One Identity Manager 8.0.1

REST API Reference Guide

One Identity Manager REST API Reference Guide
Updated - March 2018
Version - 8.0.1

Contents

1

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 will be installed with the following path.

`https://<Hostname>/AppServer`

Parameter Formats

The HTTP requests use two types of parameters:

- Path parameters
- Query parameters

Path Parameters

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

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

`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 persons 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}
```

```
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 persons that are marked as external ordered by their company membership, use the following request:

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

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.

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 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	Internal Server Error. The error message is returned in the property errorString of the response.
	`{`
	`"responseStatus": {`
	`"message": "Sample text"},`
	`"errorString": "Sample text",`
	`"exceptions": [{`
	`"number": 810017,`
	`"message": "Sample text"}`
	`]`
	`}`
	Due to security reason, the detailed error message will not be returned to the caller. More information could be retrieved from the application server log on the application server.

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 authentication modules "System user" and "Component Authenticator" and the role-based authentication modules are enabled in the One Identity Manager.
- Authentication modules are defined in the modules and are not available until the 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=<user name>;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

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": "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

HTTP Method	URI	Body
Get	<BaseURL>/appserver/authmodules/{id}	None

Table 4: Get authentication module Parameters

Parameter	Description	Parameter Type	Data Type
id	Authentication module (mandatory)	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 `valueTemplate` 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 frontends only.

For the sample 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

HTTP Method	URI	Body
Post	<BaseURL>/auth/apphost	Property `authString` containing an authentication string. For more information, see Initial Data for authentication modules on page 8.

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,
          "http://schemas.oneidentity.com/ws/2017/04/identity/claims/module": 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>",
            "http://schemas.oneidentity.com/ws/2017/04/identity/claims/useruid": "QBM-
            A60F9E5189134AFFB6711DFCBC3F260E",
            "http://schemas.oneidentity.com/ws/2017/04/identity/claims/module":
            "DialogUser"
        },
        "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

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

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 the One Identity Manager. Every time an object is created, the system internally generates a unique identifier, called a "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 7: Pagination paramters

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 variable 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

HTTP Method	URI	Body
Put	<baseURL>/appserver/variable/{name}	{"value": <value>}

Table 9: Set global variable parameters

Parameter	Description	Parameter Type	Data Type
name	Variable Name (mandatory)	path	string

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

HTTP Method	URI	Body
Put	<baseURL>/appserver/variable/{name}	{"clear": true}

Table 11: Clear global variable parameters

Parameter	Description	Parameter Type	Data Type
name	Variable Name (mandatory)	path	string

Body schema:

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

Example

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

Body:

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

2

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

HTTP Method	URI	Body
Get	<BaseURL>/api/entities/{table}	None

Table 13: Get collection (GET) parameters

Parameter	Description	Parameter Type	Data Type
table	Table name (mandatory).	path	string
where	WHERE clause.	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 listed in table Values of paramter loadType on page 18.	query	string

Table 14: Values of paramter `loadType`

Value	Description
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 will be 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 indices.
BulkReadOnly	Loads read-only entities with all columns filled. The columns defined in the query will be overridden.
ForeignDisplays	Loads display values for foreign keys contained in the display pattern too. 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.

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"
     }
}]
```

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

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

Table 16: Get collection (GET) parameters

Parameter	Description	Parameter Type	Data Type
table	Table name (mandatory).	path	string
where	WHERE clause.	body	string
orderBy	ORDER BY clause.	body	string
offset	Offset of first item.	query	integer
limit	Maximum number of results.	query	integer
displayColumns	Additional display columns, semicolon separated.	query	string

Parameter	Description	Parameter Type	Data Type
loadType	Collection load type. Specify one of the values listed in table Values of paramter loadType on page 21.	query	string

Table 17: Values of paramter `loadType`

Value	Description
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 will be 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 indices.
BulkReadOnly	Loads read-only entities with all columns filled. The columns defined in the query will be overridden.
ForeignDisplays	Loads display values for foreign keys contained in the display pattern too. 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.

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

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
        }
}]
```

3

Single Entities

The following APIs are used to handle entities of the 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

HTTP Method	URI	Body

Post	<BaseURL>/api/entity/{table}	{"values": {
		"StringColumn": "string",
		"IntColumn": 0,
		"DateColumn": "2016-05-19T11:21:33.579Z",
		"BoolColumn": True
		}

Table 19: Create single entity parameters

Parameter	Description	Parameter Type	Data Type
table	Table name (mandatory)	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"
}
```

Get a single entity

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

Table 20: Get single entity request

HTTP Method	URI	Body
Get	<BaseURL>/api/entity/{table}/{uid}	None

Table 21: Get single entity parameters

Parameter	Description	Parameter Type	Data Type
table	Table name (mandatory)	path	string
uid	Guid of this entity (mandatory)	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,
        "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,
"JPegPhoto": "",
"LastName": "Bodewell",
"MiddleName": "",
"NameAddOn": "",
"PasswordAnswer": "",
"PasswordQuery": "",
"PersonalTitle": "",
"PersonnelNumber": "",
"Phone": "",
"PhoneExtension": "",
```

```
"PhoneMobile": "",
"PostalOfficeBox": "",
"PreferredName": "",
"Remarks": "",
"RiskIndexCalculated": 0,
"Room": "",
"Salutation": "Mr",
"SecurityIdent": "",
"Sponsor": "",
"Street": "",
"SubCompany": "",
"TechnicalEntryDate": "1899-12-30T00:00:00.0000000Z",
"Title": "",
"UID_Department": "",
"UID_DialogCountry": "",
"UID_DialogCulture": "",
"UID_DialogState": "",
"UID_DialogUser": "",
"UID_FirmPartner": "",
"UID_Locality": "",
"UID_Org": "",
"UID_Person": "83b10e84-c64e-4f9f-9ecb-2d0d7c94e8ec",
"UID_PersonHead": "",
"UID_PersonMasterIdentity": "",
"UID_ProfitCenter": "",
"UID_RealPerson": "",
"UID_WorkDesk": "",
"UID_X500Person": "",
"UserIDTSO": "",
"XDateInserted": "2016-05-20T09:12:25.7300000Z",
"XDateUpdated": "2016-05-20T09:12:25.7300000Z",
"XMarkedForDeletion": 0,
"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 22: Change single entity request

HTTP Method	URI	Body

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

Table 23: Change single entity parameters

Parameter	Description	Parameter Type	Data Type
table	Table name (mandatory)	path	string
uid	Guid of the entity (mandatory)	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:

```
{}
```

Delete a single entity

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

Table 24: Delete single entity request

HTTP Method	URI	Body
Delete	<BaseURL>/api/entity/{table}/{uid}	None

Table 25: Delete single entity parameters

Parameter	Description	Parameter Type	Data Type
table	Table name (mandatory)	path	string
uid	Guid of the entity (mandatory)	path	string

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.

NOTE: This request type is able to execute customizer or dialog methods for an entity. Please be aware that currently only those methods are supported that do not return a value. The method name matching will be processed for customizer methods first. If none is 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

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

HTTP Method	URI	Body
		"Parameter value"] }

Table 27: Call method parameters

Parameter	Description	Parameter Type	Data Type
table	Table name (mandatory)	path	string
uid	Guid of the entity (mandatory)	path	string
methodName	Name of the method (mandatory)	path	string
parameters	Parameter values	body	object[]

Body schema:

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

Example 1

Execute a customizer method on an entity.

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

Response Code:

`204`

Example 2

Execute 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 28: 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 29: Generate event parameters

Parameter	Description	Parameter Type	Data Type
table	Table name (mandatory)	path	String
uid	Guid of the entity (mandatory)	path	String
eventName	Name of the event (mandatory)	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

Execute 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

4

Assignments

The following APIs are used to handle assignments in member tables for the entities of the 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

HTTP Method	URI	Body
Get	<BaseURL>/api/assignments/{table}/{column}/{uid}	None

Table 31: Get assignments parameters

Parameter	Description	Parameter Type	Data Type
table	Member table name (mandatory)	path	string
column	Column pointing to the base entity (mandatory)	path	string
uid	Guid of the base entity (mandatory)	path	string
where	WHERE clause	query	string
orderBy	ORDER BY clause	query	string
offset	Offset of first item	query	integer
limit	Maximum number of results	query	integer
loadType	Collection load type. Specify one of the values listed in table Values of paramter loadType on page 38.	query	string

Table 32: Values of paramter `loadType`

Value	Description
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 will be 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 indices.
BulkReadOnly	Loads read-only entities with all columns filled. The columns defined in the query will be overridden.
ForeignDisplays	Loads display values for foreign keys contained in the display pattern too. 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.

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 (JENNAA) - 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 33: Add assignments request

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

Table 34: Add assignments parameters

Parameter	Description	Parameter Type	Data Type
table	Member table name (mandatory)	path	string
column	Column pointing to the base entity (mandatory)	path	string
uid	Guid of the base entity (mandatory)	path	string
members	Guids of the members to add (mandatory)	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",
      "40e43904-4958-4bce-915b-f77bab675f06",
      "7c21b251-d774-4616-bc3a-b91506ddb23b"]
}
```

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

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

Table 36: Remove assignments parameters

Parameter	Description	Parameter Type	Data Type
table	Member table name (mandatory)	path	string

Parameter	Description	Parameter Type	Data Type
column	Column pointing to the base entity (mandatory)	path	string
uid	Guid of the base entity (mandatory)	path	string
members	Guids of the members to remove (mandatory)	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
}
```

5

Scripts

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

NOTE: The authenticated user must be entitled to use the program function "Allow the starting of arbitrary scripts from the frontend" in order to execute a script.

Run script request

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

Table 37: Run script request

HTTP Method	URI	Body
Put	<BaseURL>/api/script/{name}	{ "parameters": ["Parameter value"], "base": "XObjectKey", "value": "Sample value" }

Table 38: Run script parameters

Parameter	Description	Parameter Type	Data Type
name	Script name (mandatory)	path	string
parameters		body	object[]
base	Object key of base object	body	string
value	Content of the value variable in the script	body	object

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"
}
```

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)
Invoke-RestMethod -Uri "https://<Hostname>/AppServer/auth/apphost" -Body
$authJson.ToString() -Method Post -UseDefaultCredentials -Headers @
{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
Invoke-RestMethod -Uri "https://<Hostname>/AppServer/auth/logout" -WebSession
$wsession -Method Post
```

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