



One Identity Manager 9.2

REST API Reference Guide

Copyright 2024 One Identity LLC.

ALL RIGHTS RESERVED.

This guide contains proprietary information protected by copyright. The software described in this guide is furnished under a software license or nondisclosure agreement. This software may be used or copied only in accordance with the terms of the applicable agreement. No part of this guide may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying and recording for any purpose other than the purchaser's personal use without the written permission of One Identity LLC .

The information in this document is provided in connection with One Identity products. No license, express or implied, by estoppel or otherwise, to any intellectual property right is granted by this document or in connection with the sale of One Identity LLC products. EXCEPT AS SET FORTH IN THE TERMS AND CONDITIONS AS SPECIFIED IN THE LICENSE AGREEMENT FOR THIS PRODUCT, ONE IDENTITY ASSUMES NO LIABILITY WHATSOEVER AND DISCLAIMS ANY EXPRESS, IMPLIED OR STATUTORY WARRANTY RELATING TO ITS PRODUCTS INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT. IN NO EVENT SHALL ONE IDENTITY BE LIABLE FOR ANY DIRECT, INDIRECT, CONSEQUENTIAL, PUNITIVE, SPECIAL OR INCIDENTAL DAMAGES (INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS OF PROFITS, BUSINESS INTERRUPTION OR LOSS OF INFORMATION) ARISING OUT OF THE USE OR INABILITY TO USE THIS DOCUMENT, EVEN IF ONE IDENTITY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. One Identity makes no representations or warranties with respect to the accuracy or completeness of the contents of this document and reserves the right to make changes to specifications and product descriptions at any time without notice. One Identity does not make any commitment to update the information contained in this document.

If you have any questions regarding your potential use of this material, contact:

One Identity LLC.
Attn: LEGAL Dept
4 Polaris Way
Aliso Viejo, CA 92656

Refer to our website (<http://www.OneIdentity.com>) for regional and international office information.

Patents

One Identity is proud of our advanced technology. Patents and pending patents may apply to this product. For the most current information about applicable patents for this product, please visit our website at <http://www.OneIdentity.com/legal/patents.aspx>.

Trademarks

One Identity and the One Identity logo are trademarks and registered trademarks of One Identity LLC. in the U.S.A. and other countries. For a complete list of One Identity trademarks, please visit our website at www.OneIdentity.com/legal/trademark-information.aspx. All other trademarks are the property of their respective owners.

Legend

 **WARNING:** A WARNING icon highlights a potential risk of bodily injury or property damage, for which industry-standard safety precautions are advised. This icon is often associated with electrical hazards related to hardware.

 **CAUTION:** A CAUTION icon indicates potential damage to hardware or loss of data if instructions are not followed.

One Identity Manager REST API Reference Guide
Updated - 17 October 2024, 09:37

For the most recent documents and product information, see [Online product documentation](#).

Contents

Guidelines and conventions	5
BaseURL	5
Parameter formats	5
Path parameters	5
Query parameters	6
HTTP request methods	7
Request formats	7
Response formats	7
HTTP response codes	7
Authentication	8
Initial data for authentication modules	9
Identifying activated authentication modules	9
Getting details of an authentication module	12
Login	13
Logout	14
Date formats	15
Object identifiers	15
Using offset and limit parameters to cycle through records	15
Session object global variables	16
Set global variable	17
Clear global variable	18
Set global variables	18
Clear global variables	19
Running the Application Server in REST API only mode	20
Disabling WHERE clauses for the REST API of the Application Server	20
Collections	22
Get collection using GET method	22
Get collection using POST method	26
Get count of collection using GET method	33
Get count of collection using POST method	34
Single entities	36

Create a single entity	36
Create a batch of single entities	38
Get a single entity	41
Change a single entity	43
Change a batch of single entities	44
Delete a single entity	48
Delete a batch of single entities	49
Call a method on an entity	50
Generate an event for an entity	52
Assignments	55
Get assignments for a specific entity	55
Add assignments	58
Remove assignments	60
Scripts	62
Run script request	62
Appendix: PowerShell sample	66
About us	68
Contacting us	68
Technical support resources	68

Guidelines and conventions

This document provides development information for customers and partners intending to use One Identity Manager REST APIs.

The initial sections provide general guidelines and conventions for reference.

BaseURL

The One Identity Manager REST API is an integral part of the One Identity Manager application server. The BaseURL specifies the path to an application server installation. By default, the application server is installed with the following path.

```
https://<Hostname>/AppServer
```

Parameter formats

HTTP requests use two types of parameters:

- [Path parameters](#) on page 5
- [Query parameters](#) on page 6

Path parameters

Path parameters continue the URI path using a slash for a separator. For example, to get details for an identity, you specify their primary key (GUID) with a path parameter. The following shows the URI format for this request:

```
<BaseURL>/api/entity/{table}/{uid}
```

Example:

```
https://<Hostname>/AppServer/api/entity/Person/25d87790-105d-4afb-bd04-cfddc7cc9fb5
```

If a request uses path parameters, they are specified in the URI format for the request.

Query parameters

Query parameters are included in the URI path using a question mark or ampersand. For example, to receive a list of identities ordered by their surname, after specifying the table Person as a path parameter, you specify an orderBy as a query parameter. The following shows the URI format for this request:

```
<BaseUrl>/api/entities/{table}
```

Example:

```
https://<Hostname>/AppServer/api/entities/person?orderBy=LastName
```

The first query parameter is preceded by a question mark, using the following format:

```
?parameterName=parameterValue
```

Subsequent query parameters are preceded by an ampersand, using the following format:

```
&parameterName=parameterValue
```

For example, to get all identities who are marked as external ordered by their company membership, use the following request:

```
https://<Hostname>/AppServer/api/entities/person?where=IsExternal%3D0&orderBy=CompanyMember
```

NOTE: If a parameter is not shown in a request's URI format with a slash, it is a query parameter; the URI format for a request shows only the path parameters. The description that follows each URI format provides information on the query parameters, if any.

Query parameters to be used in WHERE clauses and predefined WHERE clauses can be provided in the form @paramname=value. Per default they have the type string. Additionally, it is possible to pass the type with the name: @otherParam[int]=3. Types originate from DbVal class.

HTTP request methods

Depending on the HTTP request, use one of the following HTTP request methods:

- **Get:** Use for requests that retrieve elements from the application server.
- **Put:** Use for requests that change elements on the application server.
- **Post:** Use for requests that create an element on the application server.
- **Delete:** Use for request that delete an element on the application server.

The request methods Put and Post will also be used if:

- The number or the size of the parameters would lead to problems in the URL.
- The type of parameter would lead to problems in the URL.

Each request description specifies which HTTP request method to use.

To use these request methods on an application server they must be permitted by the web server.

Request formats

Request body values have to be formatted as JSON and the request content type **application/json** has to be set.

Response formats

Code-issued requests to the API should always return JSON, based on the request headers.

To return JSON output

- Set the Accept header in the request to **application/json**.

HTTP response codes

Responses from the REST API use the codes listed below. When method executions fail, a descriptive error message is displayed.

Table 1: HTTP response codes

Response status codes	Description
200	Success.
204	Success. No content returned.
401	Unauthorized. To use the One Identity Manager REST API, you first have to authenticate it against the application server.
404	Not found. The requested entity is not found.
405	Method not allowed. The HTTP request method that was specified is not the correct method for the request.
500	<p>Internal server error. The error message is returned in the property error string of the response.</p> <pre>{ "responseStatus": { "message": "Sample text"}, "errorString": "Sample text", "exceptions": [{ "number": 810017, "message": "Sample text"}] }</pre> <p>Due to security, the detailed error message will not be returned to the caller. More information can be retrieved from the application server log on the application server.</p>

Authentication

One Identity Manager uses different authentication modules for logging into the REST API. Authentication modules identify the system users to be used and load the user interface and database resource editing permissions depending on their permission group memberships.

NOTE:

- After initial schema installation, only the **System user** and **Component Authenticator** authentication modules and the role-based authentication modules are enabled in One Identity Manager.

- Authentication modules are defined in the modules and are not available until One Identity Manager modules are installed.
- To access the REST API from external applications you can use the **OAuth 2.0/OpenID Connect** and **OAuth 2.0/OpenID Connect (rolebased)** authentication modules. For more detailed information, see the *One Identity Manager Authorization and Authentication Guide*.
- To access the REST API in the application server, users need the program function **Enables access to the REST API in the application server** (AppServer_API).

Related topics

- [Initial data for authentication modules](#) on page 9
- [Identifying activated authentication modules](#) on page 9
- [Getting details of an authentication module](#) on page 12
- [Login](#) on page 13
- [Logout](#) on page 14

Initial data for authentication modules

The authentication string is formatted as follows:

Module=<name>;<property1>=<value1>;<property2>=<value2>,...

Example:

```
Module=DialogUser;User=<user name>;Password=<password>
```

The initial data is one part of the authentication string (parameter-value pair without module ID). Initial data from the authentication string is pre-allocated by default for each authentication instance. Some authentication modules are not requiring any parameter besides specifying the authentication module.

For more information about authentication modules, see the *One Identity Manager Authorization and Authentication Guide*.

Identifying activated authentication modules

The list of supported, respectively activated authentication modules can be retrieved using the URL <BaseURL>/appserver/authmodules.

Table 2: List authentication modules request

HTTP method	URI	Body
Get	<BaseURL>/appserver/authmodules	None

Response schema:

```
{
  "id": String,
  "caption": String,
  "passwordBased": Boolean,
  "isDefault": Boolean
}
```

Example:

https://<Hostname>/AppServer/appserver/authmodules

Response:

```
[{
  "id": "RoleBasedManualADS",
  "caption": "Active Directory user account (manual input/role based)",
  "passwordBased": false,
  "isDefault": false
},
{
  "id": "RoleBasedADSAccount",
  "caption": "Active Directory user account (role based)",
  "passwordBased": false,
  "isDefault": false
},
{
  "id": "DialogUser",
  "caption": "System user",
  "passwordBased": false,
```

```

        "isDefault": true
    },
    {
        "id": "RoleBasedPerson",
        "caption": "Identity (role based)",
        "passwordBased": false,
        "isDefault": false
    },
    {
        "id": "OAuthRoleBased",
        "caption": "OAuth 2.0 (role based)",
        "passwordBased": false,
        "isDefault": false
    },
    {
        "id": "OAuth",
        "caption": "OAuth 2.0",
        "passwordBased": false,
        "isDefault": false
    },
    {
        "id": "ADSAccount",
        "caption": "Active Directory user account",
        "passwordBased": false,
        "isDefault": false
    },
    {
        "id": "DynamicPerson",
        "caption": "Identity (dynamic)",
        "passwordBased": false,
        "isDefault": false
    }
}

```

Getting details of an authentication module

To get the details for a specific authentication module, use the URL `<baseUrl>/appserver/authmodules/{id}`.

Table 3: Get authentication module request

HTTP method	URI	Body
Get	<code><BaseUrl>/appserver/authmodules/{id}</code>	None

Table 4: Get authentication module parameters

Parameter	Description	Parameter type	Data type
id	Authentication module (required)	path	string

Response schema:

```
[{
  "id": String,
  "caption": String,
  "authTemplate": String,
  "passwordBased": Boolean,
  "isDefault": Boolean
}]
```

Example:

`https://<Hostname>/AppServer/appserver/authmodules/DialogUser`

Response:

```
[{
  "id": "DialogUser",
  "caption": "System user",
  "authTemplate": "Module=DialogUser;User[VI.DB_USER]=;(Password)Password
[VI.DB_Password]=",
  "passwordBased": true,
  "isDefault": false
}]
```

```
}]
```

The values in the property `authTemplate` can be used to identify the format of the `authString` needed to authenticate against the application server. You can ignore the parts in `[]` and `()` as those are the caption keys and value types used in the front ends only.

For example, a valid authentication string would be `Module=DialogUser;User=MyUser;Password=$secret`.

Login

To use the One Identity Manager REST API, you first have to authenticate against the application server.

Table 5: Authenticate request

HTTP method	URI	Body
Post	<BaseURL>/auth/apphost	Property <code>authString</code> containing an authentication string. For more information, see Initial data for authentication modules on page 9.

Body schema:

```
{"authString": String}
```

Response schema:

```
{
  "claims": {
    "id": String,
    "userid": String,
    "module": String
  },
  "passwordBased": Boolean,
  "moduleDisplay": String,
  "sessionId": String,
  "userName": String
}
```

Example:

https://<Hostname>/AppServer/auth/apphost

Body:

```
{"authString": "Module=DialogUser;User=<user name>;Password="}
```

Response:

```
{
  "claims": {
    "id": "<user name>",
    "userid": "QBM-A60F9E5189134AFFB6711DFCBC3F260E",
    "module": "DialogUser"
  },
  "passwordBased": true,
  "moduleDisplay": "System user",
  "sessionId": "nV8R3iw4KfmEiZydA3uy",
  "userName": "<user name>"
}
```

Logout

If you want to end your session against the One Identity Manager REST API you can use the logout request.

Table 6: Logout request

HTTP method	URI	Body
Post	<BaseURL>/auth/logout	

Response schema:

```
{}
```

Example:

```
https://<Hostname>/AppServer/auth/logout
```

Response:

```
{}
```

Date formats

If date values have to be specified in requests for changing or adding objects using the REST API, these have to be specified in the ISO 8601 format in UTC.

Example:

```
2016-03-19T13:09:08.123Z, which is March 19, 2016, 1:09:08.123 PM UTC
```

Object identifiers

The requests and responses use identifiers for identifying the objects from One Identity Manager. Every time an object is created, the system internally generates a globally unique identifier (GUID). These GUIDs can be used to fetch single objects directly using the API.

Using offset and limit parameters to cycle through records

Some requests result in responses that contain many records. For example, the request for the list of all identities can match hundreds or thousands of identities.

To limit the number of records returned, some of the APIs support the limit and offset query parameters. These parameters allow you to get successive sets of records in successive responses. Specifically, use these query parameters to do the following:

- Limit the number of records returned in the response to a number you choose, using the query parameter limit.

For example, the following request returns the first 50 identities:

```
https://<Hostname>/AppServer/api/entities/person?&limit=50
```

- Specify the index of the first record to return in the response, using the query parameter offset.

The value is zero-based. For example, the following request returns 50 identities, starting with the 101st identity:

```
https://<Hostname>/AppServer/api/entities/person?&limit=50&offset=100
```

The offset parameter defaults to 0. Therefore, both of the following requests return 50 devices, starting with the first device:

```
https://<Hostname>/AppServer/api/entities/person?&limit=50&offset=0
```

```
https://<Hostname>/AppServer/api/entities/person?&limit=50
```

Therefore, to get successive sets of records in successive responses, increase the offset value by the limit value in each request. For example:

```
https://<Hostname>/AppServer/api/entities/person?&limit=50&offset=0
```

```
https://<Hostname>/AppServer/api/entities/person?&limit=50&offset=50
```

```
https://<Hostname>/AppServer/api/entities/person?&limit=50&offset=100
```

The following table summarizes the limit and offset query parameters.

Table 7: Pagination parameters

Query parameter	Description	Default value
limit	Maximum number of records to show in the response.	0
offset	Zero-based index of first record to show in the response.	0

Session object global variables

Global variables are allocated by the set up program. All environment variables and custom variables defined for the session object can be used in addition to predefined variables. Custom session variables can be defined, for example, through scripts, methods, or customizers.

NOTE: If a custom session variable is defined, it must be removed again afterward. Otherwise it remains for the rest of the session and, in certain circumstances, the wrong processes can be generated.

NOTE: New API methods to handle the global variables have been added to the /api path of the Application Server starting with One Identity Manager version 8.2. This allows to access the global variables even if the Application Server is restricted to the pure REST API only. For more information, see [Running the Application Server in REST API only mode](#) on page 20.

Related topics

- [Set global variable](#) on page 17
- [Clear global variable](#) on page 18
- [Set global variables](#) on page 18
- [Clear global variables](#) on page 19

Set global variable

To set a global variable, use the URL `<baseUrl>/api/variable/{name}`.

Table 8: Set global variable request

HTTP method	URI	Body
Put	<code><baseUrl>/api/variable/{name}</code>	<code>{"value": <value>}</code>

Table 9: Set global variable parameters

Parameter	Description	Parameter type	Data type
name	Variable name (required).	path	string

Body schema:

`{value (object): Content of the variable.}`

Example:

`https://<Hostname>/AppServer/api/variable/FullSync`

Body:

`{"value": true}`

Clear global variable

To clear a global variable, use the URL `<baseUrl>/api/variable/{name}`.

Table 10: Clear global variable request

HTTP method	URI	Body
Put	<code><baseUrl>/api/variable/{name}</code>	<code>{"clear": true}</code>

Table 11: Clear global variable parameters

Parameter	Description	Parameter type	Data type
name	Variable name (required).	path	string

Body schema:

```
{clear (boolean): True to clear a variable.}
```

Example:

```
https://<Hostname>/AppServer/api/variable/FullSync
```

Body:

```
{"clear": true}
```

Set global variables

To set a set of global variables, use the URL `<baseUrl>/api/variables`.

Table 12: Set a set of global variable request

HTTP method	URI	Body
Put	<code><baseUrl>/api/variables</code>	<code>{"variables": { "variablename": <value> } }</code>

Table 13: Set global variable parameters

Parameter	Description	Parameter type	Data type
variables	Variable names (required).	body	string

Body schema:

```
{"variables": {  
  "variablename": <value>  
}}
```

Example:

https://<Hostname>/AppServer/api/variables

Body:

```
{"variables": {"FullSync": True, "MyVariable": 42}}
```

Clear global variables

To clear a set of global variables, use the URL <baseURL>/api/variables/clear.

Table 14: Clear a set of global variable request

HTTP method	URI	Body
Put	<baseURL>/api/variables/clear	{"variables": ["string"]}

Table 15: Clear global variable parameters

Parameter	Description	Parameter type	Data type
variables	Variable names (required)	body	string

Body schema:

```
{  
  "variables": [  
    "string"  
  ]  
}
```

Example:

```
https://<Hostname>/AppServer/api/variables/clear
```

Body:

```
{"variables": ["FullSync", "MyVariable"]}
```

Running the Application Server in REST API only mode

The Application Server can be restricted to the pure REST API only. To do this, modify the `Web.config` file. All services necessary for login and status page are not affected by the `switch-off`.

Settings for the Application Server plugins in the `Web.config` file

```
<restapi>  
  <!-- Switch REST API part off -->  
  <!--<add key="off" value="true" />-->  
</restapi>  
<appserverapi>  
  <!-- Switch application server APIs off -->  
  <!--<add key="off" value="true" />-->  
</appserverapi>
```

Disabling WHERE clauses for the REST API of the Application Server

You have the option to disable the WHERE clauses or even predefined WHERE clauses through the `Web.config` file.

Settings for WHERE clauses and predefined WHERE clauses in the `Web.config` file

```
<restapi>
```

```
...
<!-- Plain WHERE clauses as parameters are forbidden -->
<!--<add key="nowhereclause" value="true" />-->
<!-- Using predefined WHERE clauses as parameter is forbidden -->
<!--<add key="nolimitedsql" value="true" />-->
</restapi>
```

Collections

To get a list of entities, you have the option to use a GET or POST request against the REST API. Both methods support to query by example. Simply provide the columns to query in the form Name=Value in the URL.

Detailed information about this topic

- [Get collection using GET method](#) on page 22
- [Get collection using POST method](#) on page 26
- [Get count of collection using GET method](#) on page 33
- [Get count of collection using POST method](#) on page 34

Get collection using GET method

To get a list of entities using the GET method, use the URL `<baseUrl>/api/entities/{table}`.

Query parameters to be used in WHERE clauses and predefined WHERE clauses can be provided in the form `@paramname=value`. Per default they have the type `string`. Additionally, it is possible to pass the type with the name: `@otherParam[int]=3`. Types originate from `DbVal` class.

Table 16: Get collection (GET) request

HTTP method	URI	Body
Get	<code><BaseUrl>/api/entities/{table}</code>	None

Table 17: Get collection (GET) parameters

Parameter	Description	Parameter type	Data type
table	Table name (required).	path	string
where	WHERE clause.	query	string
whereKey	Predefined WHERE clause from QBMLimitedSQL (Key [UID] or ID [Ident_ QBMLimitedSQL]).	query	string
orderBy	ORDER BY clause.	query	string
offset	Offset of first item.	query	integer
limit	Maximum number of results.	query	integer
displayColumns	Additional display columns, semicolon separated.	query	string
loadType	Collection load type. Specify one of the values: <ul style="list-style-type: none">• Default: Loads read-only entities according to the supplied query. Loaded columns include the primary key, display columns according to the display pattern, some special columns, and the columns defined in the select clause of the query. The entries are sorted by the defined display or the optional orderBy clause of the query. This load type is the default and be omitted.• Slim: Works mostly like Default but does not load display columns and does not build an orderBy clause per default. This type is useful when loading data not intended for display and can save much time by using database indexes.• BulkReadOnly: Loads read-only entities with all columns filled. The columns defined in the query are overridden.• ForeignDisplays: Loads display values for foreign keys contained in the display pattern. This allows	query	string

Parameter	Description	Parameter type	Data type
	<p>showing displays instead of UIDs for foreign keys.</p> <ul style="list-style-type: none"> • ForeignDisplaysForAllColumns: Like ForeignDisplays, but loads displays for all foreign keys contained in the <code>select</code> clauses of the query, not only columns referenced in the display pattern. 		
noUrls	Allow to omit the URI property from the response to reduce the response size especially if used while retrieving large M:N tables like <code>PersonHasObject</code> .	query	boolean

Response schema:

```

CollectionEntry {
    uri(string),
    display(string, optional),
    longDisplay(string, optional),
    values(SampleValues, optional)
}
SampleValues {
    StringColumn(string, optional),
    IntColumn(integer, optional),
    DateColumn(date - time, optional),
    BoolColumn (boolean, optional)
}

```

Example 1:

This sample demonstrates the use of the query by example parameters.

`https://<Hostname>/AppServer/api/entities/Person?lastname=adams&limit=2`

Response:

```
[{
```

```

"uri": "https://<Hostname>/AppServer/api/entity/Person/7f6bcca9-05dc-4857-9dc5-eff915590752",
"display": "Adams, Alexander (ALEXANDERA)",
"longDisplay": "Adams, Alexander (ALEXANDERA)",
"values": {
  "CentralAccount": "ALEXANDERA",
  "InternalName": "Adams, Alexander",
  "UID_Person": "7f6bcca9-05dc-4857-9dc5-eff915590752",
  "XMarkedForDeletion": 0
}},
{
"uri": "https://<Hostname>/AppServer/api/entity/Person/f79c30fd-87bb-4958-a812-0683ddcac7c9",
"display": "Adams, David (DAVIDA)",
"longDisplay": "Adams, David (DAVIDA)",
"values": {
  "CentralAccount": "DAVIDA",
  "InternalName": "Adams, David",
  "UID_Person": "f79c30fd-87bb-4958-a812-0683ddcac7c9",
  "XMarkedForDeletion": 0
}
}]

```

Example 2:

This sample demonstrates the use of the loadType=Slim.

<https://<Hostname>/AppServer/api/entities/Person?lastname=adams&limit=2&loadType=Slim>

Response:

```
[{
```

```

"uri": "https://<Hostname>/AppServer/api/entity/Person/18e51519-f751-4df6-8f39-24ed065c80a9",
"values": {
  "UID_Person": "18e51519-f751-4df6-8f39-24ed065c80a9"
}},
{
"uri": "https://<Hostname>/AppServer/api/entity/Person/26822a10-32bb-4268-ac59-36188301b768",
"values": {
  "UID_Person": "26822a10-32bb-4268-ac59-36188301b768"
}
}
}]

```

Example 3:

This sample demonstrates the use of the parameter `noUrls=true` while using the same parameters as in example 2.

`https://<Hostname>/AppServer/api/entities/Person?lastname=adams&limit=2&loadType=Slim&noUrls=true`

Response:

```

[[{
  "values": {
    "UID_Person": "18e51519-f751-4df6-8f39-24ed065c80a9"
  }
},
{
  "values": {
    "UID_Person": "26822a10-32bb-4268-ac59-36188301b768" }
}
]]

```

Get collection using POST method

To get a list of entities using the POST method, use the URL `<baseUrl>/api/entities/{table}`.

Table 18: Get collection (GET) request

HTTP method	URI	Body
Get	<BaseURL>/api/entities/{table}	{"where": "", "orderBy": ""}

Table 19: Get collection (GET) parameters

Parameter	Description	Parameter type	Data type
table	Table name (required).	path	string
where	WHERE clause.	body	string
whereKey	Predefined WHERE clause from QBMLimitedSQL (Key [UID] or ID [Ident_QBMLimitedSQL]).	body	string
queryParameters	Parameters to be used in where or whereKey clause.	body	string
queryByExample	Sample values to filter the entries.	body	string
orderBy	ORDER BY clause.	body	string
offset	Offset of first item.	body	integer
limit	Maximum number of results.	body	integer
displayColumns	Additional display columns, semicolon separated.	body	string
loadType	Collection load type. Specify one of the values: <ul style="list-style-type: none"> • Default: Loads read-only entities according to the supplied query. Loaded columns include the primary key, display columns according to the display pattern, some special columns, and the columns defined in the select clause of the query. The entries are sorted by the defined display or the optional orderBy clause of the query. This load type is the default and be omitted. • Slim: Works mostly like Default but does not load display columns and does not build an orderBy clause per default. This type is useful when loading data not intended for display and can save 	body	string

Parameter	Description	Parameter type	Data type
	<p>much time by using database indexes.</p> <ul style="list-style-type: none"> • BulkReadOnly: Loads read-only entities with all columns filled. The columns defined in the query are overridden. • ForeignDisplays: Loads display values for foreign keys contained in the display pattern. This allows showing displays instead of UIDs for foreign keys. • ForeignDisplaysForAllColumns: Like ForeignDisplays, but loads displays for all foreign keys contained in the select clauses of the query, not only columns referenced in the display pattern. 		
noUrls	Allow to omit the URI property from the response to reduce the response size especially if used while retrieving large M:N tables like PersonHasObject.	body	boolean

Body schema:

```

CollectionQueryParms{
  where string
    WHERE clause.
  whereKey string
    Predefined WHERE clause from QBMLimitedSQL.
  orderBy string
    ORDER BY clause.
  offset integer
    Offset of first item.
  limit integer($int32)
    Maximum number of results.
  displayColumns [...]
  loadType string
    default: Default

```

```

    Collection load type.
    Enum:
    Array [ 6 ]
  queryParameters {
    Parameters to be used in where or whereKey clause.
  }
  queryByExample {
    Sample values to filter the entries.
  }
  noUrls boolean
    Do not return URLs for the entries.
}

```

Response schema:

```

CollectionEntry {
  uri(string),
  display(string, optional),
  longDisplay(string, optional),
  values(SampleValues, optional)
}
SampleValues {
  StringColumn(string, optional),
  IntColumn(integer, optional),
  DateColumn(date - time, optional),
  BoolColumn (boolean, optional)
}

```

Example: Body

```

{
  "where": "string",
  "whereKey": "string",
  "orderBy": "string",
  "offset": 0,
  "limit": 0,
}

```

```

    "displayColumns": [
      "string"
    ],
    "loadType": "Default",
    "queryParameters": {
      "lastname": "Miller",
      "age": 42
    },
    "queryByExample": {
      "lastname": "Einstein",
      "firstname": "Albert"
    },
    "noUrls": true
  }

```

Example 1:

This sample demonstrates the use of the `where` and `orderBy` parameters in the body.

`https://<Hostname>/AppServer/api/entities/Person?limit=2`

Body:

```

{
  "where": "UID_Department in (Select UID_Department from Department
  where DepartmentName = 'Service & Support')",
  "orderBy": "LastName ASC, FirstName DESC"
}

```

Response:

```

[ {
  "uri": "https://<Hostname>/AppServer/api/entity/Person/20bac746-2121-
  4b24-a4dc-918b69584272",
  "display": "Ackermann, Steffen (STEFFENA)",

```

```

    "longDisplay": "Ackermann, Steffen (STEFFENA)",
    "values": {
      "CentralAccount": "STEFFENA",
      "FirstName": "Steffen",
      "InternalName": "Ackermann, Steffen",
      "LastName": "Ackermann",
      "UID_Person": "20bac746-2121-4b24-a4dc-918b69584272",
      "XMarkedForDeletion": 0
    }
  },
  {
    "uri": "https://<Hostname>/AppServer/api/entity/Person/f45092af-4725-4f99-b87c-00de84b7dcd7",
    "display": "Becker, Robert (ROBERTB4)",
    "longDisplay": "Becker, Robert (ROBERTB4)",
    "values": {
      "CentralAccount": "ROBERTB4",
      "FirstName": "Robert",
      "InternalName": "Becker, Robert",
      "LastName": "Becker",
      "UID_Person": "f45092af-4725-4f99-b87c-00de84b7dcd7",
      "XMarkedForDeletion": 0
    }
  }
}
]

```

Example 2:

This sample demonstrates the use of the parameter `noUrls=true` while using the same parameters as in example 1.

`https://<Hostname>/AppServer/api/entities/Person?limit=2&noUrls=true`

Body:

```
{
```

```
"where": "UID_Department in (Select UID_Department from Department
where DepartmentName = 'Service & Support')",
"orderBy": "LastName ASC, FirstName DESC"
}
```

Response:

```
[{
  "display": "Ackermann, Steffen (STEFFENA)",
  "longDisplay": "Ackermann, Steffen (STEFFENA)",
  "values": {
    "CentralAccount": "STEFFENA",
    "FirstName": "Steffen",
    "InternalName": "Ackermann, Steffen",
    "LastName": "Ackermann",
    "UID_Person": "20bac746-2121-4b24-a4dc-918b69584272",
    "XMarkedForDeletion": 0
  }
},
{
  "display": "Becker, Robert (ROBERTB4)",
  "longDisplay": "Becker, Robert (ROBERTB4)",
  "values": {
    "CentralAccount": "ROBERTB4",
    "FirstName": "Robert",
    "InternalName": "Becker, Robert",
    "LastName": "Becker",
    "UID_Person": "f45092af-4725-4f99-b87c-00de84b7dcd7",
    "XMarkedForDeletion": 0
  }
}]
```

Get count of collection using GET method

To get only the count for a list of entities using the GET method, use the URL `<baseUrl>/api/entities/{table}/count`.

Query parameters to be used in WHERE clauses and predefined WHERE clauses can be provided in the form `@paramname=value`. Per default they have the type `string`. Additionally, it is possible to pass the type with the name: `@otherParam[int]=3`. Types originate from `DbVal` class.

Table 20: Get count of collection (GET) request

HTTP method	URI	Body
Get	<code><BaseUrl>/api/entities/{table}/count</code>	None

Table 21: Get count of collection (GET) parameters

Parameter	Description	Parameter type	Data type
table	Table name (required).	path	string
where	WHERE clause.	query	string
whereKey	Predefined WHERE clause from <code>QBMLimitedSQL</code> (Key [UID] or ID [Ident_QBMLimitedSQL]).	query	string

Response schema:

`Count(integer)`

Example:

This sample demonstrates the use of the query by example parameters.

`https://<Hostname>/AppServer/api/entities/Person/count?lastname=adams`

Response:

2

Get count of collection using POST method

To get only the count for a list of entities using the POST method, use the URL `<baseUrl>/api/entities/{table}/count`.

Table 22: Get count of collection (POST) request

HTTP method	URI	Body
Get	<code><BaseUrl>/api/entities/{table}/count</code>	<code>{"where": ""}</code>

Table 23: Get count of collection (POST) parameters

Parameter	Description	Parameter type	Data type
table	Table name (required).	path	string
where	WHERE clause.	body	string
whereKey	Predefined WHERE clause from QBMLimitedSQL (Key [UID] or ID [Ident_ QBMLimitedSQL]).	body	string
queryParameters	Parameters to be used in where or whereKey clause.	body	string
queryByExample	Sample values to filter the entries.	body	string

Body schema:

```
CollectionQueryCountParams{
  where string
    WHERE clause.
  whereKey string
    Predefined WHERE clause from QBMLimitedSQL.
  queryParameters {
    Parameters to be used in where or whereKey clause.
  }
  queryByExample {
    Sample values to filter the entries.
  }
}
```

Response schema:

Count(integer)

Example: Body

```
{
  "where": "string",
  "whereKey": "string",
  "queryParameters": {
    "lastname": "Miller",
    "age": 42
  },
  "queryByExample": {
    "lastname": "Einstein",
    "firstname": "Albert"
  }
}
```

Example:

https://<Hostname>/AppServer/api/entities/Person/count

Body:

```
{"where": "LastName=adams"}
```

Response:

2

Single entities

The following APIs are used to handle entities of One Identity Manager. You can create, read, update, and delete single entities as well as call methods of an entity and generate events.

Detailed information about this topic

- [Create a single entity](#) on page 36
- [Create a batch of single entities](#) on page 38
- [Get a single entity](#) on page 41
- [Change a single entity](#) on page 43
- [Change a batch of single entities](#)
- [Delete a single entity](#) on page 48
- [Delete a batch of single entities](#) on page 49
- [Call a method on an entity](#) on page 50
- [Generate an event for an entity](#) on page 52

Create a single entity

To create a single entity, use the URL `<baseUrl>/api/entity/{table}`.

Table 24: Create single entity request

HTTP method	URI	Body
Post	<code><BaseURL>/api/entity/{table}</code>	<code>{"values": { "StringColumn": "string", "IntColumn": 0, "DateColumn": "2016-05-</code>

HTTP method	URI	Body
		19T11:21:33.579Z", "BoolColumn": True }

Table 25: Create single entity parameters

Parameter	Description	Parameter type	Data type
table	Table name (required).	path	string
values	Values to set.	body	SampleValues

Body schema:

```
SingleChangeBody {
    values(SampleValues, optional)
}
SampleValues {
    StringColumn(string, optional),
    IntColumn(integer, optional),
    DateColumn(date - time, optional),
    BoolColumn (boolean, optional)
}
```

Response schema:

```
CreateSingleResult {
    uid (string, optional),
    uri (string, optional)
}
```

Example:

https://<Hostname>/AppServer/api/entity/Person

Body:

```
{
```

```

    "values": {
      "FirstName": "Jeremia",
      "LastName": "Bodewell",
      "IsExternal": True,
      "BirthDate": "1993-05-14",
      "Gender": 1
    }
  }
}

```

Response:

```

{
  "uid": "83b10e84-c64e-4f9f-9ecb-2d0d7c94e8ec",
  "uri": "https://<Hostname>/AppServer/api/entity/Person/83b10e84-c64e-4f9f-9ecb-2d0d7c94e8ec"
}

```

Create a batch of single entities

To create a batch single entity, use the URL `<baseUrl>/api/entities`.

Table 26: Create a batch of single entities request

HTTP method	URI	Body
Post	<code><BaseURL>/api/entities</code>	<pre> { "entities": [{ "table": "string", "values": { "StringColumn": "string", "IntColumn": 0, "DateColumn": "2016-0519T11:21:33.579Z", "DoubleColumn": 3.14, "BoolColumn": true } }] } </pre>

HTTP method	URI	Body
		} }] }

Table 27: Create a batch of single entities parameters

Parameter	Description	Parameter type	Data type
entities	List of entities to create (required).	Body	array[InsertEntity]

Body schema:

```
BulkChangeBodyInsert {
    entities(array[InsertEntity])
}
InsertEntity {
    table(string),
    values(SampleValues, optional)
}
SampleValues {
    StringColumn(string, optional),
    IntColumn(integer, optional),
    DateColumn(date - time, optional),
    BoolColumn (boolean, optional)
}
```

Response schema:

```
CreateBulkResult {
    entities(Array[UidAndUri])
}
UidAndUri {
    uid (string, optional),
    uri (string, optional)
}
```

Example:

https://<Hostname>/AppServer/api/entities

Body:

```
{
  "entities": [
    {
      "table": "Person",
      "values": {
        "FirstName": "Jeremia",
        "LastName": "Bodewell",
        "IsExternal": True,
        "BirthDate": "1993-05-14",
        "Gender": 1
      }
    },
    {
      "table": "Person",
      "values": {
        "FirstName": "Rose",
        "LastName": "Gladstone",
        "IsExternal": False,
        "BirthDate": "1991-03-21",
        "Gender": 2
      }
    }
  ]
}
```

Response:

```
{
  "entities": [
```

```

    {
      "uid": "8a2eee88-0615-4595-a903-8388ca74d877",
      "uri": "/AppServer/api/entity/Person/8a2eee88-0615-4595-a903-8388ca74d877"
    },
    {
      "uid": "df563a8c-203a-4132-8fcb-bc4adaee35b6",
      "uri": "/AppServer/api/entity/Person/df563a8c-203a-4132-8fcb-bc4adaee35b6"
    }
  ]
}

```

Get a single entity

To get a single entity, use the URL `<baseUrl>/api/entity/{table}/{uid}`.

Table 28: Get single entity request

HTTP method	URI	Body
Get	<code><BaseUrl>/api/entity/{table}/{uid}</code>	None

Table 29: Get single entity parameters

Parameter	Description	Parameter type	Data type
table	Table name (required).	path	string
uid	GUID of this entity (required).	path	string

Response schema:

```

SingleEntry {
  uri (string),
  uid (string, optional),
  display (string, optional),
  values (SampleValues, optional),
  links (Array[Link], optional)
}

```

```

SampleValues {
    StringColumn (string, optional),
    IntColumn (integer, optional),
    DateColumn (date-time, optional),
    BoolColumn (boolean, optional)
}
Link {
    name (string, optional)
}

```

Example:

https://<Hostname>/AppServer/api/entity/Person/83b10e84-c64e-4f9f-9ecb-2d0d7c94e8ec

Response:

```

{
  "uri": "https://<Hostname>/AppServer/api/entity/Person/83b10e84-c64e-4f9f-9ecb-2d0d7c94e8ec",
  "display": "Bodewell, Jeremia (JEREMIAB)",
  "values": {
    "ApprovalState": 0,
    "AuthenticatorLogins": "",
    "BirthDate": "1993-05-14T00:00:00.000000Z",
    "Building": "",
    "CanonicalName": "",
    "CentralAccount": "JEREMIAB",
    "CentralPassword": "",
    "CentralSAPAccount": "BODEWELJ",
    ...
    "XObjectKey": "<Key><T>Person</T><P>83b10e84-c64e-4f9f-9ecb-2d0d7c94e8ec</P></Key>",
    "XTouched": "",
    "XUserInserted": "<user name>",
  }
}

```

```

        "XUserUpdated": "<user name>",
        "ZIPCode": ""
    }
}

```

Change a single entity

To change a single entity, use the URL `<baseURL>/api/entity/{table}/{uid}`.

Table 30: Change single entity request

HTTP method	URI	Body
Put	<code><BaseURL>/api/entity/{table}/{uid}</code>	<pre> {"values": { "StringColumn": "string", "IntColumn": 0, "DateColumn": "2016-05-19T11:21:33.579Z", "BoolColumn": true } </pre>

Table 31: Change single entity parameters

Parameter	Description	Parameter type	Data type
table	Table name (required).	path	string
uid	GUID of the entity (required).	path	string
values	Values to change.	body	SampleValues

Body schema:

```

SingleChangeBody {
    values(SampleValues, optional)
}
SampleValues {
    StringColumn(string, optional),
    IntColumn(integer, optional),

```

```
DateColumn(date - time, optional),
BoolColumn (boolean, optional)
}
```

Response schema:

```
{}
```

Example:

<https://<Hostname>/AppServer/api/entity/Person/83b10e84-c64e-4f9f-9ecb-2d0d7c94e8ec>

Body:

```
{
  "values": {
    "LastName": "Garibaldi",
    "IsExternal": false,
    "UID_Locality": "83615878-7205-408d-a5fa-f260840c867c"
  }
}
```

Response Code:

200

Response:

```
{}
```

Change a batch of single entities

To change a single entity, use the URL `<baseUrl>/api/entities`.

Table 32: Change a batch of single entities request

HTTP method	URI	Body
Post	<BaseURL>/api/entities	<pre>{ "entities": [{ "table": "string", "uid": "string", "values": { "StringColumn": "string", "IntColumn": 0, "DateColumn": "2016-0519T11:21:33.579Z", "DoubleColumn": 3.14, "BoolColumn": true } }], "insertMissing": false }</pre>

Table 33: Change a batch of single entities parameters

Parameter	Description	Parameter type	Data type
entities	List of entities to create (required).	Body	array [UpdateEntity]
insertMissing	Insert entities that could not be found using the supplied keys.	Body	Boolean

Body schema:

```
BulkChangeBodyUpdate {
  entities(Array[UpdateEntity]),
  insertMissing(boolean,default: false)
}
UpdateEntity {
  table(string),
```

```

        uid(string),
        values(SampleValues, optional)
    }
    SampleValues {
        StringColumn(string, optional),
        IntColumn(integer, optional),
        DateColumn(date - time, optional),
        BoolColumn (boolean, optional)
    }
}

```

Response Code:

204

Example 1:

This sample demonstrates the change of existing entities while omitting the `insertMissing` parameter in the body.

`https://<Hostname>/AppServer/api/entities`

Body:

```

{
  "entities": [
    {
      "table": "Person",
      "uid": "8a2eee88-0615-4595-a903-8388ca74d877",
      "values": {
        "LastName": "Coleman",
        "IsExternal": false,
        "ExitDate": "2032-12-31"
      }
    },
    {
      "table": "Person",
      "uid": "df563a8c-203a-4132-8fcb-bc4adabb35b6",

```

```

    "values": {
      "FirstName": "Rosalie",
      "LastName": "Meyer",
      "IsExternal": True,
      "BirthDate": "1991-03-21",
      "Gender": 2,
      "ExitDate": "2032-12-31",
    }
  ]
}

```

Response Code:

204

Example 2:

This sample demonstrates the change of existing entities using the `insertMissing` parameter in the body to create non-existing entities.

<https://<Hostname>/AppServer/api/entities>

Body:

```

{
  "entities": [
    {
      "table": "Person",
      "uid": "8a2eee88-0615-4595-a903-8388ca74d877",
      "values": {
        "FirstName": "Jeremy",
        "LastName": "Coleman",
        "IsExternal": false,

```

```

        "ExitDate": "2032-12-31"
    },
    {
        "table": "Person",
        "uid": "df563a8c-203a-4132-8fcb-bc4adaee35b6",
        "values": {
            "FirstName": "Rosalie",
            "LastName": "Meyer",
            "IsExternal": True,
            "ExitDate": "2032-12-31",
        }
    }
]
"insertMissing": True
}

```

Response Code:

204

Delete a single entity

To delete a single entity, use the URL `<baseUrl>/api/entity/{table}/{uid}`.

Table 34: Delete single entity request

HTTP method	URI	Body
Delete	<code><BaseUrl>/api/entity/{table}/{uid}</code>	None

Table 35: Delete single entity parameters

Parameter	Description	Parameter type	Data type
table	Table name (required).	path	string
uid	GUID of the entity (required).	path	string

Response schema:

```
{}
```

Example:

```
https://<Hostname>/AppServer/api/entity/Person/83b10e84-c64e-4f9f-9ecb-2d0d7c94e8ec
```

Response Code:

```
200
```

Response:

```
{}
```

Delete a batch of single entities

To delete a single entity, use the URL `<baseURL>/api/entities`.

Table 36: Delete a batch of single entities request

HTTP method	URI	Body
Delete	<code><BaseURL>/api/entities</code>	None

Table 37: Delete a batch of single entities parameters

Parameter	Description	Parameter type	Data type
entities	List of entities to delete.	body	DeleteEntity

Body schema:

```
BulkChangeBodyDelete {  
    entities(array[DeleteEntity])  
}  
DeleteEntity {  
    table(string),  
    uid(string)  
}
```

Response code:

204

Example:

`https://<Hostname>/AppServer/api/entities`

Body:

```
{
  "entities": [
    {
      "table": "Person",
      "uid": "8a2eee88-0615-4595-a903-8388ca74d877"
    },
    {
      "table": "Person",
      "uid": "df563a8c-203a-4132-8fcb-bc4adaee35b6"
    }
  ]
}
```

Response Code:

204

Call a method on an entity

This URL can be used to call a method on an entity. The entity will be saved automatically if required.

NOTE: This request type is able to run customizer or dialog methods for an entity. Note that currently only those methods are supported that do not return a value. The method name matching is processed for customizer methods first. If none are found, the system continues the matching using the dialog methods.

To call a method on an entity, use the URL `<baseURL>/api/entity/{table}/{uid}/method/{methodName}`.

Table 38: Call method request

HTTP method	URI	Body
Put	<BaseURL>/api/entity/{table}/{uid}/method/{methodName}	{ "parameters": ["Parameter value"] }

Table 39: Call method parameters

Parameter	Description	Parameter type	Data type
table	Table name (required).	path	string
uid	GUID of the entity (required).	path	string
methodName	Name of the method (required).	path	string
parameters	Parameter values.	body	object[]

Body schema:

```
parameters {
  parameters (object, optional)
}
```

Example 1:

Run a customizer method on an entity.

[https://<Hostname>/AppServer/api/entity/Person/7f6bcca9-05dc-4857-9dc5-
eff915590752/method/ExecuteTemplates](https://<Hostname>/AppServer/api/entity/Person/7f6bcca9-05dc-4857-9dc5-
eff915590752/method/ExecuteTemplates)

Response Code:

204

Example 2:

Run a customizer method on an entity that takes some input parameters.

https://<Hostname>/AppServer/api/entity/Person/9ec76ebd-2024-4a10-a079-948414c8b2c0/method/DelegateElement

Body:

```
{
  "parameters": [
    "f79c30fd-87bb-4958-a812-0683ddcac7c9",
    "<Key><T>HelperHeadPerson</T><P>d4943ffc-d453-4611-8365-4ba394558b15</P><P>9ec76ebd-2024-4a10-a079-948414c8b2c0</P></Key>",
    "2016-05-31",
    "2016-08-01",
    true,
    false,
    "",
    "Is on sick leave."
  ]
}
```

Response Code:

204

Generate an event for an entity

This URL can be used to generate an event for an entity. Additional generation parameters can be provided.

NOTE: The authenticated user must be entitled to use the program function **Allow to trigger any events from the frontend** in order to call an event on an entity.

To generate an event for an entity, use the URL <baseURL>/api/entity/{table}/{uid}/event/{eventName}.

Table 40: Generate event request

HTTP method	URI	Body
Put	<BaseURL>/api/entity/{table}/ {uid}/event/{eventName}	{ "parameters": { "StringValue": "string", "IntValue": 0, "DateValue": "2016-05- 19T11:21:33.579Z", "BoolValue": true }

Table 41: Generate event parameters

Parameter	Description	Parameter type	Data type
table	Table name (required).	path	String
uid	GUID of the entity (required).	path	String
eventName	Name of the event (required).	path	string
parameters	Parameter values.	body	object

Body schema:

```

GenerationParameters {
    parameters (SampleGenerationParameters, optional)
}

SampleGenerationParameters {
    StringValue(string, optional),
    IntValue(integer, optional),
    DateValue(date - time, optional),
    BoolValue (boolean, optional)
}

```

Example:

Generate an event for an entity and specify some additional generation parameters.

```
https://<Hostname>/AppServer/api/entity/Person/6ecb123a-0c8c-4eec-bded-2c4909b886f5/event/DelegateAsync
```

Body:

```
{
  "parameters": {
    "ObjectkeysToDelegate": "<Key><T>HelperHeadOrg</T><P>1f020407-
f677-4ef8-ae83-54fe3b11d71c</P><P>6ecb123a-0c8c-4eec-bded-
2c4909b886f5</P></Key>",
    "UID_PersonReceiver": "d4943ffc-d453-4611-8365-4ba394558b15",
    "UID_PersonSender": "6ecb123a-0c8c-4eec-bded-2c4909b886f5",
    "ValidFrom": "2016-05-31",
    "ValidUntil": "2016-08-01",
    "KeepMeInformed": false,
    "IsDelegable": true,
    "OrderReason": "Is on sick leave.",
    "UID_ITShopOrg": "QER-ITSHOPORG-DELEGATION-PR",
    "UID_PersonInserted": "6ecb123a-0c8c-4eec-bded-2c4909b886f5"
  }
}
```

Response Code:

204

Assignments

The following APIs are used to handle assignments in member tables for the entities of One Identity Manager.

Detailed information about this topic

- [Get assignments for a specific entity](#) on page 55
- [Add assignments](#) on page 58
- [Remove assignments](#) on page 60

Get assignments for a specific entity

To get a list of assignments for a specific entity, use the URL `<baseUrl>/api/assignments/{table}/{column}/{uid}`.

Query parameters to be used in WHERE clauses and predefined WHERE clauses can be provided in the form `@paramname=value`. Per default they have the type `string`. Additionally, it is possible to pass the type with the name: `@otherParam[int]=3`. Types originate from `DbVal` class.

Table 42: Get assignments request

HTTP method	URI	Body
Get	<code><BaseUrl>/api/assignments/{table}/{column}/{uid}</code>	None

Table 43: Get assignments parameters

Parameter	Description	Parameter type	Data type
table	Member table name (required).	path	string
column	Column pointing to the base entity (required).	path	string

Parameter	Description	Parameter type	Data type
uid	GUID of the base entity (required).	path	string
where	WHERE clause.	query	string
whereKey	Predefined WHERE clause from QBMLimitedSQL (Key [UID] or ID [Ident_QBMLimitedSQL]).	query	string
orderBy	ORDER BY clause.	query	string
offset	Offset of first item.	query	integer
limit	Maximum number of results.	query	integer
displayColumns	Additional display columns, semicolon separated.	query	string
loadType	Collection load type. Specify one of the values: <ul style="list-style-type: none"> • Default: Loads read-only entities according to the supplied query. Loaded columns include the primary key, display columns according to the display pattern, some special columns, and the columns defined in the select clause of the query. The entries are sorted by the defined display or the optional orderBy clause of the query. This load type is the default and be omitted. • Slim: Works mostly like Default but does not load display columns and does not build an orderBy clause per default. This type is useful when loading data not intended for display and can save much time by using database indexes. • BulkReadOnly: Loads read-only entities with all columns filled. The columns defined in the query are overridden. • ForeignDisplays: Loads display values for foreign keys contained in the display pattern. This allows showing displays instead of UIDs for foreign keys. 	query	string

Parameter	Description	Parameter type	Data type
	<ul style="list-style-type: none"> ForeignDisplaysForAllColumns: Like ForeignDisplays, but loads displays for all foreign keys contained in the select clauses of the query, not only columns referenced in the display pattern. 		
noUrls	Allow to omit the URI property from the response to reduce the response size especially if used while retrieving large M:N tables like PersonHasObject.	query	boolean

Response schema:

```
array {
  href(string),
  title(string),
  uid(string)
}
```

Example:

`https://<Hostname>/AppServer/api/assignments/PersonInOrg/UID_Org/007b7087-6881-44e7-8954-82374340718f?limit=2`

Response:

```
[
  {
    "href": "https://<Hostname>/AppServer/api/entity/Person/a9c6bc62-3f77-453f-b774-3afd9d4d19e0",
    "title": "Abbey, Jenna (JENNA) - KAGU Org",
    "uid": "a9c6bc62-3f77-453f-b774-3afd9d4d19e0"
  },
  {
    "href": "https://<Hostname>/AppServer/api/entity/Person/be514f69-fc8f-49c0-a791-2b79b8f5cbdf",
```

```

        "title": "Abbott, James (JAMESA) - KAGU Org",
        "uid": "be514f69-fc8f-49c0-a791-2b79b8f5cbdf"
    }
]

```

Add assignments

To add a list of assignments for a specific entity, use the URL `<baseUrl>/api/assignments/{table}/{column}/{uid}`.

Table 44: Add assignments request

HTTP method	URI	Body
Post	<code><BaseURL>/api/assignments/{table}/{column}/{uid}</code>	<code>{"members": ["string"]}</code>

Table 45: Add assignments parameters

Parameter	Description	Parameter type	Data type
table	Member table name (required).	path	string
column	Column pointing to the base entity (required).	path	string
uid	GUID of the base entity (required).	path	string
members	GUIDs of the members to add (required).	body	string[]
ignoreExisting	Ignore existing entries, do not try to add them again.	query	boolean

Response schema:

```

AssignmentResult {
    assigned (int, optional),
    removed (int, optional),
    alreadyAssigned (Array[string], optional)
}

```

Example 1:

https://<Hostname>/AppServer/api/assignments/PersonInOrg/UID_Org/007b7087-6881-44e7-8954-82374340718f

Body:

```
{ "members": [  
    "31d99791-d658-40d7-b5e5-58eecf998797",  
    "40e43904-4958-4bce-915b-f77bab675f06",  
    "7c21b251-d774-4616-bc3a-b91506ddb23b"]  
}
```

Response:

```
{  
    "assigned": 3,  
    "removed": 0  
}
```

Example 2:

https://<Hostname>/AppServer/api/assignments/PersonInOrg/UID_Org/007b7087-6881-44e78954-82374340718f?ignoreExisting=true

Body:

```
{ "members": [  
    "31d99791-d658-40d7-b5e5-58eecf998797",  
    "40e43904-4958-4bce-915b-f77bab675f06",  
    "7c21b251-d774-4616-bc3a-b91506ddb23b"]  
}
```

Response:

```
{  
    "assigned": 0,  
}
```

```

    "removed": 0,
    "alreadyAssigned": [
      "31d99791-d658-40d7-b5e5-58eecf998797",
      "40e43904-4958-4bce-915b-f77bab675f06",
      "7c21b251-d774-4616-bc3a-b91506ddb23b" ]
  }

```

Remove assignments

To remove a list of assignments for a specific entity, use the URL `<baseUrl>/api/assignments/{table}/{column}/{uid}`.

Table 46: Remove assignments request

HTTP method	URI	Body
Delete	<code><BaseURL>/api/assignments/{table}/{column}/{uid}</code>	<code>{"members": ["string"]}</code>

Table 47: Remove assignments parameters

Parameter	Description	Parameter type	Data type
table	Member table name (required).	path	string
column	Column pointing to the base entity (required).	path	string
uid	GUID of the base entity (required).	path	string
members	GUIDs of the members to remove (required).	body	string[]

Response schema:

```

AssignmentResult {
  assigned (int, optional),
  removed (int, optional)
}

```

Example:

https://<Hostname>/AppServer/api/assignments/PersonInOrg/UID_Org/007b7087-6881-44e7-8954-82374340718f

Body:

```
{"members": ["31d99791-d658-40d7-b5e5-58eecf998797"]}
```

Response:

```
{  
  "assigned": 0,  
  "removed": 1  
}
```

Scripts

The One Identity Manager REST API allows you to run any script that is stored inside of the One Identity Manager database.

NOTE: The authenticated user must be entitled to use the **Allow the starting of arbitrary scripts from the frontend** (`Common_StartScripts`) program function in order to run a script.

Detailed information about this topic

- [Run script request](#) on page 62

Run script request

To run a script, use the URL `<baseUrl>/api/script/{name}`.

Table 48: Run script request

HTTP method	URI	Body
Put	<code><BaseUrl>/api/script/{name}</code>	<pre>{ "parameters": ["Parameter value"], "base": "XObjectKey", "value": "Sample value", "returnRawResult": true }</pre>

Table 49: Run script parameters

Parameter	Description	Parameter type	Data type
name	Script name (required).	path	string
parameters	Script parameters.	body	object[]
base	Object key of base object.	body	string
value	Content of the value variable in the script.	body	object
returnRawResult	Allow to return the raw object or string as result.	body	boolean

Body schema:

```
ScriptParameters {
    parameters (object, optional),
    base (string, optional): Object key of base object,
    value (object, optional): Content of the Value variable in the script
}
```

Response schema:

```
ScriptResult {
    result (object, optional): Return value of the script,
    value (object, optional): Content of the Value variable in the script
}
```

Example 1:

https://<Hostname>/AppServer/api/script/QER_GetWebBaseURL

Body:

```
{}
```

Response:

```
{
    "result": "https://<Hostname>/IdentityManager/"
}
```

```
}
```

Example 2:

https://<Hostname>/AppServer/api/script/VI_AE_BuildCentralAccount

Body:

```
{
  "parameters": [
    "f79c30fd-87bb-4958-a812-0683ddcac7c9",
    "Adams",
    "David"
  ]
}
```

Response:

```
{
  "result": "DAVIDA"
}
```

Example 3:

https://<Hostname>/AppServer/api/script/VI_AE_BuildCentralAccount

Body:

```
{
  "parameters": [
    "f79c30fd-87bb-4958-a812-0683ddcac7c9",
    "Adams",
    "David"
  ]
}
```

```
    ],  
    "returnRawResult": true  
  }  
}
```

Response:

```
{  
  "result": "DAVIDA"  
}
```

PowerShell sample

```
# Construct auth json
$authdata = @{AuthString="Module=DialogUser;User=<user name>;Password="}
$authJson = ConvertTo-Json $authdata -Depth 2

# Login (important, pass the NAME for your session variable in -SessionVariable)
Invoke-RestMethod -Uri "https://<Hostname>/AppServer/auth/apphost" -Body
$authJson.ToString() -Method Post -UseDefaultCredentials -Headers @
{Accept="application/json"} -SessionVariable wsession -ContentType
"application/json"

# Do stuff (always pass -WebSession and use the variable you NAMED in the previous step)

# Sample 1: Load collection using Post method
$body = @{"where"="LastName like 'B%';orderBy="LastName ASC, FirstName DESC"}
| ConvertTo-Json
Invoke-RestMethod -Uri
"https://<Hostname>/AppServer/api/entities/Person?loadType=ForeignDisplays" -
WebSession $wsession -Method Post -Body $body -ContentType application/json

# Sample 2: Create a new object and return URI of new object
$body = @{"values=@
{FirstName="Jeremia";LastName="Bodewell";IsExternal=1;BirthDate="1993-05-
14";Gender=1}} | ConvertTo-Json
$newURI = (Invoke-RestMethod -Uri
"https://<Hostname>/AppServer/api/entity/Person" -WebSession $wsession -
Method Post -Body $body -ContentType application/json).uri

# Sample 3: Get all properties for new object
```

```
(Invoke-RestMethod -Uri $newURI -WebSession $wsession -Method Get -
ContentType application/json).Values
```

```
# Sample 4: Update the new object
```

```
$body=@{values=@{LastName="Garibaldi";IsExternal=0;ExitDate="2021-12-
16T14:24:32.424Z";PersonalTitle="Administration (EMEA)"}} | ConvertTo-Json
```

```
Invoke-RestMethod -Uri $NewURI -WebSession $wsession -Method Put -Body $body
-ContentType application/json
```

```
# Sample 5: Delete the new object
```

```
Invoke-RestMethod -Uri $NewURI -WebSession $wsession -Method Delete -
ContentType application/json
```

```
# Logout
```

```
Invoke-RestMethod -Uri "https://<Hostname>/AppServer/auth/logout" -WebSession
$wsession -Method Post -ContentType "application/json"
```

One Identity solutions eliminate the complexities and time-consuming processes often required to govern identities, manage privileged accounts and control access. Our solutions enhance business agility while addressing your IAM challenges with on-premises, cloud and hybrid environments.

Contacting us

For sales and other inquiries, such as licensing, support, and renewals, visit <https://www.oneidentity.com/company/contact-us.aspx>.

Technical support resources

Technical support is available to One Identity customers with a valid maintenance contract and customers who have trial versions. You can access the Support Portal at <https://support.oneidentity.com/>.

The Support Portal provides self-help tools you can use to solve problems quickly and independently, 24 hours a day, 365 days a year. The Support Portal enables you to:

- Submit and manage a Service Request
- View Knowledge Base articles
- Sign up for product notifications
- Download software and technical documentation
- View how-to videos at www.YouTube.com/OneIdentity
- Engage in community discussions
- Chat with support engineers online
- View services to assist you with your product