Copyright 2019 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 Web site (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. All other trademarks are the property of their respective owners.

Legend

⚠️ WARNING: A WARNING icon indicates a potential for property damage, personal injury, or death.

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

ℹ️ IMPORTANT, NOTE, TIP, MOBILE, or VIDEO: An information icon indicates supporting information.

One Identity Manager REST API Reference Guide
Updated - August 2019
Version - 8.1.1
Contents

Guidelines and conventions .............................................................................................................. 5
BaseUrl .............................................................................................................................................. 5
Parameter formats ............................................................................................................................... 5
Path parameters ................................................................................................................................. 5
Query parameters ............................................................................................................................... 6
HTTP request methods ...................................................................................................................... 6
Response formats .............................................................................................................................. 7
HTTP response codes ....................................................................................................................... 7
Authentication ..................................................................................................................................... 8
Initial data for authentication modules ............................................................................................ 8
Identifying activated authentication modules .................................................................................. 9
Example ............................................................................................................................................ 9
Getting details of an authentication module ................................................................................... 11
Example ........................................................................................................................................... 11
Login ............................................................................................................................................... 12
Example ........................................................................................................................................... 13
Logout ............................................................................................................................................. 13
Example ........................................................................................................................................... 13
Date formats ..................................................................................................................................... 14
Object identifiers .............................................................................................................................. 14
Using offset and limit parameters to cycle through records .......................................................... 14
Session object global variables ....................................................................................................... 15
Set global variable ............................................................................................................................ 15
Example ............................................................................................................................................ 16
Clear global variable ......................................................................................................................... 16
Example ............................................................................................................................................ 17

Collections ........................................................................................................................................ 18
Get collection using GET method ................................................................................................... 18
Example 1 .......................................................................................................................................... 20
Example 2 .......................................................................................................................................... 20
Get collection using POST method ............................................................................................... 21
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single entities</td>
<td>25</td>
</tr>
<tr>
<td>Create a single entity</td>
<td>25</td>
</tr>
<tr>
<td>Example</td>
<td>26</td>
</tr>
<tr>
<td>Get a single entity</td>
<td>27</td>
</tr>
<tr>
<td>Example</td>
<td>27</td>
</tr>
<tr>
<td>Change a single entity</td>
<td>31</td>
</tr>
<tr>
<td>Example</td>
<td>32</td>
</tr>
<tr>
<td>Delete a single entity</td>
<td>32</td>
</tr>
<tr>
<td>Example</td>
<td>33</td>
</tr>
<tr>
<td>Call a method on an entity</td>
<td>33</td>
</tr>
<tr>
<td>Example 1</td>
<td>34</td>
</tr>
<tr>
<td>Example 2</td>
<td>34</td>
</tr>
<tr>
<td>Generate an event for an entity</td>
<td>35</td>
</tr>
<tr>
<td>Example</td>
<td>36</td>
</tr>
<tr>
<td>Assignments</td>
<td>38</td>
</tr>
<tr>
<td>Get assignments for a specific entity</td>
<td>38</td>
</tr>
<tr>
<td>Example</td>
<td>39</td>
</tr>
<tr>
<td>Add assignments</td>
<td>40</td>
</tr>
<tr>
<td>Example</td>
<td>41</td>
</tr>
<tr>
<td>Remove assignments</td>
<td>41</td>
</tr>
<tr>
<td>Example</td>
<td>42</td>
</tr>
<tr>
<td>Scripts</td>
<td>43</td>
</tr>
<tr>
<td>Run script request</td>
<td>43</td>
</tr>
<tr>
<td>Example 1</td>
<td>44</td>
</tr>
<tr>
<td>Example 2</td>
<td>44</td>
</tr>
<tr>
<td>Appendix: Windows PowerShell sample</td>
<td>46</td>
</tr>
<tr>
<td>About us</td>
<td>48</td>
</tr>
<tr>
<td>Contacting us</td>
<td>48</td>
</tr>
<tr>
<td>Technical support resources</td>
<td>48</td>
</tr>
</tbody>
</table>
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 a person, you specify their primary key (GUID) with a path parameter. The following shows the URI format for this request, and an example:

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

Example:

https://<Hostname>/AppServer/api/entity/Person/25d87790-105d-4afbd04-cfdd7cc9f5b5
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 people 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, and an example:

```
<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 people 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.

**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.

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>
<thead>
<tr>
<th>Table 1: HTTP response codes</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Response status codes</strong></td>
</tr>
<tr>
<td>200</td>
</tr>
<tr>
<td>204</td>
</tr>
<tr>
<td>401</td>
</tr>
<tr>
<td>404</td>
</tr>
<tr>
<td>405</td>
</tr>
<tr>
<td>500</td>
</tr>
</tbody>
</table>

```json
{
    "responseStatus": {
        "message": "Sample text"},
    "errorString": "Sample text",
    "exceptions": [{
        "number": 810017,
        "message": "Sample text"}
    ]
}
```
<table>
<thead>
<tr>
<th>Response status codes</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>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.</td>
</tr>
</tbody>
</table>

**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.

**Initial data for authentication modules**

The authentication string is formatted as follows:

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

**Example:**

```
Module=DialogUser;User=<username>;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 detailed information about authentication modules, see the *One Identity Manager Configuration 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

<table>
<thead>
<tr>
<th>HTTP method</th>
<th>URI</th>
<th>Body</th>
</tr>
</thead>
<tbody>
<tr>
<td>Get</td>
<td><code>&lt;BaseURL&gt;/appserver/authmodules</code></td>
<td>None</td>
</tr>
</tbody>
</table>

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": ...
}
```
"id": "DialogUser",
"caption": "System user",
"passwordBased": false,
"isDefault": true
},
{
"id": "RoleBasedPerson",
"caption": "Employee (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": "Employee (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

<table>
<thead>
<tr>
<th>HTTP method</th>
<th>URI</th>
<th>Body</th>
</tr>
</thead>
<tbody>
<tr>
<td>Get</td>
<td>&lt;BaseURL&gt;/appserver/authmodules/{id}</td>
<td>None</td>
</tr>
</tbody>
</table>

Table 4: Get authentication module parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Parameter type</th>
<th>Data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Authentication module (required)</td>
<td>path</td>
<td>string</td>
</tr>
</tbody>
</table>

Response schema:

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

Example

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

Response:

```json
[{  
    "id": "DialogUser",  
    "caption": "System user",  
    "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=$ecret.

**Login**

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

**Table 5: Authenticate request**

<table>
<thead>
<tr>
<th>HTTP method</th>
<th>URI</th>
<th>Body</th>
</tr>
</thead>
<tbody>
<tr>
<td>Post</td>
<td>&lt;BaseUrl&gt;/auth/apphost</td>
<td>Property authString containing an authentication string. For more information, see Initial data for authentication modules on page 8.</td>
</tr>
</tbody>
</table>

**Body schema:**

{"authString": String}

**Response schema:**

{  
  "claims": {
    "http://schemas.oneidentity.com/ws/2017/04/identity/claims/identifier": String,
    "http://schemas.oneidentity.com/ws/2017/04/identity/claims/useruid": String,
  },
  "passwordBased": Boolean,
  "moduleDisplay": String,
  "sessionId": String,
  "userName": String,
  "responseStatus": {}  
}
Example

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

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

Response:
{
  "claims": {
    "http://schemas.oneidentity.com/ws/2017/04/identity/claims/identifier": "<user name>",
  },
  "passwordBased": true,
  "moduleDisplay": "System user",
  "sessionID": "nV8R3iw4KfmEiZYdA3uy",
  "userName": "<user name>",
  "responseStatus": {}
}

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

<table>
<thead>
<tr>
<th>HTTP method</th>
<th>URI</th>
<th>Body</th>
</tr>
</thead>
<tbody>
<tr>
<td>Post</td>
<td>&lt;BaseURL&gt;/auth/logout</td>
<td></td>
</tr>
</tbody>
</table>

Response schema:
{"responseStatus": {}}
Example

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

Response:
{"responseStatus": {}}

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 persons can match hundreds or thousands of persons.

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 persons:
  
  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 persons, starting with the 101st person:

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>
<thead>
<tr>
<th>Query parameter</th>
<th>Description</th>
<th>Default value</th>
</tr>
</thead>
<tbody>
<tr>
<td>limit</td>
<td>Maximum number of records to show in the response.</td>
<td>0</td>
</tr>
<tr>
<td>offset</td>
<td>Zero-based index of first record to show in the response.</td>
<td>0</td>
</tr>
</tbody>
</table>

### 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.

### Set global variable

To set a global variable, use the URL `<baseURL>/appserver/variable/{name}`
Table 8: Set global variable request

<table>
<thead>
<tr>
<th>HTTP method</th>
<th>URI</th>
<th>Body</th>
</tr>
</thead>
<tbody>
<tr>
<td>Put</td>
<td><code>&lt;baseURL&gt;/appserver/variable/{name}</code></td>
<td><code>{&quot;value&quot;: &lt;value&gt;}</code></td>
</tr>
</tbody>
</table>

Table 9: Set global variable parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Parameter type</th>
<th>Data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Variable name (required)</td>
<td>path</td>
<td>string</td>
</tr>
</tbody>
</table>

Body schema:

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

Example

https://<Hostname>/AppServer/appserver/variable/FullSync

Body:

{"value": true}

Clear global variable

To clear a global variable, use the URL `<baseURL>/appserver/variable/{name}`

Table 10: Clear global variable request

<table>
<thead>
<tr>
<th>HTTP method</th>
<th>URI</th>
<th>Body</th>
</tr>
</thead>
<tbody>
<tr>
<td>Put</td>
<td><code>&lt;baseURL&gt;/appserver/variable/{name}</code></td>
<td><code>{&quot;clear&quot;: true}</code></td>
</tr>
</tbody>
</table>

Table 11: Clear global variable parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Parameter type</th>
<th>Data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Variable name (required)</td>
<td>path</td>
<td>string</td>
</tr>
</tbody>
</table>

Body schema:

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

https://<Hostname>/AppServer/appserver/variable/FullSync

Body:

{"clear": true}
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.

Get collection using GET method

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

Table 12: Get collection (GET) request

<table>
<thead>
<tr>
<th>HTTP method</th>
<th>URI</th>
<th>Body</th>
</tr>
</thead>
<tbody>
<tr>
<td>Get</td>
<td>&lt;baseURL&gt;/api/entities/{table}</td>
<td>None</td>
</tr>
</tbody>
</table>

Table 13: Get collection (GET) parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Parameter type</th>
<th>Data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>table</td>
<td>Table name (required).</td>
<td>path</td>
<td>string</td>
</tr>
<tr>
<td>where</td>
<td>WHERE clause.</td>
<td>query</td>
<td>string</td>
</tr>
<tr>
<td>orderBy</td>
<td>ORDER BY clause.</td>
<td>query</td>
<td>string</td>
</tr>
<tr>
<td>offset</td>
<td>Offset of first item.</td>
<td>query</td>
<td>integer</td>
</tr>
<tr>
<td>limit</td>
<td>Maximum number of results.</td>
<td>query</td>
<td>integer</td>
</tr>
<tr>
<td>displayColumns</td>
<td>Additional display columns, semicolon separated.</td>
<td>query</td>
<td>string</td>
</tr>
<tr>
<td>loadType</td>
<td>Collection load type. Specify one of the values listed in the Values of parameter loadType table.</td>
<td>query</td>
<td>string</td>
</tr>
</tbody>
</table>
Table 14: Values of parameter loadType

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default</td>
<td>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.</td>
</tr>
<tr>
<td>Slim</td>
<td>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.</td>
</tr>
<tr>
<td>BulkReadOnly</td>
<td>Loads read-only entities with all columns filled. The columns defined in the query are overridden.</td>
</tr>
<tr>
<td>ForeignDisplays</td>
<td>Loads display values for foreign keys contained in the display pattern. This allows showing displays instead of UIDs for foreign keys.</td>
</tr>
<tr>
<td>ForeignDisplaysForAllColumns</td>
<td>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.</td>
</tr>
</tbody>
</table>

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:

```json
[{
  "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"
  }
}]

Get collection using POST method

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

Table 15: Get collection (GET) request

<table>
<thead>
<tr>
<th>HTTP method</th>
<th>URI</th>
<th>Body</th>
</tr>
</thead>
<tbody>
<tr>
<td>Get</td>
<td>&lt;BaseURL&gt;/api/entities/{table}</td>
<td>&quot;where&quot;: &quot;&quot;, &quot;orderBy&quot;: &quot;&quot;</td>
</tr>
</tbody>
</table>

Table 16: Get collection (GET) parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Parameter type</th>
<th>Data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>table</td>
<td>Table name (required).</td>
<td>path</td>
<td>string</td>
</tr>
<tr>
<td>where</td>
<td>WHERE clause.</td>
<td>body</td>
<td>string</td>
</tr>
<tr>
<td>orderBy</td>
<td>ORDER BY clause.</td>
<td>body</td>
<td>string</td>
</tr>
<tr>
<td>offset</td>
<td>Offset of first item.</td>
<td>query</td>
<td>integer</td>
</tr>
<tr>
<td>limit</td>
<td>Maximum number of results.</td>
<td>query</td>
<td>integer</td>
</tr>
<tr>
<td>displayColumns</td>
<td>Additional display columns, semicolon separated.</td>
<td>query</td>
<td>string</td>
</tr>
<tr>
<td>loadType</td>
<td>Collection load type. Specify one of the values listed in table Values of parameter loadType on page 22.</td>
<td>query</td>
<td>string</td>
</tr>
</tbody>
</table>
Table 17: Values of parameter `loadType`

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default</td>
<td>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.</td>
</tr>
<tr>
<td>Slim</td>
<td>Works mostly like <strong>Default</strong> 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.</td>
</tr>
<tr>
<td>BulkReadOnly</td>
<td>Loads read-only entities with all columns filled. The columns defined in the query are overridden.</td>
</tr>
<tr>
<td>ForeignDisplays</td>
<td>Loads display values for foreign keys contained in the display pattern. This allows showing displays instead of UIDs for foreign keys.</td>
</tr>
<tr>
<td>ForeignDisplaysForAllColumns</td>
<td>Like <strong>ForeignDisplays</strong>, but loads displays for all foreign keys contained in the select clauses of the query, not only columns referenced in the display pattern.</td>
</tr>
</tbody>
</table>

Response schema:

```json
CollectionEntry {
    uri(string),
    display(string, optional),
    longDisplay(string, optional),
    values(SampleValues, optional)
}
```

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

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
}]}
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.

Create a single entity

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

**Table 18: Create single entity request**

<table>
<thead>
<tr>
<th>HTTP method</th>
<th>URI</th>
<th>Body</th>
</tr>
</thead>
<tbody>
<tr>
<td>Post</td>
<td><code>&lt;baseURL&gt;/api/entity/{table}</code></td>
<td><code>{&quot;values&quot;: {  &quot;StringColumn&quot;: &quot;string&quot;,  &quot;IntColumn&quot;: 0,  &quot;DateColumn&quot;: &quot;2016-05-19T11:21:33.579Z&quot;,  &quot;BoolColumn&quot;: True}}</code></td>
</tr>
</tbody>
</table>

**Table 19: Create single entity parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Parameter type</th>
<th>Data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>table</td>
<td>Table name (required)</td>
<td>path</td>
<td>string</td>
</tr>
<tr>
<td>values</td>
<td>Values to set</td>
<td>body</td>
<td>SampleValues</td>
</tr>
</tbody>
</table>

**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"
}
Get a single entity

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

Table 20: Get single entity request

<table>
<thead>
<tr>
<th>HTTP method</th>
<th>URI</th>
<th>Body</th>
</tr>
</thead>
<tbody>
<tr>
<td>Get</td>
<td>&lt;baseURL&gt;/api/entity/{table}/{uid}</td>
<td>None</td>
</tr>
</tbody>
</table>

Table 21: Get single entity parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Parameter type</th>
<th>Data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>table</td>
<td>Table name (required)</td>
<td>path</td>
<td>string</td>
</tr>
<tr>
<td>uid</td>
<td>Guid of this entity (required)</td>
<td>path</td>
<td>string</td>
</tr>
</tbody>
</table>

Response schema:

**SingleEntry**

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

**SampleValues**

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

**Link**

```javascript
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,
        "AuthentifierLogins": "",
        "BirthDate": "1993-05-14T00:00:00.0000000Z",
        "Building": "",
        "CanonicalName": "",
        "CentralAccount": "JEREMIAB",
        "CentralPassword": "",
        "CentralSAPAccount": "BODEWELJ",
        "City": "",
        "CompanyMember": "",
        "CustomProperty01": "",
        "CustomProperty02": "",
        "CustomProperty03": "",
        "CustomProperty04": "",
        "CustomProperty05": "",
        "CustomProperty06": "",
        "CustomProperty07": "",
        "CustomProperty08": "",
        "CustomProperty09": "",
        "CustomProperty10": "",
        "DateLastWorked": "1899-12-30T00:00:00.0000000Z",
        "DeactivationEnd": "1899-12-30T00:00:00.0000000Z",
        "DeactivationStart": "1899-12-30T00:00:00.0000000Z",
        "DefaultEmailAddress": "",
        "Description": "",
        "DialogUserPassword": "",
        "DialogUserSalt": "",
        "DisplayTelephoneBook": false,
        "DistinguishedName": ""
    }
}
"EntryDate": "2016-05-20T00:00:00.0000000Z",
"ExitDate": "1899-12-30T00:00:00.0000000Z",
"Fax": "",
"FaxExtension": "",
"FirstName": "Jeremia",
"Floor": "",
"FormerName": "",
"Gender": 1,
"GenerationalQualifier": "",
"ImportSource": "",
"Initials": "JB",
"InternalName": "Bodewell, Jeremia",
"IsCar": false,
"IsDummyPerson": false,
"IsDuplicateName": false,
"IsExternal": true,
"IsInActive": false,
"IsNoInherite": false,
"IsNoteBookUser": false,
"IsRemoteAccessAllowed": false,
"IsSecurityIncident": false,
"IsSupporter": false,
"IsTemporaryDeactivated": false,
"IsTerminalServerAllowed": false,
"IsVIP": false,
"IsX500Dummy": false,
"J PegPhoto": "",
"LastName": "Bodewell",
"MiddleName": "",
"NameAddOn": "",
"PasswordAnswer": "",
"PasswordQuery": "",
"PersonalTitle": "",
"PersonnelNumber": "",
"Phone": "",
"PhoneExtension": ""
"XUserUpdated": "<user name>",
"ZIPCode": ""
}
}

## Change a single entity

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

### Table 22: Change single entity request

<table>
<thead>
<tr>
<th>HTTP method</th>
<th>URI</th>
<th>Body</th>
</tr>
</thead>
<tbody>
<tr>
<td>Put</td>
<td><code>&lt;baseURL&gt;/api/entity/{table}/{uid}</code></td>
<td>&quot;values&quot;: {</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&quot;StringColumn&quot;: &quot;string&quot;,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&quot;IntColumn&quot;: 0,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&quot;DateColumn&quot;: &quot;2016-05-19T11:21:33.579Z&quot;,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&quot;BoolColumn&quot;: true</td>
</tr>
<tr>
<td></td>
<td></td>
<td>}</td>
</tr>
</tbody>
</table>

### Table 23: Change single entity parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Parameter type</th>
<th>Data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>table</td>
<td>Table name (required)</td>
<td>path</td>
<td>string</td>
</tr>
<tr>
<td>uid</td>
<td>GUID of the entity (required)</td>
<td>path</td>
<td>string</td>
</tr>
<tr>
<td>values</td>
<td>Values to change</td>
<td>body</td>
<td>SampleValues</td>
</tr>
</tbody>
</table>

**Body schema:**

```plaintext
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:
{}

Delete a single entity

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

Table 24: Delete single entity request

<table>
<thead>
<tr>
<th>HTTP method</th>
<th>URI</th>
<th>Body</th>
</tr>
</thead>
<tbody>
<tr>
<td>Delete</td>
<td>&lt;BaseURL&gt;/api/entity/{table}/{uid}</td>
<td>None</td>
</tr>
</tbody>
</table>
Table 25: Delete single entity parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Parameter type</th>
<th>Data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>table</td>
<td>Table name (required)</td>
<td>path</td>
<td>string</td>
</tr>
<tr>
<td>uid</td>
<td>Guid of the entity (required)</td>
<td>path</td>
<td>string</td>
</tr>
</tbody>
</table>

Response schema:

{ }

Example

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

Response Code:

200

Response:

{ }

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.

1 | 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 26: Call method request

<table>
<thead>
<tr>
<th>HTTP method</th>
<th>URI</th>
<th>Body</th>
</tr>
</thead>
<tbody>
<tr>
<td>Put</td>
<td>&lt;BaseURL&gt;/api/entity/{table}/{uid}/method/{methodName}</td>
<td>{</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&quot;parameters&quot;: [</td>
</tr>
</tbody>
</table>
### Table 27: Call method parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Parameter type</th>
<th>Data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>table</td>
<td>Table name (required)</td>
<td>path</td>
<td>string</td>
</tr>
<tr>
<td>uid</td>
<td>GUID of the entity (required)</td>
<td>path</td>
<td>string</td>
</tr>
<tr>
<td>methodName</td>
<td>Name of the method (required)</td>
<td>path</td>
<td>string</td>
</tr>
<tr>
<td>parameters</td>
<td>Parameter values</td>
<td>body</td>
<td>object[]</td>
</tr>
</tbody>
</table>

**Body schema:**

```json
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

**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:**

```json
{
}
```
"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>
<thead>
<tr>
<th>HTTP method</th>
<th>URI</th>
<th>Body</th>
</tr>
</thead>
</table>
| Put         | <BaseURL>/api/entity/{table}/{uid}/event/{eventName} | {"parameters": {"StringValue": "string",
                            "IntValue": 0,
                            "DateValue": "2016-05-19T11:21:33.579Z",
                            "BoolValue": true
                        }"} |
Table 29: Generate event parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Parameter type</th>
<th>Data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>table</td>
<td>Table name (required)</td>
<td>path</td>
<td>String</td>
</tr>
<tr>
<td>uid</td>
<td>GUID of the entity (required)</td>
<td>path</td>
<td>String</td>
</tr>
<tr>
<td>eventName</td>
<td>Name of the event (required)</td>
<td>path</td>
<td>string</td>
</tr>
<tr>
<td>parameters</td>
<td>Parameter values</td>
<td>body</td>
<td>object</td>
</tr>
</tbody>
</table>

**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-2c490b886f5/event/DelegateAsync

**Body:**

```
{  
  "parameters": {  
    "ObjectkeysToDelegate": ":<Key><T>HelperHeadOrg</T><P>f020407-f677-4ef8-ae83-54fe3b11d71c</P><P>6ecb123a-0c8c-4eec-bded-2c490b886f5</P></Key>",  
    "UID_PersonReceiver": "d4943ffe-d453-4611-8365-4ba394558b15",
    "UID_PersonSender": "6ecb123a-0c8c-4eec-bded-2c490b886f5",
    "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.

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}`

Table 30: Get assignments request

<table>
<thead>
<tr>
<th>HTTP method</th>
<th>URI</th>
<th>Body</th>
</tr>
</thead>
<tbody>
<tr>
<td>Get</td>
<td><code>&lt;BaseURL&gt;/api/assignments/{table}/{column}/{uid}</code></td>
<td>None</td>
</tr>
</tbody>
</table>

Table 31: Get assignments parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Parameter type</th>
<th>Data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>table</td>
<td>Member table name (required)</td>
<td>path</td>
<td>string</td>
</tr>
<tr>
<td>column</td>
<td>Column pointing to the base entity (required)</td>
<td>path</td>
<td>string</td>
</tr>
<tr>
<td>uid</td>
<td>GUID of the base entity (required)</td>
<td>path</td>
<td>string</td>
</tr>
<tr>
<td>where</td>
<td>WHERE clause</td>
<td>query</td>
<td>string</td>
</tr>
<tr>
<td>orderBy</td>
<td>ORDER BY clause</td>
<td>query</td>
<td>string</td>
</tr>
<tr>
<td>offset</td>
<td>Offset of first item</td>
<td>query</td>
<td>integer</td>
</tr>
<tr>
<td>limit</td>
<td>Maximum number of results</td>
<td>query</td>
<td>integer</td>
</tr>
<tr>
<td>loadType</td>
<td>Collection load type. Specify one of the values listed in the Values of parameter loadType table.</td>
<td>query</td>
<td>string</td>
</tr>
</tbody>
</table>
Table 32: Values of parameter loadType

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default</td>
<td>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.</td>
</tr>
<tr>
<td>Slim</td>
<td>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.</td>
</tr>
<tr>
<td>BulkReadOnly</td>
<td>Loads read-only entities with all columns filled. The columns defined in the query are overridden.</td>
</tr>
<tr>
<td>ForeignDisplays</td>
<td>Loads display values for foreign keys contained in the display pattern. This allows showing displays instead of UIDs for foreign keys.</td>
</tr>
<tr>
<td>ForeignDisplaysForAllColumns</td>
<td>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.</td>
</tr>
</tbody>
</table>

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-
Add assignments

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

Table 33: Add assignments request

<table>
<thead>
<tr>
<th>HTTP method</th>
<th>URI</th>
<th>Body</th>
</tr>
</thead>
<tbody>
<tr>
<td>Post</td>
<td>&lt;BaseURL&gt;/api/assignments/{table}/{column}/{uid}</td>
<td>{&quot;members&quot;: [&quot;string&quot;]}</td>
</tr>
</tbody>
</table>

Table 34: Add assignments parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Parameter type</th>
<th>Data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>table</td>
<td>Member table name (required)</td>
<td>path</td>
<td>string</td>
</tr>
<tr>
<td>column</td>
<td>Column pointing to the base entity (required)</td>
<td>path</td>
<td>string</td>
</tr>
<tr>
<td>uid</td>
<td>GUID of the base entity (required)</td>
<td>path</td>
<td>string</td>
</tr>
<tr>
<td>members</td>
<td>GUIDs of the members to add (required)</td>
<td>body</td>
<td>string[]</td>
</tr>
</tbody>
</table>

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",
    "40e43904-4958-4bce-915b-f77bab675f06",
    "7c21b251-d774-4616-bc3a-b91506dd23b"]}

Response:

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

Remove assignments

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

Table 35: Remove assignments request

<table>
<thead>
<tr>
<th>HTTP method</th>
<th>URI</th>
<th>Body</th>
</tr>
</thead>
<tbody>
<tr>
<td>Delete</td>
<td>&lt;BaseURL&gt;/api/assignments/{table}/{column}/{uid}</td>
<td>{&quot;members&quot;: [&quot;string&quot;]}</td>
</tr>
</tbody>
</table>

Table 36: Remove assignments parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Parameter type</th>
<th>Data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>table</td>
<td>Member table name (required)</td>
<td>path</td>
<td>string</td>
</tr>
</tbody>
</table>
## Parameter Description

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Parameter type</th>
<th>Data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>column</td>
<td>Column pointing to the base entity (required)</td>
<td>path</td>
<td>string</td>
</tr>
<tr>
<td>uid</td>
<td>GUID of the base entity (required)</td>
<td>path</td>
<td>string</td>
</tr>
<tr>
<td>members</td>
<td>GUIDs of the members to remove (required)</td>
<td>body</td>
<td>string[]</td>
</tr>
</tbody>
</table>

### Response schema:

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

### Example

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

**Body:**

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

**Response:**

```json
{
    "assigned": 0,
    "removed": 1
}
```
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 program function in order to execute a script.

## Run script request

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

<table>
<thead>
<tr>
<th>HTTP method</th>
<th>URI</th>
<th>Body</th>
</tr>
</thead>
<tbody>
<tr>
<td>Put</td>
<td><code>&lt;baseURL&gt;/api/script/{name}</code></td>
<td>{</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&quot;parameters&quot;: [</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&quot;Parameter value&quot;],</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&quot;base&quot;: &quot;XObjectKey&quot;,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&quot;value&quot;: &quot;Sample value&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>}</td>
</tr>
</tbody>
</table>

### Table 37: Run script request

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Parameter type</th>
<th>Data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Script name (required)</td>
<td>path</td>
<td>string</td>
</tr>
<tr>
<td>parameters</td>
<td></td>
<td>body</td>
<td>object[]</td>
</tr>
<tr>
<td>base</td>
<td>Object key of base object</td>
<td>body</td>
<td>string</td>
</tr>
<tr>
<td>value</td>
<td>Content of the value variable in the script</td>
<td>body</td>
<td>object</td>
</tr>
</tbody>
</table>
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-a812-0683ddcacf9",
    "Adams",
  
...}
"David"
]
}

Response:
{
    "result": "DAVIDA"
}

Appendix: Windows 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)
{Accept="application/json"} -SessionVariable wsession

# 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
About us

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 or other inquiries, visit https://www.oneidentity.com/company/contact-us.aspx or call +1-800-306-9329.

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