



---

# Group Calendar Microsoft

**OnTime MS Api  
Version 3**

**Revision 4**

OnTime is a registered community trademark (#004918124). The trademark is registered with the Trade Marks and Designs Registration Office of the European Union.

OnTime is a registered Japanese trademark (#5569584). The trademark is registered with the Japan Patent Office

Documentation for OnTime MS – Api

The audience for this documentation is developers with basic knowledge of the HTTP protocol and authentication.

It is also required some knowledge in how OnTime Group Calendar works in general.

API is divided in two parts: Authentication API and Data API.

## Contents

Contents .....	3
Authentication .....	5
Data API Calls .....	8
General .....	8
General information: .....	10
General object sets .....	10
Login .....	13
POST /Login .....	13
Settings .....	14
POST /Settings/Get .....	14
POST /Settings/Set .....	15
Group .....	16
POST /Group/List .....	16
POST /Group/Search .....	17
POST /Group/Details .....	17
POST /Group/Create .....	18
POST /Group/Update .....	18
POST /Group/Delete .....	18
POST /Group/SharedDetails .....	19
User .....	20
POST /User/List .....	20
POST /User/Search .....	20
POST /User/SearchPrivateContacts .....	21
POST /User/SearchEwsDirectory .....	21
POST /User/SearchGraphPeople .....	21
POST /User/SearchInDomainGroups .....	22
Room .....	23
POST /Room/List .....	23
POST /Room/Options .....	23
Equipment .....	24
POST /Equipment/List .....	24
POST /Equipment/Options .....	24
Event .....	25
POST /Event/List .....	25
POST /Event/Details .....	25
POST /Event/Create .....	26
POST /Event/Update .....	27
POST /Event/Delete .....	28
POST /Event/Notices .....	28
POST /Event/Accept .....	28
POST /Event/AcceptAll .....	29
POST /Event/Decline .....	29
POST /Event/DeclineAll .....	29
Free .....	30
POST /Free/Rooms .....	30
POST /Free/Equipments .....	30
POST /Free/Search .....	32

Mail.....	33
POST /Mail/Send .....	33
POST /Mail/SendMany .....	33
Image.....	34
POST /Image/Details.....	34
Folder.....	34
POST /Folder/Create.....	34
POST /Folder/List.....	35
POST /Folder/Delete.....	35
Contact .....	35
POST /Contact/Create .....	35
POST /Contact/Update .....	35
POST /Contact/List .....	36
POST /Contact/Delete .....	36
Version history:.....	36
11. nov. 2024 – api 3 rev 4 .....	36
5 apr. 2020 – api 3 rev 1 .....	37
5 feb. 2018 – api 2 rev 3 .....	37
8 jun. 2017 – api 2 rev 2 .....	37
29 aug. 2016 – api 2 rev 1.....	37
18 jan. 2016 - api 1 rev 4 .....	39
21 dec. 2015 - api 1 rev 3 .....	39
21 dec. 2015 - api 1 rev 2 .....	39
16 dec. 2015 - api 1 rev 1 .....	39

---

## Authentication

---

This section describes the Authentication API.

OnTime for Microsoft authentication can be configured in a number of ways. Methods of authentication include but are not limited to Form-Based, Basic, NTLM, Kerberos, and SAML claim-based. However describing all supported methods of authentication are beyond the scope of this document.

The authentication method described here is HTTP Basic authentication but the method of redirecting to an Identity Provider applies to all authentication methods. Enabling/disabling authentication methods are described in the OnTime Installation Guide.

OnTime for MS has two different login methods depending on configuration in the Admin, under “Global” – “Backend” – “Authentication”. If it is not configured, then the system uses “Form Based” authentication. If it is configured, then the system uses “Authentication Service” which resides on the URL provided in the configuration. The two authentication methods are different in terms of API and functionality, but their end result is the same. Both of them return an initial short lived OnTime Token that in turn is used to call the /Login endpoint (described in the Data API).

The /Login then returns another token, which can be then used for all other calls in Data API.

### *Authentication Service*

Authentication Service is a service which authenticates the user and then redirects (using 303 See Other) to the URL provided to it as a parameter, which is usually the application which requires the login. The info about login result is appended to the redirected URL.

If a user need to login, the browser should be redirected to the Authentication Service URL specified in the OnTime configuration, with the parameter “redirect={url}”, where {url} is the url the browser should be redirected to when authenticated, regardless of result.

When Authentication Service redirects to the URL (provided as a “redirect” parameter) it will append the “?response={response\_data}” to it, where the response\_data the base64 encoded json.

Example of response json follow (remember that they will be base64 encoded):

### Successful login

```
{
  "Status": "OK",
  "Token": "eyJUaW1lc3RhbXAiOjE0OTc0NDIzMjQzMTQsIkVtYWlsIjoibW5zQG9udGltZS5sb2NhbCJ9"
```

### Incorrect login

```
{
  "Status": "ERROR",
  "Error": {
    "Code": "AuthenticationError",
    "Description": "User not authenticated"
  }
}
```

If you as a developer need to use the Authentication Service outside the browser, you may use the Basic Authentication as described below:

#### Request

```
GET /ontime/auth.html?redirect=none HTTP/1.1
Host: localhost:80
Accept: */*
Authorization: Basic am9obmRvZUBhY211LmNvbTpPblRpbWUzMjM=
```

PS: Authorization header contains a base64 encoding of {user}:{password}

#### Response

```
HTTP/1.1 303 See Other
Location: none?response=eyJTdGF0dXMiOiJPSyIsIlRva2VuI...
```

Where the response= value from the "Location" header is the {response\_data} like above.

If your environment automatically follows 303 redirects, then the {response\_data} can be retrieved from the redirected url parameter "response=".

In case of invalid user/password it could return

#### Response

```
HTTP/1.1 401 Unauthorized
Content-Type: text/html
Date: Wed, 14 Jun 2017 14:20:05 GMT
Server: Microsoft-HTTPAPI/2.0
WWW-Authenticate: Negotiate
WWW-Authenticate: NTLM
WWW-Authenticate: Basic realm="OnTimeSuite"
```

Where the "WWW-Authenticate" tells valid authentication methods supported.

---

## Data API Calls

---

### General

This section describes the Data API.

All calls are as standard POST. Sometimes the call is implemented as GET to simplify caching of resources, for example images.

The main concept of all POST calls is “json in – json out”-encoded in UTF-8. Server always returns http code 200 OK.

The prefix URL for all Api calls is “[contextPath]/api/[apiVersion]/[data]/...”

Where

[contextPath] is the protocol, host, port, installPath defined by setup.

[apiVersion] is the api version for the request – currently ONLY 2 is supported.

The request is a Json string with some mandatory elements (“ApplID”, “ApplVer”, “Token”) to identify request, and a Data object with several request specific elements.

Result is json data with fields “Status”, “Token” and “Data” (result of request)

Example:

#### Request

```
{
  "ApplID" : "ApiExplorer"   String*   The application id according to license
  "ApplVer" : "1.0.0"       String   The application version, only for logging
  "Token" : "xxxxx..."    String*   The user token from previous response
  "OnBehalfOf" : "..."    String*
  "Data" : {                Object*   Object with request elements
  }
}
```



### Response

```
{
  "Status" : "OK",
  "Token" : "xxxx...",
  "Data" : {
    ::
  }
}
```

```
{
  "Status" : "ERROR",
  "Error" : {
    "Code" : "InvalidToken",
    "Description" : "The provided token is invalid",
    ::
  }
}
```

If a return Token present, this should be use during next call to API, as server can issue new token whenever needed.

Status can be "OK" or "ERROR". In case of "ERROR" an error object with "Code" and a "Description" will be provided, where the "Code" is a short unique code for the error, and can be used to identify how to handle the error, and as an index for a translated description. "Description" is an English description of error, if no translation is available.

Other Error info can be provided for tracing/debugging purpose

Note: description of all of the API calls will mostly show only the Data section, except places where the Status is also relevant.

## General information:

- \* Indicates a mandatory field
- DateTime format** Is following format, and only Zulu time: YYYY-MM-DDThh:mm:ssZ
- Boolean** lowercase true or false (no quotes)

## General object sets

Following is a list of “standard” objects that the request/response may use – See each API call for details and reference to these objects

### {User}

```

{
  "UserID"           String
  "Email"           String
  "Type"            String           "Person"|"Room"|"Equipment",
  "ContactSource"   String           "ActiveDirectory"|???
  "CanCreate"       Boolean
  "DeleteItems"     String           "All"|"None"
  "ReadItems"       String           "FullDetails"|"TimeOnly"|???
  "EditItems"       String           "All"|"Owned"|"None"
  "DisplayName"     String
  "FirstName"       String
  "LastName"        String
  "CompanyName"     String
  "Department"      String
  "JobTitle"        String
  "MobilePhone"     String
  "Culture":        String           Language selected on Exchange profile
  "Photo"           String           Base64 representation of a small jpg image.
  "ImageType"       String           "jpg"|"png"|"gif"
  "ImageChangeKey": String           Value that changes if image is updated.
  "WorkHours"       Json
  "DomainID"        String
  "DomainName"      String
  "CateringEnabled" Boolean
  "PhoneticDisplayName" String
}

```

**{Group}**

```
{
  "GroupID"      String
  "Name"         String
  "MembersCount" Number
  "DisplayName"  String      Only available for Distribution List groups
  "Private"      Boolean     Only available if a private group
  "Application"  Boolean     Only available if an application group
  "SortManually" Boolean     Only available manually sorted group
  "SortOrder"    String[]
  "UserIDs"      String[]
}
```

**{Event}**

```

{
  "EventID"           String
  "Start"             Date
  "End"               Date
  "Subject"           String
  "ShowAs"            String      "Away"|"Busy"|"Tentative"|"Free"|"WorkingElsewhere"
  "Sensitivity"       String      "Normal"|"Private"
  "Categories"        String[]
  "Type"              String      "Meeting"|"Invitation"
  "ChangeKey"         String
  "Organizer"         String      Here it's only Display Name
  "RequiredAttendees" String[]    Here it's only Display Name
  "OptionalAttendees" String[]   Here it's only Display Name
  "IsCancelled"       Boolean
  "IsRecurring"       Boolean
  "MyResponseType"    String
  "Location"          String
  "CreatorID"         String
  "LegendID"          Integer
  "OOO"               Boolean      Time off
  "PollID"            String
  "ReminderSet"       Boolean
  "ReminderMinutesBeforeStart" Number
}

```

**{Contact}**

```

"FolderId"           String
"ContactItemId"      String
"GivenName"          String
"MiddleName"         String
"Surname"            String
"CompanyName"        String
"EmailAddress1"      String
"EmailAddress2"      String
"EmailAddress3"      String
"Department"         String
"DisplayName"        String
"Initials"           String
"JobTitle"           String
"OfficeLocation"     String
"NickName"           String
"Profession"         String
"Body"               String
"Culture"            String
"Subject"            String
"PhysicalAddressHome" PhysicalAddressEntryContact
"PhysicalAddressBusiness" PhysicalAddressEntryContact
"PhysicalAddressOther" PhysicalAddressEntryContact
}

```

Where PhysicalAddressEntryContact is:

```

{
  "Street"           String
  "City"             String
  "State"            String
  "PostalCode"       String
  "CountryOrRegion" String
}

```

## Login

### POST /Login

Used to verify a Token and ApplID, and getting current user and some server info.

This is the only call that accepts the “onetime” token return from the Authentication.

If this call is called with the API User and OnBehalfOf then the User Token is returned in the UserToken field. The token created with an APIUser is not renewable.

If is called with an API User and no OnBehalfOf specified, then no User Data is returned.

### Request

```
"Data" : {}
```

### Response

#### Ex 1 – Valid Token:

```
{
  Status: "OK",
  Token: "...",
  Data: {
    Me: {User},           Object      UserDetails for current logged in user.
    Server: {
      Version: "...",
      SyncBackTime: "...",
      SyncForwardTime: "...",
      ExchangeFeatures:           1 = Basic
                                   2 = supports ShowAs "WorkingElsewhere", Event Body info"
    }
    pollarity_availability: Boolean
    teams_availability: Boolean
    graph_availability: Boolean
    allow_conflicts_on_rooms: Boolean
  }
}
```

#### Ex 2 – Token Expired:

```
{
  "Status" : "ERROR",
  "Error" : {
    "Code" : "TokenExpired",
    "Text" : "Token has been expired"
  }
}
```

#### Typical ErrorCodes:

TokenInvalid

TokenIsEmpty

TokenExpired

Ex 3 – when called with an API User and OnBehalfOf:

```
{
  "Me": {
    <UserData>
  },
  "Server": {
    <ServerData>
  },
  "UserToken": "<UserToken>",
}
```

## Settings

### POST /Settings/Get

Get settings for given Application and all Admin settings

Please note that Get and Set formats are different.

### Request

```
"Data" : {}
```

### Response

Ex - ApplID eq "Desktop"

```
"Data" : {
  "Admin" : {
    "DefaultSettings"  Object
    "Legends"          Object
    ::                 Some few extra only relevant for the Admin client.
  },
  "ApiExplorer" : {
    "[Key1]"           Object
    ::
    "[KeyN]"           Object
  }
}
```

**POST /Settings/Set**

Save settings for given application. Settings are list of Key/Object pair, where key can be any json key

**Request**

```
"Data" : {  
  "[Key1]"      Object  
  ::  
  "[KeyN]"      Object  
}
```

**Response**

```
"Data" : {}
```

## Group

### POST /Group/List

Return a list of groups, according to parameters.

No parameters mean a list of ALL groups, available for user.

#### Request

```
"Data" : {
  "IncludePublic"      Boolean
  "IncludePrivate"    Boolean
  "Partial"            String      Find group entries matching start (only for public)
}
```

#### Response

Ex - No Param

```
"Data" : {
  "Items" : [
    {Group},
    ::
  ]
}
```

Ex - "Partial":"" and "IncludePublic":true

Partial groups do not include information about subgroups, and are only showing the partial name. This is done to reduce the bandwidth and load the group structure only when the partial group is opened from UI.

```
"Data" : {
  "Items" : [
    { "Name" : "A", "GroupID" : "..." },
    { "Name" : "B\\" },
    { "Name" : "C\\" },
    { "Name" : "D", "GroupID" : "..." },
  ]
}
```

Ex - "Partial":"B\\" and "IncludePublic":true

```
"Data" : {
  "Items" : [
    { "Name" : "B\\1", "GroupID" : "..." },
    { "Name" : "B\\2", "GroupID" : "..." },
    { "Name" : "B\\3\\" },
    { "Name" : "B\\Test\\" }
  ]
}
```



## POST /Group/Search

Following patterns do only a substring search. Regexp or wildcards are not supported. Search is done by Group Name and Display Name.

### Request

```
"Data" : {  
  "Pattern"      String      Search for groups that match,  
  "MaxCount"    Number      Default: 10  
  "ExactMatch"  Boolean  
}
```

### Response

```
"Data" : {  
  "Items" : [  
    {Group},          Group Object  
    ::  
  ]  
}
```

## POST /Group/Details

Return details of a given group.

### Request

```
"Data" : {  
  "GroupID"      String*}
```

### Response

```
"Data" : {Group}      Group Object  
}
```

## POST /Group/Create

Create a Group with given Name and user IDs, and return the new GroupID.

### Request

```
"Data" : {
  "Name"           String*           (Empty string is not allowed)
  "UserIDs"        String[]*
  "SortManually"   Boolean           (default: false)
  "Application"    Boolean           (default: false)
  "SharedWith"     String[]          User IDs
  "ManagedBy"     String[]          User IDs
  "SharedWithAllMembers" Boolean
  "Shared"         Boolean
}
```

### Response

```
"Data" : {
  "GroupID":      String
}
```

## POST /Group/Update

Update existing group

### Request

```
"Data" : {
  "GroupID"        String*
  "Name"           String
  "UserIDs"        String[]
  "SortManually"   Boolean
  "SharedWith"     String[]          User IDs
  "ManagedBy"     String[]          User IDs
  "SharedWithAllMembers" Boolean
  "Shared"         Boolean
}
```

### Response

```
"Data" : {}
```

## POST /Group/Delete

Remove a given group.

### Request

```
"Data" : {
  "GroupID"        String*
}
```

**Response**

```
"Data" : {}
```

**POST /Group/SharedDetails**

Group sharing details

**Request**

```
"Data" : {  
  "GroupID"      String*  
}
```

**Response**

```
"Data" : {  
  "GroupID"      String*  
  "SharedWith"   String[]      User IDs  
  "ManagedBy"   String[]      User IDs  
  "SharedWithAllMembers" Boolean  
  "Shared"       Boolean  
}
```

## User

### POST /User/List

Returns user details of a list of UserIDs, Emails addresses or a GroupID.

#### Request

```
"Data" : {
  "UserIDs"      String[]
  "Emails"       String[]
  "GroupID"      String
  "SyncedAfter"  ZonedDateTime
}
```

Note: Only one of the fields should be provided. Combining "UserIDs", "Emails", "GroupID" has an undefined result: it may return one result, all results or it may fail with an error.

#### Response

```
"Data" : {
  "Items" : [
    {User}          User Object
    ::
  ],
  "SortOrder" : ["userID",... ]    Only if by GroupID and group is manually sorted.
}
```

### POST /User/Search

Search for user by Pattern, and return MaxCount of results. Pattern does not support Regex or wildcards and search is done by a substring search by using First Name, Last Name, and Email.

#### Request

```
"Data" : {
  "Pattern"      String*      (future split with space)
  "UserType"     String       "Person"|"Room"|"Equipment" (none means all)
  "MaxCount"     Number       Default 10 - Absolut max ? (100)
}
```

### Response

```
"Data" : {
  "Items" : [
    {User},      User Object
    ::
  ]
}
```

### POST /User/SearchPrivateContacts

#### Request

```
"Data" : {
  "Pattern"      String*      (future split with space)
  "UserType"     String       "Person"|"Room"|"Equipment" (none means all)
  "MaxCount"     Number       Default 10 - Absolut max ? (100)
}
```

### Response

```
"Data" : {
  "Items" : [
    {
      "Email"      String
      "DisplayName" String
    }
  ]
}
```

### POST /User/SearchEwsDirectory

#### Request

```
"Data" : {
  "Pattern"      String*
}
```

### Response

```
"Data" : {
  "Items" : [
    {
      "Email"      String
      "DisplayName" String
    }
  ]
}
```

### POST /User/SearchGraphPeople

#### Request

```
"Data" : {
  "Pattern"      String*
}
```

**Response**

```
"Data" : {
  "Items" : [
    {
      "Email"      String
      "DisplayName" String
    }
  ]
}
```

**POST /User/SearchInDomainGroups****Request**

```
"Data" : {
  "GroupID"      String*
}
```

**Response**

```
"Data" : {
  "Items" : [
    {User},      User Object
  ]
}
```

## Room

### POST /Room/List

Returns a list of all Rooms.

#### Request

```
"Data" : {}
```

#### Response

```
"Data" : {  
  "Items" : [  
    {User},           User Object  
    ::  
  ]  
}
```

### POST /Room/Options

Returns a list of unique Locations from all rooms

#### Request

```
"Data" : {}
```

#### Response

```
"Data" : {  
  "Locations" : [...] String[]  Array of unique Locations from all rooms.  
}
```

## Equipment

### POST /Equipment/List

Returns a list of all Equipment's.

#### Request

```
"Data" : {}
```

#### Response

```
"Data" : {  
  "Items" : [  
    {User},           User-object  
    ::  
  ]  
}
```

### POST /Equipment/Options

Returns a list of Locations for all equipment

#### Request

```
"Data" : {}
```

#### Response

```
"Data" : {  
  "Locations" : String[]   Array of unique Locations from all equipment.  
}
```



## Event

### POST /Event/List

#### Request

```
"Data" : {
  "UserIDs"      String[]*
  "From"         Date*
  "To"           Date*
}
```

#### Response

```
"Data" : {
  "Items" : [
    {
      "Events" : [
        "UserID"      String      UserID as in request
        {Event}      Event-Object
      ]
    }
  ]
}
```

### POST /Event/Details

Events retrieved from POST /Event/List include only basic information (described in General information section). /Event/Details call is used to get extended information about event.

#### Request

```
"Data" : {
  "EventID"      String*
}
```

**Response**

```

"Data" : {
  "Event" : {      Event-object extended with the following fields:
    ::
    "RequiredAttendees"   String[]      Array of emails of Required attendees
    "OptionalAttendees"  String[]      Array of emails of Optional attendees
    "Body"                String          The content of body
    "EffectiveRights" : {
      "CreateAssociate"   Boolean
      "CreateContents"    Boolean
      "CreateHierarchy"   Boolean
      "Delete"            Boolean
      "Modify"            Boolean
      "Read"              Boolean
      "ViewPrivateItems" Boolean
    },
    "ShowAs"              LegacyFreeBusyStatus ("Free","Tentative","Busy","OOE","WorkingElsewhere",
    "NoData")
    "Type"                String
    "BodyHTML"            BodyHtml {
      String SessionId,
      String HTML,
      AttachmentInfo attachments
      (AttachmentInfo{
        boolean Inline,
        String ContentType,
        String Name,
        String ContentID,
        Long Size,
        String UrlID,
        boolean Deleted
      })
    }
    "ResponseStatuses"    ResponseStatus[] ResponseStatus{
      String Email,
      String ResponseType
    }
    "Recurrence"          RecurrenceContainer RecurrenceContainer{
      Integer Type,
      Integer NumberOfOccurrences,
      ZonedDateTime StartDate,
      ZonedDateTime EndDate,
      RecurrenceParameters Parameters (
        RecurrenceParameters{
          Integer DayOfTheWeek,
          Integer DayOfTheWeekIndex,
          Integer DayOfMonth,
          Integer[] DayOfTheWeekCollection,
          Integer Month,
          Integer Interval
        }
      )
    }
  }
}

```

**POST /Event/Create**

Note: When properties are not provided then the defaults are used

### Request

```
"Data" : {
  "UserID"           String*
  "Start"            Date*
  "End"              Date*
  "Subject"          String (max 255 chars) (default: empty)
  "ShowAs"           String (default: "Busy")
  "Sensitivity"      String (default: "Normal")
  "Body"             String (newline "\n" as line separator) (default: empty)
  "Categories"       String[] (default: none)
  "Location"         String (default: empty)
  "RequiredAttendees" String[] (max 255 chars pr. email) (default: none)
  "OptionalAttendees" String[] (max 255 chars pr. email) (default: none)
}
```

### Response

```
"Data" : {
  "Event" : {Event} Event-Object
}
```

## POST /Event/Update

Note: When fields are not set then their value does not change

### Request

```
"Data" : {
  "EventID"          String*
  "ChangeKey"        String
  "Start"            Date
  "End"              Date
  "Subject"          String (max 255 chars)
  "Location"         String
  "Categories"       String[]
  "ShowAs"           String
  "Sensitivity"      String
  "RequiredAttendees" String[] (max 255 chars per email)
  "OptionalAttendees" String[] (max 255 chars per email)
}
```

### Response

```
"Data" : {
  "Event" : {Event} Event-object
}
```

## POST /Event/Delete

### Request

```
"Data" : {
  "EventID"      String*
  "ChangeKey"    String
}
```

### Response

No Data, only Status

## POST /Event/Notices

### Request

```
"Data" : {
  "UserIDs"      String[]*
}
```

### Response

```
"Data" : {
  "Items" : [
    {
      "Events" : [
        {
          "UserID"      String      UserID as in request
          {Event}      Event-Object
        }
      ]
    }
  ]
}
```

## POST /Event/Accept

### Request

```
"Data" : {
  "EventID"      String
  "Subject"      String
  "ShowAs"       LegacyFreeBusyStatus
  "Sensitivity"  (Normal, Personal, Private, Confidential)
  "Category"     String[]*
  "Location"     String
}
```

### Response

```
"Data" : {
  "Event"        Details (extended Event from details call)
}
```

**POST /Event/AcceptAll****Request**

```
"Data" : {
  "EventID"      String
  "Subject"      String
  "ShowAs"       LegacyFreeBusyStatus
  "Sensitivity"  (Normal, Personal, Private, Confidential)
  "Categories"   String[]*
  "Location"     String
}
```

**Response**

```
"Data" : {
  "Event"        Details (extended Event from details call)
}
```

**POST /Event/Decline****Request**

```
"Data" : {
  "EventID"      String
}
```

**Response**

```
"Data" : {}
```

**POST /Event/DeclineAll****Request**

```
"Data" : {
  "EventID"      String
}
```

**Response**

```
"Data" : {}
```

## Free

### POST /Free/Rooms

Get a list of free Rooms, within a given time and options.

If a capacity is set then all the rooms with equal or greater capacity are returned as well as rooms which capacity is unknown (not set).

If JoinedRooms flag is set to true, then joined rooms will also be returned in the call.

#### Request

```
"Data" : {
  "From"      Date*
  "To"        Date*
  "Capacity"  String
  "Location"  String
  "DisplayName" String
  "Building"  String
  "Floor"     Number
  "AlwaysShowNoCapacity" Boolean (=true)
  "JoinedRooms" Boolean
}
```

#### Response

```
"Data" : {
  "UserIDs"  String[]
}
```

### POST /Free/Equipments

Get a list of free Equipment's, within a given time and options.

If a capacity is set, then the equipment with equal or greater capacity is returned as well as equipment which capacity is unknown (not set).

#### Request

```
"Data" : {
  "From"      Date*
  "To"        Date*
  "Capacity"  String
  "Location"  String
}
```

**Response**

```
"Data" : {  
  "UserIDs"    String[]  
}
```

## POST /Free/Search

Finds free slots for a number of users in a given timeframe. This can be used to find appropriate time for meetings.

### Request

```
"Data" : {  
  "From"      Date*  
  "To"        Date*  
  "UserIDs"   String[]*  
}
```

### Response

```
"Data" : {  
  "Items" : [  
    {  
      Start      Date  
      End        Date  
    },  
    ::  
  ]  
}
```

### Comment

This call does not take into account users' working hours. It is a client responsibility to filter out the results, which do not fit the users' working hours.



## Mail

### POST /Mail/Send

Send a mail to a user.

{Mail}

```
{
  "To"           String* (Email of receiver)
  "Subject"      String
  "Body"         String (newline "\n" are line separator)
}
```

### Request

```
"Data" : {
  Mail
}
```

### Response

```
"Data" : {}
```

No Data, only a Status.

### POST /Mail/SendMany

Send many mails

### Request

```
"Data" : {
  [Mail, Mail...]
}
```

### Response

```
"Data" : {}
```

No Data, only a Status.

## Image

### POST /Image/Details

Request base64 representation of a user's image.

In case the image requested size is not available then the nearest is returned. Known-sizes are currently 48x48 and 96x96

#### Request

```
"Data" : {  
  "UserID"      String*  
  "Width"       Number*  
  "Height"      Number*  
}
```

Note: Sizes can currently only be "Width:48, Height:48" or "Width:96, Height:96"

#### Response

```
"Data" : {  
  "Base64"      String  
  "Width"       Number  
  "Height"      Number  
  "Type"        String      ("jpg"|"png"|"gif")  
}
```

## Folder

### POST /Folder/Create

#### Request

```
"Data" : {  
  "FolderId"   String*  
  "Name"       String*  
}
```

#### Response

```
"Data" : {  
  "FolderId"   String  
  "Name"       String  
}
```

## POST /Folder/List

### Request

```
"Data" : {}
```

### Response

```
"Data" : {  
  [  
    {  
      "FolderId"    String  
      "Name"       String  
    },  
    ...  
  ]  
}
```

## POST /Folder/Delete

### Request

```
"Data" : {  
  "FolderId"    String*  
}
```

### Response

```
"Data" : {}
```

## Contact

## POST /Contact/Create

### Request

```
"Data" : { Contact }
```

### Response

```
"Data" : {  
  "Contact" : { Contact }    Contact -object  
}
```

## POST /Contact/Update

### Request

```
"Data" : { Contact }
```

### Response

```
"Data" : {  
  "Contact" : { Contact }    Contact -object  
}
```

**POST /Contact/List****Request**

```
"Data" : {  
  "FolderId"      String*  
}
```

**Response**

```
"Data" : {  
  "Items" : [ {Contact}, ... ]      Contact -object  
}
```

**POST /Contact/Delete****Request**

```
"Data" : {  
  "ContactItemId" String
```

**Response**

```
"Data" : {}
```

---

*Version history:*

---

11. nov. 2024 – api 3 rev 4

Error in pdf's index

17 jun. 2021 – api 3 rev 3

Added ReminderSet and ReminderMinutesBeforeStart to Event object

Login call:

graph\_availability: Boolean

allow\_conflicts\_on\_rooms: Boolean

Free/Rooms:

"DisplayName" String

"Building" String

"Floor" Number

"AlwaysShowNoCapacity" Boolean (=true)

"JoinedRooms" Boolean

/Mail/SendMany

Users

SearchEwsDirectory

SearchGraphPeople

*5 apr. 2020 – api 3 rev 1*

Added: Folder, Contact

Changed: Image

*5 feb. 2018 – api 2 rev 3*

Authentication section rewritten

Form based authentication section removed

Minor changes, corrections, and clarifications

*8 jun. 2017 – api 2 rev 2*

Authentication section added.

New layout of this document.

SortManually added on the Groups requests.

Made a General object sets section.

*29 aug. 2016 – api 2 rev 1*

'.../Room/Options': New.

'.../Equipment/List': New.

'.../Equipment/Options': New.

'.../Free/Rooms': New.

'.../Free/Equipments': New.

'.../Free/Search': New.

'.../Group/Search': New

'.../Mail/Send': New

'.../Image/Get': New.

'.../Image/Details': New.

'.../User/List':

Added GroupID in request

Added OfficeLocation

Added ImageType

Added ImageChangeKey

'.../User/Search':

Added UserType

'.../Group/List':

Removed Seperator,

Partial implemented,

Added InclPrivate, InclPublic

Added MembersCount

Added Application

'.../Group/Details':

Returns UserIDs instead of User objects

Application added

'.../Group/Create':

Application added

*18 jan. 2016 - api 1 rev 4*

'.../Event/Details'

'.../Group/Members' renamed to '.../Group/Details'

*21 dec. 2015 - api 1 rev 3*

"Data" in request, and fixup

*21 dec. 2015 - api 1 rev 2*

Multiple updates

*16 dec. 2015 - api 1 rev 1*

First version of this documentation