate. You can also reverse the \$filter condition by placing the date/time value before lastModifiedDateTime to get different query results.

## Using lastModifiedDateTime with Time Parameters

When you use both lastModifiedDateTime and a date parameter in the same request, the following rules apply:

- The lastModifiedDateTime filter finds all **external keys** that have at least one record satisfies the filter condition.

- The time parameter then filters all **records** under the external keys and returns the ones that satisfy the condition in the result.

This means that in the returned result, the `lastModifiedDateTime` value may or may not satisfy the filter condition in the request. The following example explains how it works:

#### ❖ Example

```
https://<API-Server>/odata/v2/EmpJob?$format=JSON&$filter=lastModifiedDateTime  
gt datetime'2017-01-01T00:00:00'&toDate=2019-01-01
```

In this example, you try to find all employees who have their job information changed after 2017-01-01 and pull their entire job history, including the ones that were last modified before 2017-01-01.

## Placing `lastModifiedDateTime` Before or After `DateTime` Value

The position of date/time value in the `lastModifiedDateTime` filter determines how the server interprets the query. When the date/time value is placed after `lastModifiedDateTime`, the filter finds all external keys that satisfy the condition. When the date/time value is placed before `lastModifiedDateTime`, the filter works as you normally understand it: it returns all **records** with last modified date/time satisfying the filter.

In the previous example, if you want to display only the job history that were last modified after 2017-01-01, you can change the query by moving the date/time value before `lastModifiedDateTime` in the filter:

```
https://<API-Server>/odata/v2/EmpJob?$format=JSON&  
$filter=datetime'2017-01-01T00:00:00' gt lastModifiedDateTime&toDate=2019-01-01
```

## Related Information

[Using the DateTime Format](#)

[asOfDate](#) [page 137]

[fromDate](#) [page 138]

[toDate](#) [page 139]

## 6.5.3 Edit Operations

Parameters and examples of edit operations for MDF OData API entities, including insert, upsert, replace, merge, and delete operations.

### MDF Upsert Operation

As of Q4 2019 Release, upsert operations on all MDF OData entities and MDF Foundation Objects can return business keys in the response if the keys are provided in the request. That includes both parent objects and child objects. This information helps you identify which record has been successfully upserted and which has failed.

### 6.5.3.1 Edit Parameters

#### 6.5.3.1.1 workflowConfirmed

Use parameter `workflowConfirmed` to control whether or not the insert or upsert operation triggers a workflow.

### Prerequisites

Either you've defined the workflow routing for the MDF object or you've associated the object with a business rule that triggers a workflow.

The API user doesn't have the [Admin access to MDF OData API](#) permission.

#### ⚠ Caution

The parameter has no impact on admin users with the [Admin access to MDF OData API](#) permission. If an admin user creates or changes a record for a workflow-enabled entity, no workflow is triggered and the record is saved as normal data.

The parameter doesn't work with MDF Foundation Objects.

### What Happens If Two Workflows Are Defined?

If the MDF object is associated with one workflow in the [Workflow Routing](#) field and another workflow through a business rule, the business rule-based workflow supersedes when the rule is triggered.



## ❖ Example

Object `cust_ABC` is associated with two workflows: workflow **Manager** from object definition and workflow **EmployeeHR** from a business rule. When you change data that doesn't trigger the business rule, the **Manager** workflow applies. When you change data that triggers the business rule, the **EmployeeHR** workflow applies.

## Parameter Values

Value	Description
<b>false</b> (default)	A workflow isn't triggered.
<b>true</b>	A workflow is triggered.

## Use Case 1: Upsert a Record With `workflowConfirmed=true`

A typical scenario to use the `workflowConfirmed` parameter is on the UI where you create or edit a record. When you save the record, a dialog pops up for you to submit the changes for approval. If you submit, an API request is sent with `workflowConfirmed=true`:

### Request

Operation	Upsert
HTTP Method	POST
URI	<code>https://&lt;API-Server&gt;/odata/v2/upsert?workflowConfirmed=true</code>
Payload	<pre>{   "__metadata":{     "uri":"cust_TestObject('TO-001') "   },   "externalCode":"TO-001",   "externalName":"Test Object 001" }</pre>

### Response

If the upsert operation is successful, a workflow request is created.

```
{
  "d": [
    {
      "key": null,
      "status": "OK",
      "editStatus": "UPsertED",
      "message": null,
      "index": 0,
      "httpCode": 200,
    }
  ]
}
```

```

    "inlineResults": null
  }
]
}

```

## Use Case 2: Upsert a Record With `workflowConfirmed=false`

As a user without the [Admin access to MDF OData API](#) permission, if you upsert a workflow-enabled entity without the `workflowConfirmed` parameter or set the parameter value to **false**, you get an error:

### Request

Operation	Upsert
HTTP Method	POST
URI	<code>https://&lt;API-Server&gt;/odata/v2/upsert?workflowConfirmed=false</code>
Payload	<pre> {   "__metadata":{     "uri":"cust_TestObject('TO-002') "   },   "externalCode":"TO-002",   "externalName":"Test Object 002" } </pre>

### Response

```

{
  "d": [
    {
      "key": null,
      "status": "ERROR",
      "editStatus": null,
      "message": "To trigger workflow, please set the 'workflowConfirmed'
flag to true. with the index 0",
      "index": 0,
      "httpCode": 500,
      "inlineResults": null
    }
  ]
}

```

## 6.5.3.1.2 warningAcknowledged

Use parameter `warningAcknowledged` to skip warning messages during upsert of an MDF record.

### Prerequisites

A business rule has been defined for the MDF object to raise a warning message.

The API user doesn't have the [Admin access to MDF OData API](#) permission.

#### Note

The parameter has no impact on admin users with the [Admin access to MDF OData API](#) permission. If an admin user creates or changes a record for an entity that triggers a warning message, the warning is skipped.

### Parameter Values

Value	Description
<code>false</code> (default)	If the business rule is triggered, a warning message appears and the upsert fails.
<code>true</code>	No warning message appears even if the business rule is triggered. Upsert is successful.

### Use Case 1: Upsert a Record With `warningAcknowledged=true`

A typical scenario to use the `warningAcknowledged` parameter is on the UI where you create or edit a record. When you save the record, a warning message pops up. If you choose OK, the warning is acknowledged and an API request is sent with `warningAcknowledged=true`.

### Request

Operation	Upsert
HTTP Method	POST
URI	<code>https://&lt;API-endpoint-URL&gt;/odata/v2/upsert&amp;warningAcknowledged=true</code>

Payload

```
{
  "__metadata":{
    "uri":"cust_TestObject('TO-001') "
  },
  "externalCode":"TO-001",
  "externalName":"Test Object 001"
}
```

## Response

If the upsert is successful, a record is created.

```
{
  "d": [
    {
      "key": null,
      "status": "OK",
      "editStatus": "UPsertED",
      "message": null,
      "index": 0,
      "httpCode": 200,
      "inlineResults": null
    }
  ]
}
```

## Use Case 2: Upsert a Record With `warningAcknowledged=false`

As a user without the [Admin access to MDF OData API](#) permission, if you upsert a warning-enabled entity without the `warningAcknowledged` parameter or set the parameter value to **false**, you get an error:

## Request

Operation	Upsert
HTTP Method	POST
URI	<code>https://&lt;API-endpoint-URL&gt;/odata/v2/upsert&amp;warningAcknowledged=false</code>
Payload	<pre>{   "__metadata":{     "uri":"cust_TestObject('TO-002') "   },   "externalCode":"TO-002",   "externalName":"Test Object 002" }</pre>

## Response

```
{
  "d": [
    {
```

```

        "key": null,
        "status": "ERROR",
        "editStatus": null,
        "message": "<user-defined message text>; To be able to save MDF object
with warning, please set the 'warningAcknowledged' flag to true. with the index 0",
        "index": 0,
        "httpCode": 500,
        "inlineResults": null
    }
}

```

## 6.5.3.2 Sample Operations

### 6.5.3.2.1 Inserting An Entry to A Composite Child Entity

We use PickListValueV2 as the example of a composite child entity and show you how to insert an entry.

#### Use Case: Create a Picklist Value

##### Request

Operation	Insert
HTTP Method	POST
URI	https://<API-server>/odata/v2/PickListValueV2
Payload	<pre> {   "__metadata": {     "uri":       "PickListValueV2(PickListV2_effectiveStartDate=datetime' 1900-01-01T00:00:00',PickListV2_id='OfficeLocation',externalCode='717011')",     "type": "SFOData.PickListValueV2"   },   "externalCode": "717011",   "PickListV2_effectiveStartDate": "/ Date(-2208988800000)/",   "PickListV2_id": "OfficeLocation",   "status": "A",   "label_en_US": "Guangzhou" } </pre>
<b>i Note</b> You must include at least one label property in the request payload.	

##### Response

```
{
```

```

    "d": {
      "__metadata": {
        "uri": "https://<API-server>/odata/v2/
PickListValueV2(PickListV2_effectiveStartDate=datetime'1900-01-01T00:00:00',PickList
V2_id='OfficeLocation',externalCode='717011')",
        "type": "SFOData.PickListValueV2"
      },
      "status": "A",
      "label_en_US": "Guangzhou",
      "PickListV2_id": "OfficeLocation",
      "PickListV2_effectiveStartDate": "/Date(-2208988800000)/",
      "externalCode": "717011"
    }
  }
}

```

### 6.5.3.2.2 Updating Picklist Values Using Upsert with Purge Type Parameters

You can use the `upsert` operation with the `purgeType` parameters to perform a full purge or an incremental update on an entity.

We use `PickListValueV2` as the example of a composite child entity and show you how to fully purge picklist values or update a picklist value of a picklist.

#### Deleting All Picklist Values

##### Request

Operation	Upsert
HTTP Method	POST
URI	<code>https://&lt;API-server&gt;/odata/v2/upsert?purgeType=full</code>
Payload	<pre> {   "__metadata":{     "uri":"PickListV2(effectiveStartDate=datetime'2020-07-17 T00:00:00',id='ExistingPickList') "   },   "status":"A" } </pre>

##### Response

###### Sample Code

```

{
  "d": [
    {
      "key": "PickListV2/
effectiveStartDate=2020-07-17T00:00:00.000-04:00,PickListV2/id=ExistingPickList",

```

```

    "status": "OK",
    "editStatus": "UPsertED",
    "message": null,
    "index": 0,
    "statusCode": 200,
    "inlineResults": null
  }
]
}

```

The picklist now has no picklist values.

## Replacing Old Picklist Values with New Ones

### Request

Operation	Upsert
HTTP Method	POST
URI	https://<API-server>/odata/v2/upsert?purgeType=full
Payload	<pre> {   "__metadata":{     "uri":"PickListV2(effectiveStartDate=datetime'2020-07-17T00:00:00',id='PickListExample') "   },   "status":"A",   "values":[{     "__metadata":{       "uri":"PickListValueV2(PickListV2_effectiveStartDate=datetime'2020-07-17T00:00:00',PickListV2_id='PickListExample',externalCode='NEWVALUE1') "     },     "status":"A",     "label_en_US":"New Value 1"   },   {     "__metadata":{       "uri":"PickListValueV2(PickListV2_effectiveStartDate=datetime'2020-07-17T00:00:00',PickListV2_id='PickListExample',externalCode='NEWVALUE2') "     },     "status":"A",     "label_en_US":"New Value 2"   } ]} </pre>

### Response

#### Sample Code

```

{
  "d": [
    {

```

```

        "key": "PickListV2/
effectiveStartDate=2020-07-17T00:00:00.000-04:00,PickListV2/id=PickListExample",
        "status": "OK",
        "editStatus": "UPsertED",
        "message": null,
        "index": 0,
        "httpCode": 200,
        "inlineResults": null
    }
]
}

```

All previous picklist values of the picklist are replaced with the two new values.

## Updating the Label of an Existing Picklist Value

### Request

Operation	Upsert
HTTP Method	POST
URI	https://<API-server>/odata/v2/upsert?purgeType=incremental or https://<API-server>/odata/v2/upsert
Payload	<pre> {   "__metadata":{     "uri":"PickListV2(effectiveStartDate=datetime'2020-07-17T00:00:00',id='PickListExample') "   },   "status":"A",   "values":[{"__metadata":{     "uri":"PickListValueV2(PickListV2_effectiveStartDate=datetime'2020-07-17T00:00:00',PickListV2_id='PickListExample',externalCode='NEWVALUE1') "   },     "status":"A",     "label_en_US":"New Label"   }] } </pre>

### Response

#### Sample Code

```

{
  "d": [
    {
      "key": "PickListV2/
effectiveStartDate=2020-07-17T00:00:00.000-04:00,PickListV2/id=PickListExample",
      "status": "OK",
      "editStatus": "UPsertED",
      "message": null,
      "index": 0,
      "httpCode": 200,

```



```

    "inlineResults": null
  }
}

```

Only the label of picklist value "NEWVALUE1" is updated.

## Related Information

[Upsert](#)

[Edit Parameter: purgeType](#)

### 6.5.3.2.3 Updating Picklist Values Using Merge

You can use the merge operation to incrementally update a property without providing all required properties in the payload.

We use `PickListValueV2` as the example of a composite child entity and show you how to update picklist values using merge.

#### Request

Operation	Merge
HTTP Method	POST
URI	<pre> https://&lt;API-server&gt;/odata/v2/ PickListValueV2(PickListV2_effectiveStartDate=datetime'2020 -07-17T00:00:00',PickListV2_id='ExamplePicklist',externalCo de='OldValue') </pre>
Headers	x-http-method: MERGE
Payload	<pre> {   "externalCode": "NewValue" } </pre>

#### Response

A successful operation returns status 200 OK with no payload.

## Related Information

[Merge](#)

## 6.5.3.2.4 Replacing Picklist Using Replace

You can use the replace operation to update an existing record with the fields specified in the request payload and resets all other fields to their default value. Unlike the merge operation, you must provide all required fields in the request payload.

We use `PickListV2` as an example to show you how to replace an old picklist with a new one. The example picklist had an old name and its display order was numeric.

### Request

Operation	Replace
HTTP Method	PUT
URI	<code>https://&lt;API-server&gt;/odata/v2/ PickListV2(effectiveStartDate=datetime'2020-07-17T00:00:00',id='OldID')</code>
Payload	<pre>{   "id": "NewID",   "status": "A" }</pre>

### Response

A successful operation returns status 200 OK with no payload. In the updated version, the example picklist has no name and its display order is reset to the default value that is alphabetic.

## Related Information

[Replace](#)

## 6.5.3.2.5 Deleting Picklist or Picklist Value

You can use the delete operation to delete an entry by passing the key predicate in the request URI.

We use `PickListV2` and `PickListValueV2` as examples to show you how to delete an entry.

### Request

Deleting a Picklist

Operation	Delete
HTTP Method	DELETE

URI	<code>https://&lt;API-server&gt;/odata/v2/ PickListV2(effectiveStartDate=datetime'2020-07-17T00:00:00', id='ExamplePicklist')</code>
Deleting a Picklist Value	
Operation	Delete
HTTP Method	DELETE
URI	<code>https://&lt;API-server&gt;/odata/v2/ PickListValueV2(PickListV2_effectiveStartDate=datetime'2020-07-17T00:00:00', PickListV2_id='ExamplePicklist',externalCode='ExampleValue')</code>

## Response

A successful delete operation returns status 200 OK with no response payload. The picklist and picklist value specified in the URI are deleted from the system.

## 6.5.3.2.6 Creating Links for MDF Entities

You can use the `$links` option to create links for MDF entities.

### Rules for Creating Links for MDF Entities

The following rules apply when you create links between MDF entities (including MDF Foundation Objects):


- You can't create links for MDF Generic Objects with composite type associations.
- A new link is always inserted for a 1:n association. If the same link already exists, the POST operation returns `204 No Content` without data change.
- For 1:1 relationship, if a link already exists, the POST operation fails.

### Adding a Parent Picklist Using Links

We use `PickListV2` as the example to show you how to create links for MDF entities. You can't create a new picklist value by using the `$links` option because the association between `PickListV2` and `PickListValueV2` is composite.

## Request

HTTP Method	POST
-------------	------

URI	<pre>https://&lt;API-server&gt;/odata/v2/ PickListV2(effectiveStartDate=datetime'2020-07-17T00:00:00', id='ChildPicklist')/\$links/parentPickListNav</pre>
Payload	<div>  Sample Code </div> <pre>{   "uri": "PickListV2(effectiveStartDate=datetime'2020-07-10T00:00:00',id='ParentPickList')"</pre>

## Response

A successful operation returns status 204 (No Content) with no payload. In the updated version, the child picklist is assigned with a parent picklist.

## Related Information

[Working with Links](#)

### 6.5.3.2.7 Updating Parent Picklist Value Using Links

You can use the `$links` option to update links of noncomposite associations.

We use `PickListValueV2` as the example and updates the parent picklist value.

#### Note

You can't update picklist values of a picklist using the `$links` option because the association type between `PickListV2` and `PickListValueV2` is composite.

## Request

HTTP Method	PUT
URI	<pre>https://&lt;API-server&gt;/odata/v2/ PickListValueV2(PickListV2_effectiveStartDate=datetime'2020-07-17T00:00:00',PickListV2_id='ChildPicklist',externalCode='ExampleValue')/\$links/parentPickListValueNav(PickListV2_effectiveStartDate=datetime'2020-07-10T00:00:00',PickListV2_id='ParentPicklist',externalCode='OldValue')</pre>

Payload

#### Sample Code

```
{
  "uri": "PickListV2(PickListV2_effectiveStartDate=datetime'2020-07-10T00:00:00',PickListV2_id='ParentPicklist',externalCode='NewValue')"
```

## Response

A successful operation returns status 204 (No Content) with no payload. In the updated version, the value of the child picklist is assigned with a new value from the parent picklist.

## Related Information

[Working with Links](#)

## 6.5.3.2.8 Uploading Attachments to an MDF Entity

How to upload an attachment to an MDF OData API entity.

## Context

The attachment navigation property in an MDF entity doesn't allow you to inline edit attachments. To upload an attachment to an MDF entity, you first create the attachment object with the [Attachment](#) entity, and then add this object to the MDF entity.

### Note

You're not allowed to add the same attachment ID to multiple objects. If you want to add the same attachment to another object, upload it again to create a new attachment ID.

## Procedure

1. Create an attachment object with the [Attachment](#) entity.

Sample Request:

Operation	Upsert
HTTP Method	POST
URI	https://<API-Server>/odata/v2/upsert
Payload	<pre>{   "_metadata": {     "uri": "Attachment"   },   "fileName": "Resume.pdf",   "module": "RECRUITING",   "userId": "cgrant",   "viewable": true,   "fileContent": "PHA +VGhp...KSHlsHcq9JS" }</pre>

Sample Response:

```
{
  "d": [
    {
      "key": "Attachment/attachmentId=499",
      "status": "OK",
      "editStatus": "UPsertED",
      "message": "Upserted successfully",
      "index": 0,
      "httpCode": 200,
      "inlineResults": null
    }
  ]
}
```

2. Create a new entry or update an existing entry for the MDF entity with the new attachment created in the previous step.

Sample Request:

Operation	Upsert
HTTP Method	POST
URI	https://<API-Server>/odata/v2/upsert

Payload

```
{
  "__metadata": {
    "uri": "cust_TestMDFObject"
  },
  "externalCode": "test entry 01",
  "effectiveStartDate": "/Date(1565222400000)/",
  "externalName": "Test MDF Entry 01",
  "cust_attachmentNav": {
    "__metadata": {
      "uri": "Attachment(499L)"
    }
  }
}
```

### 6.5.3.2.9 Creating Self-Referenced Record Using Upsert

You can use the `upsert` operation to create a self-referenced record for custom entities whose effective dating type isn't MCPD.

#### Example Entity

We use `cust_EntityP` as the example entity.

- `cust_EntityP` is a custom entity.
- `cust_EntityP` has a *Generic Object* field `cust_goNav` and the *Valid Value Source* is `cust_EntityP` itself.
- The effective dating type of `cust_EntityP` isn't MCPD.

#### Request

Operation	Upsert
HTTP Method	POST
URI	<code>https://&lt;API-server&gt;/odata/v2/upsert</code>

Payload

```
{
  "__metadata": {
    "uri":
    "cust_EntityP(effectiveStartDate=datetime'2020-07-17T00:00:00',externalCode='ValueA')"
  },
  "cust_goNav": {
    "__metadata": {
      "uri":
      "cust_EntityP(effectiveStartDate=datetime'2020-07-17T00:00:00',externalCode='ValueA')"
    },
    "externalCode": "ValueA"
  }
}
```

## Response

Sample Code

```
{
  "d": [
    {
      "key": "cust_EntityP/
effectiveStartDate=2020-07-17T00:00:00Z,cust_EntityP/externalCode=ValueA",
      "status": "OK",
      "editStatus": "UPSERVED",
      "message": null,
      "index": 0,
      "httpCode": 200,
      "inlineResults": null
    }
  ]
}
```

## Wrong Method to Create Self-Referenced Record Using Insert

You can't create self-referenced record using insert.

### Request

Operation	Insert
HTTP Method	POST
URI	https://<API-server>/odata/v2/cust_EntityP



Payload

```
{
  "__metadata": {
    "uri":
    "cust_EntityP(effectiveStartDate=datetime'2020-07-17T00:00:00',externalCode='ValueA')"
  },
  "cust_goNav": {
    "__metadata": {
      "uri":
      "cust_EntityP(effectiveStartDate=datetime'2020-07-17T00:00:00',externalCode='ValueA')"
    },
    "externalCode": "ValueA"
  }
}
```

## Response

Sample Code

```
{
  "error": {
    "code": "COE_GENERAL_BAD_REQUEST",
    "message": {
      "lang": "en-US",
      "value": "[COE0018]Record '(effectiveStartDate=2020-07-17,externalCode=ValueA)' of type 'cust_EntityP' already exists."
    }
  }
}
```

### 6.5.3.2.10 Upsert Self-Referenced Entity with Different Field Value in Payload

Refer to the sample operation for how the system handles different values of the self-referenced field in payload.

## Example Entity

We use `cust_EntityP` as the example entity.

- `cust_EntityP` is a custom entity.
- `cust_EntityP` has a *Generic Object* field `cust_goNav` and the *Valid Value Source* is `cust_EntityP` itself.
- The effective dating type of `cust_EntityP` isn't MCPD.

## Request

Operation	Upsert
HTTP Method	POST
URI	https://<API-server>/odata/v2/upsert
Payload	<pre>{   "__metadata": {     "uri":     "cust_EntityP(effectiveStartDate=datetime'2020-07-17T00:00:00',externalCode='ValueA') "   },   "name": "A",   "cust_goNav": {     "__metadata": {       "uri":       "cust_EntityP(effectiveStartDate=datetime'2020-07-17T00:00:00',externalCode='ValueA') "     },     "name": "B"   } }</pre>

## Response

The *name* of cust\_EntityP's record **ValueA** is updated as **A**.

### Sample Code

```
{
  "d": [
    {
      "key": "cust_EntityP/
effectiveStartDate=2020-07-17T00:00:00Z,cust_EntityP/externalCode=ValueA",
      "status": "OK",
      "editStatus": "UPSERTEDED",
      "message": null,
      "index": 0,
      "httpCode": 200,
      "inlineResults": null
    }
  ]
}
```

## 6.5.3.2.11 Upsert Handling Different Field Values in Payload

Refer to the sample operation for how the system handles different values of the same *Generic Object* field in payload.

### Example Entity

We use `cust_EntityP` as the example entity.

- `cust_EntityP` is a custom entity.
- `cust_EntityP` has a *Generic Object* field `cust_goNav` and the *Valid Value Source* is `cust_EntityC`.
- The effective dating type of both `cust_EntityP` and `cust_EntityC` isn't MCPD.

### Request

Operation	Upsert
HTTP Method	POST
URI	<code>https://&lt;API-server&gt;/odata/v2/upsert</code>

Payload

```
[
  {
    "__metadata": {
      "uri":
"cust_EntityP(effectiveStartDate=datetime'2020-07-17T00:00:00',externalCode='ValueA')"
    },
    "cust_goNav": {
      "__metadata": {
        "uri":
"cust_EntityC(effectiveStartDate=datetime'2020-07-17T00:00:00',externalCode='ValueB')"
      },
      "name": "Name A"
    }
  },
  {
    "__metadata": {
      "uri":
"cust_EntityP(effectiveStartDate=datetime'2020-07-17T00:00:00',externalCode='ValueA')"
    },
    "cust_goNav": {
      "__metadata": {
        "uri":
"cust_EntityC(effectiveStartDate=datetime'2020-07-17T00:00:00',externalCode='ValueB')"
      },
      "name": "Name B"
    }
  }
]
```

## Response

The *name* of cust\_EntityC's record **ValueB** is updated as **Name B**.

### Sample Code

```
{
  "d": [
    {
      "key": "cust_EntityP/
effectiveStartDate=2020-07-17T00:00:00Z,cust_EntityP/externalCode=ValueA",
      "status": "OK",
      "editStatus": "UPserted",
      "message": null,
      "index": 0,
      "httpCode": 200,
      "inlineResults": null
    },
    {
      "key": "cust_EntityP/
effectiveStartDate=2020-07-17T00:00:00.000Z,cust_EntityP/externalCode=ValueA",
      "status": "OK",
      "editStatus": "UPserted",
      "message": null,
      "index": 1,
      "httpCode": 200,
      "inlineResults": null
    }
  ]
}
```

```
}  
  ]  
}
```

# 7 API Center

Describes the features of the API Center and required permissions .

The API center is a one-stop shop for accessing OData API tools. You'll find the API Center in the Admin Center. If you can't see it there, check that you have the permission for at least one of the tools hosted on the API Center. For more information, see [Permissions for API Center \[page 182\]](#).

## i Note

Users can only access tools for which they have permission. Otherwise the tools are displayed but they are greyed out.

## Related Information

[Enabling API Audit Logs \[page 185\]](#)

[Creating and Editing Users Using an API Option Profile \[page 189\]](#)

[OData API Data Dictionary \[page 183\]](#)

[Restricting OData API Access through Basic Authentication \[page 12\]](#)

[OData API Metadata Management \[page 188\]](#)

[Permissions for API Center \[page 182\]](#)


[SFAPI Metering Details \[page 193\]](#)

## 7.1 Permissions for API Center

A list of role-based permissions required to access *API Center* tools.

The following role-based permissions are available under ► *Administrator Permissions* ► *Manage Integration Tools* ► for *API Center* tools:

Permission	Description
Access to SFAPI Audit Log	Access the SFAPI Audit Log.

 **Caution**  
This permission allows user to view data sent in every SFAPI call. The data could be sensitive.

Permission	Description
Access to SFAPI Metering Details	Access the SFAPI Metering Details to see SFAPI usage statistics
Access to SFAPI Data Dictionary	Access SFAPI Data Dictionary.
Manage OAuth2 Client Applications	Manage OAuth2 client applications
Access to OData API Audit Log	Access to the OData API audit log tool
Manage OData API Basic Authentication	Manage OData API Basic Authentication settings
Access to OData API Version Control	Check the current OData API version and switch versions
Access to API Center	Access to <a href="#">API Center</a>
Access to API Option Profile	Access the API option profile tool to manage processing parameters for the User entity.
Access to OData API Metadata Refresh and Export	Access the OData API Metadata Refresh and Export tool.
Access to OData API Data Dictionary	Access the OData API Data Dictionary tool.

## 7.2 Data Dictionary

### 7.2.1 OData API Data Dictionary

Describes the features, required permissions, and performance tips for the OData API Data Dictionary

In the OData API data dictionary, you can see information about attributes and supported operations.

The look and feel of the OData API data dictionary has been enhanced. It's available in the Admin Center. If you can't see it there, check that you have the permission, [Access to OData API Dictionary](#) (available in [Manage Integration Tools](#) and applicable to both RBP and non-RBP users).

#### i Note

To optimize performance, you can request that Customer Services activate the provisioning switch, [Enable OData API Dictionary Cache](#) (in [Web Services](#)).

In the OData API data dictionary, you can use [Search All](#), or narrow your search to [Entity](#), [Complex Type](#), or [Function Import](#).

Using the search filters [Entity](#), [Complex Type](#), or [Function Import](#) also lets you narrow your search by [Tag](#), for example EC - Payment Information . This then ensures that under [Name](#), you only see the entities available for this tag.

Information is displayed in a tabular form. A horizontal and vertical scroll bar are available to make viewing easier. The following columns are available and can be sorted alphabetically in ascending or descending order:

- [Property Name](#)  
In addition to sorting alphabetically, you can also filter by a defined term.
- [Property Label](#)
- [Type](#)

These columns are fixed. However, the following columns can be arranged to meet your requirements:

- [sap.picklist](#)
- [Business Key](#)
- [Nullable](#)
- [sap: required](#)
- [sap: creatable](#)
- [sap: updatable](#)
- [sap: upsertable](#)
- [sap:visible](#)
- [sap: sortable](#)
- [sap: filterable](#)
- [sap: semantics](#)
- [sap: display-format](#)
- [sap: aggregation role](#)
- [DefaultValue](#)
- [MaxLength](#)
- [Precision](#)
- [Scale](#)
- [sap: inlineRequired](#)
- [Navigation Target](#)

## 7.2.2 Legacy SFAPI Data Dictionary

Lists all available SFAPI entities and the fields in each entity.

You can use the [Legacy SFAPI Data Dictionary](#) tool to look up SFAPI entities, supported operations, and field properties.

### i Note

In Admin Center, the tool is called [SFAPI Data Dictionary](#), but in API Center, it is called [Legacy SFAPI Data Dictionary](#).

To access the dictionary, you must have the [Admin Access to SFAPI Data Dictionary](#) permission under [Manage Integration Tools](#) available in both user-based and role-based permission systems.

The SFAPI Data Dictionary lists all available SFAPI entities in your company. You can expand any entity to see the supported operations, field names, and detailed attributes for each field in a table. The following attributes are available for an SFAPI entity:

- [Name](#)
- [Data Type](#)



- [Label](#)
- [Max Length](#)
- [Picklist ID](#)
- [Required](#)
- [Insertable](#)
- [Upsertable](#)
- [Updateable](#)
- [Seletable](#)
- [Filterable](#)
- [Sortable](#)
- [Sensitive](#)
- [Supported Operators](#)
- [Enumerations](#)

## 7.3 Enabling API Audit Logs

Enable API audit logs in [API Center](#) so you can view and download API transaction history for troubleshooting API issues.

### Prerequisites

You have the following permissions:

- [Manage Integration Tools](#) > [Access to API Center](#) >
- [Manage Integration Tools](#) > [Access to OData API Audit Log](#) > for viewing OData API audit logs.
- [Manage Integration Tools](#) > [Access to SFAPI Audit Log](#) > permission for viewing SFAPI audit logs.

#### Note

The audit log can contain sensitive data. Make sure that the access to API audit logs is given only to trusted users. As a safety strategy, grant permission to developers only when necessary for development and debugging purposes, and then remove access when the tasks are finished. Note that removing a user's access to the audit log doesn't remove the data from the server.

### Context

API Audit logs include API request and response information, including payloads. This information help developers diagnose issues happened during an API call.

## Note

A maximum log count is set for your company. This number varies from 10,000 to 500,000 depending on the Provisioning setting. When the limit is reached, older entries are purged.

Additionally, audit data in preview instances is subject to the retention time defined in [Impact of Audit Data Purge in Preview Instances](#). Any audit data that meets either criteria is purged in preview instances.

## What Information is Logged?

Depending on the operation type and status, the information kept in the audit log varies. The following tables show what information is logged in SFAPI and OData API audit logs:

OData Audit Log

Condition		Result (Y = Logged, N = Not Logged)			
Operation Type	Operation Status	Request Header	Request Payload	Response Header	Response Payload
Query	Success	Y	N	Y	N
	Failure	Y	N	Y	N
Edit	Success	Y	N	Y	N
	Failure	Y	Y	Y	Y
	Including partial failure in \$batch and upsert				

Legacy SFAPI Audit Log

Condition		Result (Y = Logged, N = Not Logged)			
Operation Type	Operation Status	Request Header	Request Payload	Response Header	Response Payload
Query	Success	Y	N	Y	N
	Failure	Y	N	Y	N
Edit	Success	Y	Y	Y	Y
	Failure	Y	Y	Y	Y

## Procedure

1. Go to [Admin Center](#) > [API Center](#) > [Audit Log Settings](#) and turn on audit logs for SFAPI or OData API using the following options:

Option	Description
OData API Audit Log	Switch it on if you want to enable audit log for OData API.

Option	Description
Enable Payloads in OData API Audit Log for Edit Errors	Enable this option if you want to log request and response payloads for failed edit operations of OData API.
Enable All Payloads	If you choose this option, all payloads of query and edit operations for OData API are logged for the next 4 hours, regardless of whether it's successful or not.
SFAPI Audit Log	Switch it on if you want to enable audit log for SFAPI.
Enable Payloads in SFAPI Audit Log for Edit Operations	Enable this option if you want to log request and response payloads for all edit operations for SFAPI.
Enable Payloads in SFAPI Audit Log for Edit Operations	If you choose this option, all payloads of query and edit operations of SFAPI are logged for the next 4 hours, regardless of whether it's successful or not.

Audit log is enabled for SFAPI or OData API.

2. Go to ► [Admin Center](#) ► [API Center](#) ► and choose one of the following tools:
  - [OData API Audit Log](#): View OData API transaction history and payloads.
  - [Legacy SFAPI Audit Log](#): View SFAPI transaction history and payloads.
3. On the audit log screen, use the filter section to filter or search for the transaction history. For each log, you can see the following information:
  - [Log ID](#) (filterable, text only)
  - [Login ID](#) (filterable, text only)
  - [Session ID](#)
  - [Request ID](#)
  - [Status](#)
  - [OData API Call](#)
  - [Entity](#)
  - [Request Time](#)
  - [Responses \(ms\)](#) (filterable)

The response filter supports the following operations:

  - Equal to (=)
  - Greater than (>)
  - Less than (<)
  - Greater than or equal to (>=)
  - Less than or equal to (<=)
  - Not equal to (!=)
  - Range (..). Example: filter 100..200 means >=100 and <=200.

- [HTTP Message](#) (downloadable audit log detail with header and payload information)
4. On the [HTTP Message](#) column, choose .... to display log details.

The detail screen contains request and response header information, and payload information if it's enabled. Switch tabs to view detailed information.

Request header:

```
GET https://<API-Server>/odata/v2/checkUserPermission?
%24format=json&accessUserId=
%27extuser133%27&company=<CompanyID>&permLongValue=-1L&permStringValue=
%27cubetree_access%27&permType=%27cubetree%27
ip address: <Server IP>
host: <API-Server>
x-forwarded-for: <Server IP>
x-forwarded-port: 9003
x-forwarded-proto: http
authorization: Bearer *****
accept-encoding: gzip;q=1.0,deflate;q=0.6,identity;q=0.3
accept: */*
user-agent: Ruby
x-forwarded-host: <API-Server>
```

Response header:

```
X-Event-ID: EVENT-UNKNOWN-UNKNOWN-or3areboh40s-20200330013237-191176
CorrelationId: 2febeba5-963d-4f5e-b0b4-1ef2b3566878
Content-Type: application/json;charset=utf-8
DataServiceVersion: 1.0
Vary: Accept-Encoding
Content-Encoding: gzip
SFODataServerTimeZone: America/New_York
Status: 200
```

5. (Optional) Choose [Download](#) on each tab to download the log.

## 7.4 OData API Metadata Management

You can use the OData API Metadata Management tool to refresh metadata cache and export OData API metadata.

To view OData API Metadata Refresh and Export, select the [OData API Metadata Management](#) link in [API Center](#).

You will see two buttons:

- [Refresh](#)
- [Export](#)

Refresh and Export operations are described below:

Operation	Description
Refresh	Refreshes the metadata cache. The updated metadata is re-generated, and saved into the cache and is available for future operations such as export.
Export	Exports and downloads the metadata. The default export file name is <code>CompanyName-Metadadata.xml</code>

In most cases, refresh is automatic, for example when an admin changes or creates an object definition in the Extension Center or in the Configure Object Definition page or when an Admin enables/disables a feature in the Upgrade Center leading to a change that requires a refresh of OData metadata.

Use the Refresh option whenever entities or fields are not reflecting the latest changes made to them.

#### **i Note**

A manual refresh is always required when a user with provisioning access chooses to enable/disable a product feature or change the language packs via provisioning leading to a corresponding OData API change. In this case, the change will only be effective after a manual refresh.

## **7.5 Creating and Editing Users Using an API Option Profile**

You can use the [Manage API Option Profile](#) tool to maintain API option settings and manage API option profiles for the `User` entity.

### **Prerequisites**

You have the [Manage Integration Tools](#) > [Access to API Option Profile](#) permission assigned to your role.

### **Context**

The `User` entity allows you to specify a few processing parameters to control extra business logic that can be optionally applied when you edit a user. For example, if you enable the [Process inactive employees](#) option in a profile. When you use this profile in a User API request, you're able to modify the information of an inactive user. The options are stored in API option profiles. You can choose different options to enable in different profiles.

### **Procedure**

1. Go to [Admin Center](#) and search for [Manage API Option Profile](#).
2. On the [List API Option Profile](#) screen, a list of existing profiles is displayed. You can manage the list by editing, deleting, or adding new profiles. Choose [Add](#) to create a new profile.
3. On the [Add API Option Profile](#) screen, select [User](#) from the [Entity Type](#) dropdown list and enter a profile ID and a description.

#### **i Note**

Currently the OData API Option Profile tool supports only the User OData entity.

- The corresponding options to the selected entity are displayed on the right. Choose the ones you want to enable for this profile.

Option	Description
Use the user name	Choose the default password format for creating users. If no option is specified, the system uses user name as the default password option.
Use the UserID	
Use the e-mail address	
Use a system-generated random password	Starting from Jun 19, 2020, you can only choose the system-generated password option when you create new profiles. Existing profiles that use user name, user ID, and e-mail as default password values continue to work.

*Default password format*

**⚠ Caution**

We encourage you to review your existing API calls and stop using unsafe password options. You can either edit your API option profiles and change the password option to [Use a system-generated random password](#) or enable the [Use System Generated Password by Default](#) option in [Admin Center](#) > [Platform Feature Settings](#) to force all API calls to use the safe password option.

Users in instances provisioned after Aug 14 are required to log in to the system and reset their passwords after they are set by system administrators.

Option	Description
<a href="#">Send welcome email to new users</a>	<p>Choose this option if you want the system to automatically send welcome e-mails to users when they're created. If you choose system-generated passwords when creating users, this option is required so users can reset their passwords through a link in the e-mail.</p> <div data-bbox="1011 512 1409 907"> <p><b>i Note</b></p> <p>Welcome e-mails can't be sent to rehired employees. For security reasons, e-mails can only be sent to corporate e-mail accounts.</p> <p>This option is ignored and e-mail notifications are sent to new users by default if the <a href="#">Send Welcome Message</a> option is enabled in <a href="#">Admin Center</a> &gt; <a href="#">Platform Feature Settings</a>.</p> </div>
<a href="#">Validate Manager and HR fields</a>	<p>Choose this option if you want to validate if the manager and HR fields of the user are valid.</p>
<a href="#">Process inactive Employees</a>	<p>This option allows you to edit employees in inactive status. You can also use parameter <code>processInactiveEmployees</code> for the same purpose.</p> <div data-bbox="1011 1264 1409 1402"> <p><b>i Note</b></p> <p>The Effective Dated Data Platform feature must be enabled for your instance.</p> </div> <p>See <a href="#">User</a> for more information.</p>
<a href="#">Specify Form routing options</a>	<p>This option adds the new manager as the next person on the approval chain to receive the form. Choose this if you want the new manager to be a part of the review process and remove the old manager from the review process.</p>
<p><a href="#">Automatic Manager Transfer</a></p> <p>Automatic insertion of new manager as next document recipient if not already</p> <p>For each option, you can specify if you want to transfer the document to manager, matrix manager, or both.</p>	<p>This option adds the new manager as the next person on the approval chain to receive the form. Choose this if you want the new manager to be a part of the review process and remove the old manager from the review process.</p>

Option	Description
Automatic Inbox Document Transfer To New Manager	Choose this option if you want to move all forms from the old manager's inbox to the new manager's inbox. Forms in all other folders, such as en route, aren't transferred.
Automatic En Route Document Transfer To New Manager	Choose this option if you want to move en-route forms from the old manager to the new manager. Use this option for transferring 360 forms.
Automatic Completed Document Copy to New Manager	Choose this option if you want to move all completed forms of the employee to the new manager's Completed folder. All other forms aren't transferred.
Automatic Process Owner Change To New Manager For In-Progress Documents When Old Manager is Process Owner (Only for 360)	This option is currently not supported in OData API.
Automatic Process Owner Change To New Manager For Completed Documents When Old Manager is Process Owner (Only for 360)	This option is currently not supported in OData API.
<i>Automatic Document Removal</i>	Remove Inactive Employees' In-Progress Documents
	Choose this option if you want to delete forms from the inbox of inactive users.
	Remove Inactive Employees' Completed Documents
	Choose this option if you want to delete completed forms for inactive users.
	Remove Inactive Employees' 360 Evaluation Documents
	Choose this option if you want to delete 360 participant evaluation forms for inactive users.

5. Add parameter `apiOptionProfileID=<profile ID>` to your URI when you insert, update, or upsert a user.

Operation	Insert
HTTP Method	POST
URI	<code>https://&lt;API-Server&gt;/odata/v2/User?apiOptionProfileID=ID01</code>



Payload

```
{
  "_metadata":{
    "uri":"User('acraig')",
    "type":"SFOData.User"
  },
  "userId":"acraig",
  "username":"acraig",
  "status":"t",
  "firstName":"Amy",
  "lastName":"Craig",
  "email":"amy.craig@bestrun.com"
}
```

### ⚠ Caution

Multiple parameter values aren't supported. If you include multiple values of the same parameter, for example, `apiOptionProfile=ID01&apiOptionProfile=ID02`, only the first parameter value is processed.

## Next Steps

We recommend all users log in to the system and reset their passwords to ensure account security.

## 7.6 SFAPI Metering Details

The SFAPI Metering Details tool gives you analytics on your API usage up to the last 30 days. You can use this page to see API call history analytics like how many times the API was called, or what was the total record counts accessed in your system.

Select one of the following dimensions (views) from the drop-down list to display the API call history:

- Call Count
- Processing Time (Avg)
- Processing Time (Total)
- Bandwidth (Avg)
- Bandwidth (Total)
- Bandwidth (Request Avg)
- Bandwidth (Request Total)
- Bandwidth (Response Avg)
- Bandwidth (Response Total)
- Record Count (Avg)
- Record Count (Total)
- Record Count (Request Avg)
- Record Count (Request Total)

- Record Count (Response Avg)
- Record Count(Response Total)

You can choose to display the API call history in the following time periods:

- 6 hours
- 12 hours
- 24 hours
- 2 days
- 4 days
- 7 days
- 15 days
- 30 days

## 8 Metadata

This section describes the OData v2 metadata in SAP SuccessFactors HXM Suite.

The OData metadata is a static resource that describes the data model and type system understood by that particular OData service. You can use the metadata to learn how to query and navigate between the entities in the system. Metadata extensions provide additional metadata information, which gives you access to advanced operations such as data retrieval filtration.

### Cache-Control for OData API \$metadata Operations

Entity Tag (ETag) is used to check if the metadata saved in client cache is the same as the one on the server. The value of max-age in the `Cache-Control` header is set to the lifecycle of the client metadata cache. If the cache is valid, no new request is sent to the server. When a client raises a request for metadata the first time, the server sends back a response with the latest metadata, along with a response header named ETag. The value of ETag is unique and matches the metadata version. This value is used for "If-None-Match" in the request header the next time the same request is raised. The server checks the "If-None-Match" value when a new request arrives. If its value is the same as the latest ETag generated by the server, the server simply sends back a status code of 304 (Not-Modified) instead of resending the entire metadata.

### Metadata and the API

The OData metadata describes the capabilities of the API in your SAP SuccessFactors HXM Suite instance. It contains the details of each entity that is accessible through the API, including fields, their names and labels, their data types, and the relationships (associations) between entities.

The OData metadata also describes the operations available in the API. The OData protocol specifies four basic database style operations: query, insert, update, and delete. SAP SuccessFactors has added a custom operation called "upsert" which performs an insert or update operations. Typically custom operations perform specific business transactions, especially if a custom API is easier to manage versus a database style approach against multiple entities.

### Development Using OData Metadata and API

Regardless of which operations are used, SAP SuccessFactors HXM Suite applies the appropriate business logic for each entity type. In other words, even though the operations appear to be database-centric, the API goes through the business logic in the application layer. The API doesn't go directly against the database, nor does it bypass the business logic layer. The entities in the API represent logical application objects familiar to an application user. Note that the entities don't represent the actual physical data storage implementation, which can be in a different structure.

Using the metadata for customized development is optional and considered advanced behavior for API clients. It can be critical to API client systems that need to write general code, which automatically adjusts to the system configuration. For example, if you're writing a middleware tool that allows runtime discovery of the SAP SuccessFactors HXM Suite OData system, you can use the metadata to discover the entities and fields, and their data types.

## 8.1 Retrieving Metadata

Learn how to retrieve OData metadata in SAP SuccessFactors HXM Suite.

There are two ways to retrieve OData metadata in SAP SuccessFactors HXM Suite:

- The *OData API Data Dictionary* tool in *API Center* provides a user-friendly view of OData metadata. You can use the dictionary to look up EntitySets, Complex Types, and Function Imports. The dictionary also includes all properties, navigations, and supported operations of the APIs.
- You can also use API queries to retrieve OData metadata or a subset of the metadata from the API server.

### Retrieving Metadata Using `$metadata`

You can use the `$metadata` system query option to query the entire metadata document or the metadata document for a subset of entity types.

A metadata request using `$metadata` returns an XML serialization of the service, including the entity data model (EDM) and the service operation descriptions. The metadata response supports only `application/atom+xml` type.

- The following URI retrieves the entire OData metadata document:

```
https://<API-Server>/odata/v2/$metadata
```

- You can also request part of the metadata document by specifying a subset of entity types:

```
https://<API-Server>/odata/v2/<Entity1>,<Entity2>...<EntityN>/$metadata
```

For example, the following URI identifies the metadata of the User entity type:

```
https://<API-Server>/odata/v2/User/$metadata
```

The following query returns the metadata of the User and Photo entity types:

```
https://<API-Server>/odata/v2/User,Photo/$metadata
```

### Specifying the Language for `sap:label`

The default language for all property labels (`sap:label`) in the metadata document is English (en-US). You can request the metadata in different languages by specifying the language with the `sap-language` parameter. For example, the following URI returns a metadata document with all labels in German:

```
https://<API-Server>/odata/v2/User/$metadata?sap-language=de-DE
```

## Retrieving Metadata Using `/Entity('EntityName')`

You can also use `Entity('EntityName')` to query the metadata of a single entity type. This type of query allows you to specify the format of content using the `$format` query option.

The following example query retrieves the metadata document of the `User` entity type in JSON format:

```
https://<API-Server>/odata/v2/Entity('User')?$format=JSON
```

The returned metadata contains entity set information of navigation properties as shown in the snippet below:

```
{
  "aggregationRole": null,
  "businessKey": false,
  "defaultValue": null,
  "displayFormat": null,
  "elmStrength": "Unclassified",
  "fieldControl": null,
  "filterable": true,
  "fromRole": {
    "multiplicity": {
      "name": "ZERO_TO_ONE",
      "symbolString": "0..1"
    },
    "path": "approverNav_of_Advance/approverNav",
    "role": "approverNav",
    "EntitySet": "User"
  },
  "id": false,
  "insertable": false,
  "insertablePath": null,
  "label": null,
  "maxLength": null,
  "name": "approverOfAdvanceNav",
  "navigateToPojo": false,
  "path": "User/approverOfAdvanceNav",
  "precision": null,
  "relationship": {
    "deletable": true,
    "end1": {
      "multiplicity": {
        "name": "MANY",
        "symbolString": "*"
      },
      "path": "approverNav_of_Advance/Advance",
      "role": "Advance",
      "EntitySet": "Advance"
    },
    "end2": {
      "multiplicity": {
        "name": "ZERO_TO_ONE",
        "symbolString": "0..1"
      },
      "path": "approverNav_of_Advance/approverNav",
      "role": "approverNav",
      "EntitySet": "User"
    },
    "insertable": true,
    "name": "approverNav_of_Advance",
    "path": "approverNav_of_Advance",
    "updatable": true,
    "upsertable": true
  },
  "required": false,
  "scale": null,
```

```

    "semantics": null,
    "sensitive": false,
    "sortable": false,
    "text": null,
    "toRole": {
      "multiplicity": {
        "name": "MANY",
        "symbolString": "*"
      },
      "path": "approverNav_of_Advance/Advance",
      "role": "Advance",
      "EntitySet": "Advance"
    },
    "type": {
      "name": "string",
      "path": "string"
    },
    "updatable": false,
    "upsertable": false,
    "viewable": true
  }
}

```

## 8.2 OData Annotations for EntitySet

The OData Metadata document for SAP SuccessFactors HXM Suite contains the following information about the EntitySet extension.

The EntitySet SF Extension

ATTRIBUTE	DEFAULT	DESCRIPTION
sap:label	-	Description of the entity type.
sap:creatable	True	Instances of this entity type can be created.
sap:updatable	True	Instances of this entity type can be updated.
sap:upsertable	True	Instances of this entity type can be upserted.
sap:deletable	True	Instances of this entity type can be deleted.

### ❖ Example

```

<EntitySet Name="PicklistOption" EntityType="SFOData.PicklistOption"
  sap:label="PicklistOption" sap:creatable="true" sap:updatable="true"
  sap:upsertable="false" sap:deletable="false">

```

## 8.3 OData Annotations for Properties

The OData metadata document for SAP SuccessFactors HXM Suite contains the following information about properties.

Annotations for OData Properties

Attribute	Default Value	Meaning
Nullable	true	Values of this property can be empty (null).
sap:required	false	Value of this property is required.
sap:creatable	true	Values of this property can be chosen by client when creating an instance.
sap:updatable	true	Values of this property can be changed.
sap:upsertable	true	Values of this property can be upserted.
sap:visible	true	Values of this property are visible to end user. The property can be used in the <code>\$select</code> system query option. If <code>sap:field-control</code> attribute is available for the property, the visibility is further determined by the value of <code>sap:field-control</code> .
sap:sortable	true	The property can be used in the <code>\$orderby</code> system query option.
sap:filterable	true	The property can be used in the <code>\$filter</code> system query option.
sap:label	-	Description of the property.
sap:field-control	-	A path expression that identifies a property containing a numeric value that controls visibility: <ul style="list-style-type: none"><li>0 = hidden: The property isn't visible on user interfaces.</li><li>1 = read-only: The property can't be changed.</li><li>3 = optional: The property may contain a null value (indicates writeable).</li><li>7 = mandatory: The property must contain a value (indicates writeable).</li></ul>

### ❖ Example

```
<Property Name="instrInterview" Type="Edm.String" Nullable="true"
sap:required="false" sap:creatable="true" sap:updatable="true"
sap:upsertable="true" sap:visible="true" sap:sortable="false"
sap:filterable="false" sap:field-control="jobReqPermissionsNav/instrInterview"
sap:label="Attach interview questions and supplementary companion documents:">
</Property>
```

## 8.4 OData Annotations for Navigation Properties

The OData metadata document for SAP SuccessFactors HXM Suite contains the following information about navigation properties.

Annotations for OData Navigation Properties

Attribute	Default Value	Meaning
sap:required	false	Value of this navigation property is required.
sap:creatable	true	Values of this navigation property can be chosen by client when creating an instance.
sap:updatable	true	Values of this navigation property can be changed.
sap:upsertable	true	Values of this navigation property can be upserted.
sap:visible	true	Values of this navigation property are visible to end users. If <code>sap:field-control</code> attribute is available for the navigation property, the visibility is further determined by the value of <code>sap:field-control</code> .
sap:sortable	true	The navigation property can be used in the <a href="#">\$orderby [page 46]</a> system query option.
sap:filterable	true	The navigation property can be used in the <a href="#">\$filter [page 49]</a> system query option.
sap:field-control	-	<p>A path expression that identifies a property containing a numeric value that controls visibility:</p> <ul style="list-style-type: none"><li>• 0 = hidden: The property isn't visible on user interfaces.</li><li>• 1 = read-only: The property can't be changed.</li><li>• 3 = optional: The property may contain a null value (indicates writeable).</li><li>• 7 = mandatory: The property must contain a value (indicates writeable).</li></ul>

### Example

```
<NavigationProperty Name="skillNav" sap:required="true" sap:creatable="true"
sap:updatable="true" sap:upsertable="true" sap:visible="true" sap:sortable="true"
sap:filterable="true" Relationship="SFOData.skillNav_of_SelfReportSkillMapping"
FromRole="SelfReportSkillMapping" ToRole="skillNav" sap:field-
control="SelfReportSkillMappingFieldControlsNav/skillNav"
sap:label="Skill"></NavigationProperty>
```



## 8.5 OData Service Catalog

Service Catalogs provide a way to group APIs together in contextually-relevant groups, greatly reducing the size of the \$metadata for each catalog.

Use the `http://<api-server>/odata/v2/CatalogService/$metadata` API call to generate a catalog of services.

### Sample XML Response

Sample XML response is shown below :

```
<?xml version='1.0' encoding='utf-8'?>
  <edmx:Edmx Version="1.0" xmlns:edmx= "http://schemas.microsoft.com/ado/2007/06/edmx" xmlns:atom= "http://www.w3.org/2005/Atom" >
    <edmx:DataServices m:DataServiceVersion= "2.0" xmlns:m= "http://schemas.microsoft.com/ado/2007/08/dataservices/metadata" >
      <Schema Namespace= "CATALOGSERVICE" xmlns= "http://schemas.microsoft.com/ado/2008/09/edm" xmlns:sf= "http://www.successfactors.com/edm/sf" >
        <EntityType Name= "Service">
          <Key>
            <PropertyRef Name= "ID" />
          </Key>
          <Property Name= "ID" Type= "Edm.String " Nullable= "false " />
          <Property Name= "Description" Type= "Edm.String " Nullable= "false " />
          <Property Name= "Author" Type= "Edm.String " Nullable= "false " />
          <Property Name= "MetadataUrl" Type= "Edm.String " Nullable= "false " />
          <Property Name= "ServiceUrl" Type= "Edm.String " Nullable= "false " />
        </EntityType>
        <EntityContainer Name= "CATALOGSERVICE_Entities"
m:IsDefaultEntityContainer= "true ">
          <EntitySet Name= "ServiceCollection" EntityType=
"CATALOGSERVICE.Service" />
        </EntityContainer>
        <atom:link rel= "self" href= "http://<api-server>/odata/v2/CatalogService/$metadata" />
        <atom:link rel= "latest-version" href= "http://<api-server>/odata/v2/CatalogService/$metadata" />
        <atom:link rel= "analytics" href= "https://<api-server>/ProductionData/ODataService.svc" />
      </Schema>
    </edmx:DataServices>
  </edmx:Edmx>
```

### Response Description

This XML output follows regular OData metadata format. It contains 2 parts:

1. An OData entity named 'Service'. Currently this entity is not accessible excepts it's definition in metadata.
2. Several `<atom:link>` links, each of which publishes a service URL. Each data center is be configured to publish different service URLs. The `self` and `latest-version` links always point to the catalog service itself. The `analytics` link published the Analytics OData API URL which is the only URL available currently.

# 9 Errors

This section contains lists of OData v2 errors.

[Common OData Errors \[page 202\]](#)

A list of common OData errors.

[Logon Errors \[page 206\]](#)

A list of API logon errors.

[OAuth 2.0 Errors \[page 208\]](#)

A list of error codes and messages related to OAuth 2.0.

## 9.1 Common OData Errors

A list of common OData errors.

Error Code	Refine Error Message	HTTP Code	Solutions
COE0001 - COE0002	Error messages related to missing permissions.	N/A	Assign the corresponding permissions.
COE0003	Bad property expression.	404  400	N/A
COE0005	The given value <a> for property <b> is invalid. <c>. Please check the property in <a href="#">Admin Center</a> > <a href="#">OData API Data Dictionary</a> .	400	<ul style="list-style-type: none"><li>Check entity's property attribute in <a href="#">OData API Data Dictionary</a>.</li><li>Check the entity metadata and for the property attribute.</li></ul>
COE0007	Entity or function name <a> is not found. <b>	400	<ul style="list-style-type: none"><li>Check spellings.</li><li>Check if a navigation is missing and add it in object definition.</li><li>If there's a recent change in object definition in the data model, refresh the OData metadata.</li></ul>
COE0008	Invalid function import name: <a>. Please check the name in <a href="#">Admin Center</a> > <a href="#">OData API Data Dictionary</a> .	400	Enter the correct function name.
COE0009	Entity {0} is not insertable.	400	Do not insert records to this entity.
COE0010	Entity {0} is not updatable.	400	Do not update records of this entity.
COE0011	Entity {0} is not upsertable.	400	Do not upsert records of this entity.
COE0012	Entity {0} is not deletable.	400	Do not delete records of this entity.

Error Code	Refine Error Message	HTTP Code	Solutions
COE0013	Entity {0} is not replaceable.	400	Do not replace records of this entity.
COE0014	Links {0} is not insertable.	400	Do not insert links.
COE0015	Links {0} is not updatable.	400	Do not update links.
COE0016	Links {0} is not upsertable.	400	Do not upsert links.
COE0017	Links {0} is not deletable.	400	Do not delete links.
COE0018	Bad request.	400	<ul style="list-style-type: none"> <li>• If it is a GET request, make sure the field names are correct and the navigation is properly formed.</li> <li>• If it is a POST request, make sure the mandatory fields present in the request payload and verify the format in which the request is being sent.</li> </ul>
COE0021	Invalid property names: {0}. Please check the property name in <a href="#">Admin Center</a> > <a href="#">OData API Data Dictionary</a> or entity metadata.	400	Enter a correct property name.
COE0022	Properties {0} are not accessible. Please check the property name in <a href="#">Admin Center</a> > <a href="#">OData API Data Dictionary</a> or the entity metadata.	400	Do not query inaccessible properties.
COE0023	Property {0} is not visible. Please check the visibility setting of the property in <a href="#">Admin Center</a> > <a href="#">OData API Data Dictionary</a> or the entity metadata.	400	Do not query invisible properties.
COE0024	Unsupported entity set category: <a>. Only public, restricted, internal, and beta categories are supported.	400	Query only supported categories.
COE0025	Unsupported feature: <detail error message>	400	<ul style="list-style-type: none"> <li>• Check if the operation which is being carried on the entity is supported by the entity in the data dictionary.</li> <li>• Check if navigation is expanded without the base entity.</li> <li>• Check if you're expanding the wfRequestNav property of an MDF entity that does not have workflow enabled. If workflow is not enabled, you're not allowed to expand wfRequestNav.</li> </ul>
COE0026	Unsupported locale: <a>. For a list of supported languages and locales, please visit <a href="#">2269945</a>	400	Use a supported locale.
COE0027	Please add required properties in payload. <a>. You can check which properties are required for an entity in <a href="#">Admin Center</a> > <a href="#">OData API Data Dictionary</a> or the entity metadata.	400	Check the dictionary and add required properties in the payload.

Error Code	Refine Error Message	HTTP Code	Solutions
COE0028	Record for entity <a> with key <b> does not exist. Please check the key value.	400  404	Change key value.
COE0029	A record for entity <a> with key <b> already exists. Please use another key.	400	Change key value.
COE0030 - COE0032	<Message text from another module>. Please check the property setting in <a href="#">Admin Center</a> > <a href="#">OData API Data Dictionary</a> or the entity metadata.	400	Check the OData API Data Dictionary or entity metadata.
COE0033	Maximum length limit exceeded for field <a>: actual length: <b>; maximum length: <c>.	400	Reduce the length of the field value.
COE0034	Property <a> is not inline-required. Please check the sap:inlineRequired setting of the property in <a href="#">Admin Center</a> > <a href="#">OData API Data Dictionary</a> or the entity metadata.		Check the OData API Data Dictionary or entity metadata.
COE0035	Unable to parse option roleNames {0}. Please verify the format.		Verify the format of customOption roleNames.
COE0037	Entity <a> is not editable. Please check the entity setting in <a href="#">Admin Center</a> > <a href="#">OData API Data Dictionary</a> or the entity metadata.	400	Do not edit records for the entity.
COE0038	Invalid service alias name: {0}. Please contact your system administrator to check the service alias name.		Use the correct alias name. For now, we only have three service aliases: _CPMEntityService, _FeatureProvisioningService, and _TodoEntryV2Service.
COE0039	<Message text from another module>		Do not use service aliases along with entity/function/complexType.
COE0040	<Message text from another module>	503	Dependent on modules.
COE0041	The given value <a> for parameter <b> is invalid. Please check the value type in <a href="#">Admin Center</a> > <a href="#">OData API Data Dictionary</a> and use the correct value in API request.	400	Enter the correct parameter value.
COE0042	Parameter {0} is missing. Please check required parameters in <a href="#">Admin Center</a> > <a href="#">OData API Data Dictionary</a> and add the required parameters in API request.	400	Add the required parameter and value.
COE0044	Expand size {0} exceeds the limit {1}.	454	Reduce \$expand size.

Error Code	Refine Error Message	HTTP Code	Solutions
COE0046	Expand depth {0} of {1} exceeds the limit {2}.	454	Reduce \$expand depth.
COE0048	Property <a> is missing in function import payload. Please add required parameters in payload.	400	Add the missing property and its value to the payload.
COE0049	Value <a> for property <b> in function import payload is invalid. Please correct the value in payload.	400	Enter the correct value for the property.
COE0050	The query has exceeded the <n>-minute timeout limit.	500	Retry.
COE0051	The server has run out of resource. Please retry after <n> minutes.	503	Retry.
COE0052	Expand limit of <n> records was exceeded. Please decrease the number of records to expand or use filters to narrow down the result.	454	Reduce the expand size.
COE0053	Invalid service group name: <a>. Please contact your system administrator to check the service group name.		Enter a valid service group.

Parent topic: [Errors \[page 202\]](#)

## Related Information

[Logon Errors \[page 206\]](#)

[OAuth 2.0 Errors \[page 208\]](#)

## 9.2 Logon Errors

A list of API logon errors.

Error Code	Error Message	HTTP Code	Solution
LGN0001	<p>Internal: Invalid endpoint odata/v2/internal. You aren't allowed to access the internal category with an external request.</p> <p>Test: Invalid endpoint odata/v2/test. The test category is not enabled in the production environment.</p> <p>Beta: Invalid endpoint odata/v2/beta. The beta category is not enabled. Please contact your system administrator to enable the OData API Public Beta Category.</p>	404 400	<p>You're trying to access an OData API category that's not available in your instance.</p> <ul style="list-style-type: none"> <li>• If it's an internal category, you have no access to it.</li> <li>• If it's a test category, you can only access it in a test instance.</li> <li>• If it's a beta category, contact your system administrator and find out whether you have the access.</li> </ul>
LGN0002	<p>Authentication failed. You do not have the admin permission to access OData API. Please check the permission setting for your role at <a href="#">Manage Integration Tools &gt; Allow Admin to Access OData API through Basic Authentication</a>.</p>	403	Assign the corresponding permission in RBP.
LGN0003	<p>Authentication credentials are required. Please provide a valid username, password, and company ID.</p>	401	Provide your credentials ( <b>&lt;username&gt;@&lt;companyID&gt;: &lt;password&gt;</b> ) and encode them using Base64 format.
LGN0004	<p>You are not allowed to access OData APIs using Basic Auth or OAuth on a non-API server. To do this, you need to switch to the corresponding API server. For a list of available servers, visit <a href="#">About HXM Suite OData APIs (V2) [page 5]</a>.</p>	401	Choose the right API server.
LGN0005	<p>Wrong token format. Please check your token for OAuth authentication.</p>	400	Follow the steps in <a href="#">Authentication Using OAuth 2.0 [page 15]</a> to generate a new OAuth token.
LGN0006	<p>Token {0} has expired. Please log in again or perform a getToken operation to refresh your access token.</p>	401	Follow the steps in <a href="#">Authentication Using OAuth 2.0 [page 15]</a> to generate a new OAuth token.
LGN0007	<p>Invalid OAuth token {0}. Please check the token value.</p>	401 403	Follow the steps in <a href="#">Authentication Using OAuth 2.0 [page 15]</a> to validate your token or generate a new OAuth token if needed.
LGN0008	<p>Company {0} not found. Make sure you have entered the correct company ID. Please note that the company ID is case-sensitive.</p>	401	Enter the correct company ID.
LGN0009	<p>OData API is not enabled for this company. Please contact SAP support to enable the feature.</p>	401	Contact Product Support to enable OData API for your company.

Error Code	Error Message	HTTP Code	Solution
LGN0010	Basic Authentication is disabled for company <companyID> or your IP <IP address> is not authorized to access OData API. To enable Basic Authentication, grant the permission to your role at <a href="#">Manage Integration Tools</a> > <a href="#">Manage OData API Basic Authentication</a> . To authorize your IP address for OData API access, go to <a href="#">API Center</a> > <a href="#">OData API IP Whitelisting</a> .	401	Assign the corresponding permission and add your IP address to the allowlist.
LGN0011	Authentication failed. You have provided an expired company (status code = <code>). Please contact your system administrator to reset the expiry date.	401	Contact Product Support.
LGN0012	Authentication failed. You have provided an inactive company (status code = <code>). Please contact your system administrator to re-activate the company.		Contact your system administrator to re-activate the company.
LGN0013	Authentication failed. We have prevented an attempted login from unauthorized IP: <IP address> to company ID: <companyID> with username: <username> (status code = <code>). To authorize the login, add an exception in <a href="#">Admin Center</a> > <a href="#">Company Settings</a> > <a href="#">Password &amp; Login Policy Settings</a> > <a href="#">Set API Login Exceptions</a> . For more information, visit <a href="#">Setting the Password Policy for API Access [page 13]</a> .	401	Add an exception as instructed.
LGN0014	Authentication failed. The user account is locked (status code = <code>). Please contact your system administrator to unlock your account.	401	Unlock the user.
LGN0015	Authentication failed. You have entered an incorrect username or password (status code = <code>).	401	Enter the correct username and password.
LGN0016	Authentication failed. The user has no login permission (status code = <code>). Please check if the following permissions are granted: <ul style="list-style-type: none"> <li>• <a href="#">General User Permission</a> &gt; <a href="#">User Login</a> .</li> <li>• For SFAPI: <a href="#">General User Permission</a> &gt; <a href="#">SFAPI User Login</a> .</li> <li>• For OData API: <a href="#">Manage Integration Tools</a> &gt; <a href="#">Allow Admin to Access OData API through Basic Authentication</a> .</li> </ul>	401	Check permission and contact your system administrator to enable permission. If the user is locked, go to <a href="#">Admin Center</a> > <a href="#">Reset User Account</a> .
LGN0018	Authentication failed. The password has expired (status code = <code>). Please log in to SAP SuccessFactors and reset the password, or contact your system administrator.	401	Reset your password or contact your system administrator..

Error Code	Error Message	HTTP Code	Solution
LGN0020	Incorrect format for HTTP Basic Authentication. Please encode your credentials ( <b>&lt;username&gt;@&lt;companyID&gt;:&lt;password&gt;</b> ) using Base64 format. For more information, visit <a href="#">HTTP Basic Authentication (Deprecated) [page 14]</a> .	401	Enter the credentials in correct format.
LGN0023	Person-only is not supported when Global Assignment and Concurrent Employment features are not enabled. Please contact your system administrator to enable the features.	400	Contact your system administrator to turn on Global Assignment and Concurrent Employment features.
LGN0024	Employee Central V2 feature is not enabled. Please contact your system administrator to enable this feature.		Contact your system administrator to turn on Employee Central V2 feature.
LGN0025	Authentication failed. The company ID is invalid (status code = <code>).	401	Enter a valid company ID.

Parent topic: [Errors \[page 202\]](#)

## Related Information

[Common OData Errors \[page 202\]](#)

[OAuth 2.0 Errors \[page 208\]](#)

## 9.3 OAuth 2.0 Errors

A list of error codes and messages related to OAuth 2.0.

ERROR STRING	HTTP RESPONSE CODE	RESPONSE ERROR	COMMENTS
OAUTH2_ERROR_INVALID_CLIENT_INFO	500	Client information contains error.	
OAUTH2_ERROR_MISSING_REQUIRED_CLIENT_INFO_FIELD	500	Missing required field in client information.	
OAUTH2_ERROR_INVALID_CLIENT_INFO_FIELD	500	Some client information field contains an error.	
OAUTH2_ERROR_CLIENT_ALREADY_EXISTS	500	The client application (API key = "###") already exists.	



ERROR STRING	HTTP RESPONSE CODE	RESPONSE ERROR	COMMENTS
OAUTH2_ERROR_CLIENT_NOT_EXIST	500	The client application (API key = "###") does not exist.	
OAUTH2_ERROR_UNABLE_TO_REGISTER_CLIENT	500	Uable to register client application.	
OAUTH2_ERROR_UNABLE_TO_DEREGISTER_CLIENT	500	Unable to deregister client application.	
OAUTH2_ERROR_INVALID_TOKEN_URL	500	The token URL is invalid.	The token URL is in an incorrect format.
OAUTH2_ERROR_INVALID_X509_PRIVATE_KEY	500	The X.509 private key contains an error.	The X.509 private key is invalid.
OAUTH2_ERROR_UNABLE_TO_GENERATE_SAML_ASSERTION	500	Unable to generate SAML assertion.	OAuth server is unable to construct a valid X.509 private key object.
OAUTH2_ERROR_UNABLE_TO_SIGN_SAML_ASSERTION	500	Unable to sign SAML assertion.	Failed to sign SAML assertion using the provided X.509 private key.
OAUTH2_ERROR_MISSING_REQUIRED_PARAM	400	Parameter "###" is required in the OAuth request.	For a get access token request, grant_type, client_id, company_id and assertion fields are required. To generate a SAML assertion, client_id, token_url, user_id and private_key fields are required.
OAUTH2_ERROR_INVALID_GRANT_TYPE	400	The grant_type is not supported.	grant_type must be set to urn:ietf:params:oauth:grant-type:saml2-bearer
OAUTH2_ERROR_UNABLE_TO_COLLECT_SAML_ASSERTION	400	Unable to retrieve SAML assertion.	The SAML assertion is not Base64 encoded. Missing Issuer, SubjectNameId, Audiences, or Recipients in the assertion.
OAUTH2_ERROR_SAML_ASSERTION_EXPIRED	400	The provided SAML assertion is expired.	The SAML assertion usually expires in 5-10 minutes.

ERROR STRING	HTTP RESPONSE CODE	RESPONSE ERROR	COMMENTS
OAUTH2_ERROR_UNABLE_TO_VERIFY_SAML_ASSERTION	401	Unable to verify the signature of the SAML assertion.	SAML assertion is verified using the same X.509 certificate that was provided during OAuth client application registration.
OAUTH2_ERROR_UNABLE_TO_VALIDATE_SAML_ASSERTION	401	Unable to validate "###" in the SAML assertion.	The Issuer field in the assertion does not match the API_KEY. Audiences does not include www.successfactors.com. Recipients points to a different token host.
OAUTH2_ERROR_COMPANY_NOT_EXIST	401	Company "###" does not exist.	
OAUTH2_ERROR_COMPANY_INACTIVE	401	Company "###" is not currently active.	
OAUTH2_ERROR_COMPANY_LICENSE_EXPIRED	401	Company "###" has expired license.	
OAUTH2_ERROR_API_KEY_NOT_EXIST	401	API key "###" does not exist in company "###".	
OAUTH2_ERROR_API_KEY_DISABLED	401	API key "###" has been disabled in company "###".	
OAUTH2_ERROR_AUTHENTICATION_FAILED	401	Unable to authenticate the client.	An OAuth server error occurred while authenticating access_token.
OAUTH2_ERROR_AUTHORIZATION_FAILED	403	Unable to authorize the client.	An OAuth server error occurred while authorizing access_token.
OAUTH2_ERROR_UNABLE_TO_SIGN_TOKEN	403	Unable to sign access token.	The OAuth server uses an SAP SuccessFactors server certificate to sign the access token. An error occurs if the access token has already been signed, or if the SAP SuccessFactors server certificate is missing.
OAUTH2_ERROR_UNABLE_TO_GRANT_TOKEN	403	Unable to grant access token.	Content is missing in the access token, or a runtime error happened while constructing the access token (this includes JSON-generation exceptions).

ERROR STRING	HTTP RESPONSE CODE	RESPONSE ERROR	COMMENTS
OAUTH2_ERROR_UNABLE_TO_BUILD_TOKEN_RESPONSE	500	Unable to build OAuth response.	
OAUTH2_ERROR_MISSING_REQUIRED_HEADER	400	Header "###" is required.	Missing Authorization header in validate token request, or when protected resources were accessed.
OAUTH2_ERROR_MISSING_ACCESS_TOKEN	400	Access token is required.	Authorization header is empty.
OAUTH2_ERROR_MUST_BE_BEARER_TOKEN	400	Access token has to be a Bearer token.	Authorization header is not a Bearer format.
OAUTH2_ERROR_UNABLE_TO_COLLECT_TOKEN	400	Unable to retrieve access token.	The access token is missing content, or a parsing error happened while deserializing an access token in JSON format.
OAUTH2_ERROR_UNABLE_TO_VERIFY_TOKEN	401	Unable to verify the signature of the access token.	The access token has modified the HTTP MITMA. The SAP SuccessFactors server public key couldn't be found.
OAUTH2_ERROR_UNABLE_TO_VALIDATE_TOKEN	401	Unable to validate access token.	The Access token contains invalid content or has insufficient information (for example, missing client_id, company_id, and so on).
OAUTH2_ERROR_TOKEN_INVALID_APIKEY	401	The access token does not apply to API key "###".	
OAUTH2_ERROR_TOKEN_INVALID_COMPANY	401	The access token does not apply to company "###".	
OAUTH2_ERROR_TOKEN_INVALID_APPLICATION_NAME	401	The access token does not apply to application name "###".	Verify the application name specified during the OAuth client application registration.
OAUTH2_ERROR_TOKEN_REJECTED_OR_EXPIRED	403	The access token is either rejected or expired.	Token has expired (default is 24 hours).
OAUTH2_ERROR_INSUFFICIENT_PERMISSION_SCOPE	403	Insufficient permission scope.	
OAUTH2_ERROR_UNABLE_TO_GRANT_ACCESS	403	Unable to grant access.	External User.

ERROR STRING	HTTP RESPONSE CODE	RESPONSE ERROR	COMMENTS
OAUTH2_ERROR_UNABLE_TO_BUILD_RESPONSE	500	Unable to build token validation response.	
INTERNAL_ERROR	500	Internal error.	

**Parent topic:** [Errors \[page 202\]](#)

## Related Information

[Common OData Errors \[page 202\]](#)

[Logon Errors \[page 206\]](#)

# 10 OData API Best Practices

Read and follow the best practices in this topic for optimal OData API performance and better user experience.

Best Practice	Details	More Information
Query Only Modified Records	Instead of querying all records, query only the records that have been modified since your last execution for integration use cases, and use server pagination to ensure stable results.	<a href="#">Pagination [page 67]</a>
Use Server-Side Pagination for Integrations	Compared with client pagination, server pagination is faster and doesn't suffer data loss and duplication issues when simultaneous edits occur in the same instance. You can choose either cursor-based or snapshot-based pagination for your integration use case.	<a href="#">Pagination [page 67]</a>
Avoid Frequent Job Runs to Get Real-Time Results	Repeating queries in much less than one hour will consume too much API resource, which may be throttled in the future. In extreme cases, you may even be denied of service because of frequent queries, especially ones that require heavy backend processing.	
Avoid Excessive Queries for Single Records	<p>Querying records one at a time doesn't take advantage of database optimization capabilities. Singleton queries are often used to perform in-memory joins and must be avoided whenever possible. Here are two bad examples:</p> <pre>&amp;\$filter= &lt;key&gt; eq &lt;keyvalue&gt;</pre> <pre>/odata/v2/User('user1')</pre>	

Best Practice	Details	More Information
Use \$batch or \$filter to Get Multiple Records	<p>Instead of pulling many records one at a time using key predicate, use \$batch or the \$filter IN clause.</p> <p>In OData API, a user login session is created on the server for each request. This is a resource demanding process, especially when login audit is enabled. A high volume of requests may lead to excessive usage of system resource and drastically slow down performance. Therefore, we encourage you to use \$batch operation or the IN clause in \$filter query option whenever possible. For example, instead of using the ToDo API to query multiple users with many requests, use the TodoEntryV2, which allows you to do it in one request.</p> <p>Use \$batch if you want to operate on different entities or if you want to perform different operations on the same entity.</p> <p>Use \$filter IN if you want to query multiple keys using the same entity.</p>	<p><a href="#">\$filter [page 49]</a></p> <p><a href="#">Batch Operations [page 108]</a></p>
Use \$select to Get Only the Properties You Need	<p>Without the \$select query option, a query returns all available properties of an entity. For better performance, we recommend that you add the \$select system query option to specify only the properties you need.</p>	<p><a href="#">\$select [page 58]</a></p>
Do Not Use API for Integration That is Only Designed for Single User	<p>APIs designed for single users aren't ideal in integration scenarios. Iterating requests for all uses creates one login session for each user and significantly slows down performance. One example is the ToDo API, which only allows querying data of a single user. For integration purpose, use the new TodoEntryV2 API. TodoEntryV2 allows you to query items of multiple users with the <a href="#">OData API Todo Export</a> permission.</p>	<p><a href="#">TodoEntryV2</a></p>

Best Practice	Details	More Information
Tune Your Batch Requests into Proper Sizes	<p>The OData API can return a maximum number of 1000 records in a single page. You should tune your batch sizes to be as large as possible. For more complex transactions, you need to decrease the size to avoid HTTP timeouts.</p>	
Avoid Large \$expand Statements	<p>The \$expand statement can be used to request master-detail data. However, using \$expand to join a large number of tables can lead to poor performance. We recommend that you first fetch master by batch and detail on demand when user requests.</p> <p>For example, A query that retrieves all JobRequisitions and expands all attachments for each application is considered too complex and large. This type of queries is error-prone and will be throttled in the future.</p> <p>The recommend master-detail implementation steps are:</p> <ol style="list-style-type: none"> <li>1. Bind the master list to simple top-level Job Requisition so UI5 loads in on demand via batch operations as the user scrolls.</li> <li>2. When the user clicks on a job requisition, only then do you bind the detail list to candidate, filtering on the selected job requisition id.</li> <li>3. Only load resume and cover letter when you click on a job candidate.</li> </ol> <div> <p><b>i Note</b></p> <p>Caching Picklist and Foundation Object lookups might be efficient for a large data set where "cache efficiency" can be high.</p> </div>	
Keep Transactions Simple to Avoid Poor Performance	<p>Poor performance is often a sign of misuse of APIs. As a rule of thumb, always keep your transactions simple.</p>	
Compress Data for Faster Transmission	<p>Compress the data by adding header <code>Accept-Encoding=gzip</code> in your requests for faster data transmission.</p>	

Best Practice	Details	More Information
Sync Frequently and Sync Only Delta Changes	Full data sync can lead to high resource consumption and poor performance. Instead of doing a full data sync intermittently, schedule more frequent data sync but limit the scope to only delta changes.	
Query Properties and Expand Entities Only When You Need Them	<p>It's easy to build a query that does more \$select and \$expand than you need. It's especially easy with the Boomi connector, which has UI limitations due to limitations in the Dell Boomi connector toolkit.</p> <p>As a developer, you might get tired of tweaking your queries to match the content as requirements change so often. You might be tempted to pull more than you need and release the queries to production without tuning.</p> <p>The Integration Center can help avoid this situation. Instead of a building a query manually based on the mapping of fields, the Integration Center generates the query based on the mapping.</p>	
Tune Client Wait Time to Match System Wait Time	<p>Set your client to wait a reasonable amount of time before timing out. Complex operations can take as long as 10 minutes and the network and servers continue to process a transaction during that time. It's best to wait for the transaction to complete rather than wasting it.</p> <p>You can also tune your transaction to avoid timeouts.</p>	



Best Practice	Details	More Information
Implement Retry Logic Properly	<p>Retry logic can help recover failed transactions due to Internet connectivity or backend server issues, but retry must be attempted with caution based on the type of HTTP error.</p> <p>In all cases, wait at least between 1 to 5 minutes before attempting a retry. Excessive immediate retry can cause denial of service giving little time for the server to recover.</p> <p>Retry no more than 5 times before abandoning your task. For example, the Boomi connector retries only a maximum of 5 times.</p> <p>Error types eligible for retry:</p> <ul style="list-style-type: none"> <li>• HTTP response code 5XX other than 500. A 500 error can't recover because there's a problem with your query or payload. Note <a href="#">503 Service Not Available</a> can occur when a server is overloaded. It can also occur during maintenance periods. Avoid running queries during these published maintenance periods.</li> <li>• HTTP response code 412 can occur to edit operations during optimistic locking on back-end transactions. <ul style="list-style-type: none"> <li>◦ For batch operations, only retry failed records. Do not include successful edit operations in the retry payload. This includes specific transaction in a \$batch and specific instances in a multirecord upsert payload.</li> <li>◦ For 412 errors, only retry once.</li> </ul> </li> <li>• HTTP timeout errors can be retried only if your client timeout matches the infrastructure timeout of 5 minutes. If you retry a timeout before this period, you're stacking transactions on top of each other and performing a possible denial-of-service attack.</li> </ul>	

Best Practice	Details	More Information
	<ul style="list-style-type: none"> <li>• HTTP connection reset error and no response are given from server side.</li> </ul> <p>Errors for which retry shouldn't be attempted:</p> <ul style="list-style-type: none"> <li>• 401 Authentication Failed error. You have provided incorrect credentials and retrying doesn't help.</li> <li>• 404 Not Found error. You can't recover something that doesn't exist.</li> <li>• 400 Bad Request error. This error can occur for edit operations that are missing required data or have incorrect formats.</li> <li>• Do not retry Insert or Upsert operations where the object has an auto-increment key. Both can result in duplicate data.</li> </ul>	
Avoid Inline Upserting Referenced Objects	<p>Although it's technically possible to inline upsert a source object that is referenced in an MDF object as a Generic Object (GO) or as a Valid When relationship, we don't recommend it because of performance issues. Use inline upsert only for objects with a parent-child relationship, in other words, a composite association.</p>	
Avoid Upserting Composite Objects Directly	<p>If a parent entity has multiple composite type associations and more than one of them point to the same destination object, do not upsert the destination object directly. OData API does not know which association to edit and it always edits the first association. To be able to distinguish one association from another, inline upsert the parent with the corresponding association.</p>	

Best Practice	Details	More Information
Assign <a href="#">Admin access to MDF OData API</a> Permission Properly	The <a href="#">Admin access to MDF OData API</a> permission is an admin permission that allows users to query and edit all MDF OData entities. This permission overrides all entity-level permissions and is intended for integration scenarios only. If you want to use an MDF OData entity on UI, entity-level permissions might be a better choice.	
Avoid Excessive Client Multithreading	<p>To prevent server overloads and ensure high availability, we recommend a maximum of 10 concurrent API requests for any company instance.</p> <p>Multithreading is only allowed for editing single records. Do not use it for queries/read operations, massive upserts, or \$batch operations.</p>	
Design Integration with Security in Mind	<p>Observe the following rules while you design your integrations:</p> <ul style="list-style-type: none"> <li>• Use OAuth 2.0 as your authentication method for the technical user.</li> <li>• Create one user for each integration process. Do not share users among different processes.</li> <li>• Do not put sensitive information in URIs. If you can't avoid it, use HTTP POST tunnel to hide it or use \$batch operations.</li> </ul>	



Best Practice	Details	More Information
Add Tracing Headers to Requests	<p>Always include the following tracing headers in your requests so that Product Support Ops team can easily trace the API calls and troubleshoot your problems faster:</p> <ul style="list-style-type: none"> <li>• <code>X-SF-Correlation-Id</code>: A unique GUID included in the header and in the log file for each HTTP request/page request.</li> <li>• <code>X-SF-Execution-Id</code>: Execution ID of the integration process.</li> <li>• <code>X-SF-Process-Name</code>: Process name of the integration.</li> <li>• <code>X-SF-Process-Id</code>: Process GUID of the integration.</li> <li>• <code>X-SF-Client-Tenant-Id</code>: Integration account ID.</li> </ul>	
Use the Latest Metadata	<p>Before you implement any integration, always try to get the latest metadata of the entities. Data model changes can happen without you knowing it and the metadata you have can be out of date.</p> <p>If an integration breaks, the first thing to do is to check whether the entity data model has changed by looking at the latest metadata.</p>	
Use Immutable IDs to Retrieve Data	<p>Always use immutable IDs such as external ID, person GUID, and external code to retrieve data. Mutable IDs such as assignment ID are prone to change. Using mutable IDs can break your integrations when the IDs change.</p>	

# Important Disclaimers and Legal Information

## Hyperlinks

Some links are classified by an icon and/or a mouseover text. These links provide additional information.

About the icons:

- Links with the icon : You are entering a Web site that is not hosted by SAP. By using such links, you agree (unless expressly stated otherwise in your agreements with SAP) to this:
  - The content of the linked-to site is not SAP documentation. You may not infer any product claims against SAP based on this information.
  - SAP does not agree or disagree with the content on the linked-to site, nor does SAP warrant the availability and correctness. SAP shall not be liable for any damages caused by the use of such content unless damages have been caused by SAP's gross negligence or willful misconduct.
- Links with the icon : You are leaving the documentation for that particular SAP product or service and are entering a SAP-hosted Web site. By using such links, you agree that (unless expressly stated otherwise in your agreements with SAP) you may not infer any product claims against SAP based on this information.

## Videos Hosted on External Platforms

Some videos may point to third-party video hosting platforms. SAP cannot guarantee the future availability of videos stored on these platforms. Furthermore, any advertisements or other content hosted on these platforms (for example, suggested videos or by navigating to other videos hosted on the same site), are not within the control or responsibility of SAP.

## Beta and Other Experimental Features

Experimental features are not part of the officially delivered scope that SAP guarantees for future releases. This means that experimental features may be changed by SAP at any time for any reason without notice. Experimental features are not for productive use. You may not demonstrate, test, examine, evaluate or otherwise use the experimental features in a live operating environment or with data that has not been sufficiently backed up.

The purpose of experimental features is to get feedback early on, allowing customers and partners to influence the future product accordingly. By providing your feedback (e.g. in the SAP Community), you accept that intellectual property rights of the contributions or derivative works shall remain the exclusive property of SAP.

## Example Code

Any software coding and/or code snippets are examples. They are not for productive use. The example code is only intended to better explain and visualize the syntax and phrasing rules. SAP does not warrant the correctness and completeness of the example code. SAP shall not be liable for errors or damages caused by the use of example code unless damages have been caused by SAP's gross negligence or willful misconduct.

## Bias-Free Language

SAP supports a culture of diversity and inclusion. Whenever possible, we use unbiased language in our documentation to refer to people of all cultures, ethnicities, genders, and abilities.

© 2022 SAP SE or an SAP affiliate company. All rights reserved.

No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP SE or an SAP affiliate company. The information contained herein may be changed without prior notice.

Some software products marketed by SAP SE and its distributors contain proprietary software components of other software vendors. National product specifications may vary.

These materials are provided by SAP SE or an SAP affiliate company for informational purposes only, without representation or warranty of any kind, and SAP or its affiliated companies shall not be liable for errors or omissions with respect to the materials. The only warranties for SAP or SAP affiliate company products and services are those that are set forth in the express warranty statements accompanying such products and services, if any. Nothing herein should be construed as constituting an additional warranty.

SAP and other SAP products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of SAP SE (or an SAP affiliate company) in Germany and other countries. All other product and service names mentioned are the trademarks of their respective companies.

Please see <https://www.sap.com/about/legal/trademark.html> for additional trademark information and notices.