Alfresco

The Alfresco API

Contents

The Alfresco API

The Alfresco API lets you access content managed by Alfresco Cloud from your own applications. The API is RESTful, which means each call is an HTTP request, so you don't even need a programming language to try it out. You can just type a URL address in a web browser. It consists of two parts, the standard CMIS API, which lets you manage and access content, and the new Alfresco REST API which lets you manage Alfresco's additional features such as ratings and comments, that are not covered by the CMIS standard.

All you need to get started is an Alfresco Cloud account. Once you have that, you can try out API calls using:

- A web browser
- An HTTP URL tool such as cURL or RESTClient. Some of these tools let you build your GET, PUT, POST, and DELETE commands simply, take care of authentication, and will save your test calls for repeated use.

You make API requests by sending a URL using one of five HTTP API methods, GET, POST, PUT, DELETE, and OPTIONS. Here's an example:-

```
https://api.alfresco.com/yourcompany.com/public/alfresco/versions/1/sites/fred-bloggs-yourcompany-com
```

Sending this URL using the HTTP GET method invokes the `sites` Alfresco Public RESTFul method. This call requests information on the site with the id fred.blogs.yourcompany.com. The server will return an HTTP response with the following JSON body :-

```
{
   "entry":{
      "title":"Fred Blogg's Home",
      "description":"Fred Blogg's private home site.",
      "visibility":"PRIVATE",
      "id":"fred-bloggs-yourcompany-com"
   }
}
```

How does an application do work on behalf of a user?

You can register your application with Alfresco to use our authentication.

An Alfresco application uses the OAuth 2.0 authorization code flow to authenticate itself with Alfresco Cloud and to allow users to authorize the application to access data on their behalf.

You first register your application on the Alfresco Developer site. You provide a callback URI, and a scope. Registration will provide you with an API key and a key secret which are required by your application to authorize itself. When a user runs your application, the application requests an authorization code from Alfresco using its API key, key secret, callback URI and scope. Alfresco will inform the user that your application wishes to access resources, and asks the user to grant or deny access.

If the user grants access, Alfresco returns an authorization code to the application. Your application then exchanges the authorization code for an access token. Your application can then call the Alfresco CMIS API and the Alfresco REST API with the access token.

Registering your application

To use the Alfresco API, your application must first be registered with the Alfresco Developer Portal.

Go to https://developer.alfresco.com and sign up. When your account is approved, you can register your application. You must provide a callback URL and a scope. The callback URL is the part of your application that exchanges the authorization code for an access token. The scope should always be set to `public_api`. call.

Once your application is approved, The **Auth** tab will show an **API Key**, and a **Key Secret**. These will be needed by your application for authorization.

Authorization

Your application uses the information registered with Alfresco to authorize itself when it is run by a user.

Requesting an authorization code

The following HTML is from the Alfresco OAuth sample and shows an application with a API Key (client_id) of `l74dx104ddc00c3db4509b2d02f62c3a01234` , a redirect URI of `http://localhost:8080/alfoauthsample/mycallback.html` and a scope of `public_api` authorizing with Alfresco. You should always use the value `public_api` for scope.

```
<!DOCTYPE html>
<html>
<head>
<title>Alfresco OAuth Sample Demo</title>
<meta http-equiv="Pragma" content="no-cache">
<meta http-equiv="Expires" content="-1">
</head>
<body>
<h1>Welcome to the Alfresco OAuth Sample App</h1>
<form action="https://api.alfresco.com/auth/oauth/versions/2/authorize">
client_id: <input
   name="client_id"
   value="l74dx104ddc00c3db4509b2d02f62c3a01234"
   size="50px"
>
This must match the registered value
```

```
<br />
redirect_uri: <input
   name="redirect_uri"
   value="http://localhost:8080/alfoauthsample/mycallback.html"
   size="70px"
>
* This must match the registered value
<br />
scope: <input
   name="scope"
   value="public_api"
>
<br />
response_type: <input
   name="response_type"
   value="code"
   readonly="readonly"
><br />
<input type="submit"></form>
</html>
```

Alfresco will ask the user for their userid and password to grant or deny access to resources for your application. If they grant access, then Alfresco will invoke the callback URI with the authorization code.

Exchanging the authorization code for an access token

Once the application has an authorization code, it can exchange this for an access token. The following HTML is from the Alfresco OAuth sample and shows an application with an authorization code of `f9d9f182-700b-4c67-8235-b6ea08870872` API Key (client_id) of `l74dx104ddc00c3db4509b2d02f62c3a01234` , and a key secret (client_secret) of `ebf0708b9c8a46efb0115024a7a204e0` requesting an access token. Note that once the application has an authorization code, it has 10 minutes to exchange it. After that, the authorization code is invalid and the application must request a new one.

```
<!DOCTYPE html>
<html>
<head>
<title>OAuth Callback page</title>
<meta http-equiv="Pragma" content="no-cache">
<meta http-equiv="Expires" content="-1">
</head>
<body>
<h1>OAuth Sample - Callback page</h1>
<form id="tokenForm" action="https://api.alfresco.com/auth/oauth/versions/2/
token" method="post" target="ipostresponse">
code: <input id="authCode" name="code" value="f9d9f182-700b-4c67-8235-
b6ea08870872" size="50px"><br/>
client_id: <input name="client_id"
 value="l74dx104ddc00c3db4509b2d02f62c3a01234" size="50px">
* This must match the registered value in the developer portal</font><br/>

client_secret: <input name="client_secret"
 value="ebf0708b9c8a46efb0115024a7a204e0" size="50px">
* This must match the registered value in the developer portal</font><br/>

redirect_uri: <input name="redirect_uri" value="http://localhost:8080/
alfoauthsample/mycallback.html" size="70px">
* This must match the registered value in the developer portal</font><br/>
grant_type: <input name="grant_type" value="authorization_code"
 readonly="readonly"><br/>
<input type="submit">
</form>
```

```
</html>
```

The application will get a JSON response body like this:

```
{
  "access_token":"87727764-3876-43b9-82a1-1ca917302ce5",
  "token_type":"Bearer",
  "expires_in":3600,
  "refresh_token":"596f6074-f432-4aeb-a162-8196213c659c",
  "scope":"public_api"
}
```

The following table explains the response properties :-

Property	JSON Type	Description
access_token	string	An access token that can be used to make authenticated calls using the Alfresco API for one hour.
token_type	string	The type of token.
expires_in	number	The number of seconds the access token will be valid for. Alfresco will issue access tokens valid for one hour.
refresh_token	string	Once the access token expires, the application must Refreshing an access token on page 7 using this refresh token. The refresh token is valid for seven days.
scope	string	Always use public_api as the value of scope.

The access token can be used to make authenticated calls using the Alfresco API for one hour. After that period, the application must Refreshing an access token on page 7 using the refresh token.

Using the access token

For simplicity the example below adds the access token to the query as a parameter. Note that the preferred method to pass the access token to Alfresco is to include it in the HTTP request header in the `Authorization` field in the following format:

```
            Value: Bearer [your access token]
```

This is a an example:

```
Bearer d1358c05-6564-4086-94b6-a7e14ce3490
```

The application now has an access token, and can use it to make API calls. The following HTML code is from the Alfresco OAuth sample and shows an authenticated call to the `sites` API.

```
<!DOCTYPE html>
<html>
<head>
<title>Alfresco OAuth Sample Demo</title>
<meta http-equiv="Pragma" content="no-cache">
<meta http-equiv="Expires" content="-1">
</head>
<body>
<h1>OAuth Sample - Use the access token</h1>
<form id="callerForm" action="" method="get" target="ipostresponse">
Paste your Access token here: <input name="access_token" value=""
 size="60px"><br/>
API url to call (via HTTP.GET) <input id="urlToCall" value="https://
api.alfresco.com/alfresco.com/public/alfresco/versions/1/sites"
 size="70px"><br/>
<input type="submit">
```

```
</form>
</body>
</html>
```

The application will get a JSON response body like this:

```
{
  "list" : {
    "pagination" : {
      "count" : 2,
      "hasMoreItems" : false,
      "skipCount" : 0,
      "maxItems" : 100
    },
    "entries" : [ {
      "entry" : {
        "site" : {
          "id" : "general-test-site",
          "title" : "General Test Site",
          "visibility" : "PRIVATE",
          "description" : "Test Site"
        },
        "id" : "general-test-site",
        "role" : "SiteCollaborator"
      }
    }, {
      "entry" : {
        "site" : {
          "id" : "fred-bloggs-yourcompany-com",
          "visibility" : "PRIVATE",
          "description" : "Fred Bloggs's private home site."
        },
        "id" : "fred-bloggs-yourcompany-com",
        "role" : "SiteManager"
      }
    } ]
  }
}
```

Refreshing an access token

After one hour, your application's access token will be invalid. You can use the refresh token to request a new access token without having to re-authenticate with the user. The refresh token is valid for 7 days or until a new access token is requested.

When the access token expires, API requests will receive an HTTP 401 response with the following body:

```
{
  "error":"invalid_request",
  "error_description":"The access token expired"
}
```

The error description `The access token expired` is the only way your application can recognize this error. Your application should request a new access token using the refresh token.

The following HTML is from the Alfresco OAuth sample and shows an application with a refresh token of `e98f372c-e5a6-49e5-ba55-a035234577eb2` API Key (client_id) of `l74dx104ddc00c3db4509b2d02f62c3a01234`, and a key secret (client_secret) of `ebf0708b9c8a46efb0115024a7a204e0` requesting a new access code.

```
<!DOCTYPE html>
```

```
<html>
<head>
<meta charset="UTF-8">
<title>Alfresco OAuth Sample Demo</title>
<meta http-equiv="Pragma" content="no-cache">
<meta http-equiv="Expires" content="-1">
</head>
<body>
<h1>OAuth Sample - Refresh the access token</h1>
<form id="tokenForm" action="https://api.alfresco.com/auth/oauth/versions/2/
token" method="post" target="ipostresponse">
refresh_token: <input name="refresh_token" value="e98f372c-e5a6-49e5-ba55-
a035234577eb2" size="60px"><br/>
client_id: <input name="client_id"
 value="l74dx104ddc00c3db4509b2d02f62c3a01234" size="50px">
* This must match the registered value in the developer portal<br/>
client_secret: <input name="client_secret"
 value="ebf0708b9c8a46efb0115024a7a204e0" size="50px">
* This must match the registered value in the developer portal<br/>
grant_type: <input name="grant_type" value="refresh_token"
 readonly="readonly"><br/>
<input type="submit">
</form>
</body>
</html>
```

The response will have a body that looks like this:

```
{
  "access_token":"28f88a82-a62b-4e44-9312-16a4a5d2e71c",
  "token_type":"Bearer",
  "expires_in":3600,
  "refresh_token":"e98f372c-e5a6-49e5-ba55-a0358d877eb2",
  "scope":"public_api"
}
```

Note that you can refresh the access token at any time before the timeout expires. The old access token becomes invalid when the new one is granted. The new refresh token supplied in the response body can be used in the same way.

Alfresco CMIS API

CMIS (Content Management Interoperability Services) is a vendor-neutral OASIS Web services interface specification that enables interoperability between Enterprise Content Management (ECM) systems. CMIS allows rich information to be shared across Internet protocols in vendor-neutral formats, among document systems, publishers and repositories, in a single enterprise and between companies.

You can use basic HTTP methods to invoke CMIS methods, or you can use one of the many language-specific libraries that wrap CMIS. One such example for the Java language is the OpenCMIS Client API provided by the Apache Chemistry project. Apache Chemistry provides client libraries for many other languages such as Python, PHP, and .NET.

Getting Started

To get you started with CMIS, this section explains the format of the URL you will use, and what responses to expect.

Note when reading this documentation and any other information on CMIS, that the CMIS term **repository** maps directly to the Alfresco Cloud term **network**.

The domain model

CMIS defines a domain model. A client will access a CMIS service endpoint described by a URL. A service endpoint must have at least one repository. A repository, in this case an instance of Alfresco, is a data store which contains content. Each item of content is an object such as a folder, or a document. A repository is identified by its ID, and has a set of capabilities which describe what optional CMIS functionality the repository supports.

Using the CMIS service endpoint in an HTTP Get call will return the endpoint's CMIS service document which describes the CMIS functionality it supports.

Each CMIS object has an ID, type, and a set of properties for that type. There are four base types for a CMIS object :-

Document
: An item of content. The document may have a content stream, which is the actual file associated with the document. A content stream exists only as part of its containing document object. A content stream has a mimetype associated with it. A document object may contain one or more renditions, which are alternate views of the content. Documents objects are the only objects that are versionable. Each version of a document has its own object ID. All the versions of a document make up a version series and share a version series ID. You can create, read, update and delete documents using CMIS methods.

Folder
: A container used to organize the document objects. A repository has one root folder. All other folder objects have one parent folder. A folder has a folder path representing its place in the repository's folder hierarchy.

Relationship
: A relationship between a source object and a target object. Creating, changing and deleting relationships does not change the source or target objects themselves.

Policy
: An optional repository-specific object that can be applied to controllable objects. The behavior of policies are not modeled by the CMIS specification. A policy object may be applied to multiple controllable objects and a controllable object may have multiple policies applied to it. A policy object can not be deleted if it is currently applied to one or more controllable objects.

What does a request look like?

You call a method on the CMIS AtomPub REST API by issuing an authenticated HTTP request with a URL.

The four HTTP methods are used to support the traditional Create, Read, Update, and Delete (CRUD) operations of content management:-

POST
is used to create a new entities

GET
is used to retrieve information

PUT
is used to update a single entity

DELETE
is used to delete a single entity

Request URL format

Each request is a URL with a specific format.

This is an example of a request URL

```
   https://api.alfresco.com/yourcomopany.com/public/cmis/versions/1.0/atom/
content?id=a99ae2db-0e40-4fb6-bf67-3f331a358cfc
```

Each request URL is made up of the following elements:-

1. The protocol, which will always be `https`
2. The hostname which will always be `api.alfresco.com`
3. Your network id, which in this case is `yourcompany.com`
4. The API you want to call. In this case it is the public Alfresco CMIS API identified as `/public/cmis`.
5. `/versions/n`. This specifies the version of the CMIS API you are using. Currently `n` will always be `1.0`.
6. The CMIS binding. Currently the Alfresco cloud supports the `atom` binding.
7. The CMIS method itself. In this case the request is to get the content of a CMIS document with a specific id.

Getting the service document

The capabilities available to your application from the Alfresco Cloud are described in a an AtomPub document returned when calling the base URL. The service document contains information on the repository, the CMIS methods that can be called on it, and the parameters for those methods.

Getting the service document for all networks

To retrieve the service document for the current authenticated user's networks in the Alfresco cloud, use the HTTP GET method with this URL:

```
   https://api.alfresco.com/cmis/versions/1.0/atom/
```

The response body is an AtomPub XML document which describes the CMIS capabilities in a standard way. See the CMIS specification for more details.

Getting the service document for a specific network

To retrieve the service document for a specific network that the current authenticated user is a member of, use the HTTP GET method with a URL that specifies the network. For example this URL returns the service document for the `yourcompany.com` network.

```
https://api.alfresco.com/yourcompany.com/public/cmis/versions/1.0/atom
```

The response body is an AtomPub XML document which describes the CMIS capabilities in a standard way. See the CMIS specification for more details.

Getting information on a node

You can get information on a specific node in the repository by using its `id`. The resulting AtomPub XML document describes the node. You can tailor the information returned by providing HTML parameters.

URL format

Here is an example of a URL to retrieve information on a specific node:

```
https://api.alfresco.com/yourcompany.com/public/cmis/versions/1.0/atom/id?
id=5dba1525-44a6-45ed-a42e-4a155a3f0539
```

The response body is an AtomPub XML document which describes the CMIS capabilities in a standard way. See the CMIS specification for more details.

Parameters

You can add the following optional HTTP parameters to the URL:

Parameter	Optional?	Default value	Description
filter	Yes	Repository specific	A comma-separated list of query names that defines which properties must be returned by the repository.
includeAllowableActions	Yes	false	A boolean value. A value of `true` specifies that the repository must return the allowable actions for the node.
includeRelationships	Yes	IncludeRelationships.NONE	The relationships in which the node participates that must be returned in the response.
renditionFilter	Yes	cmis:none	A filter describing the set of renditions that must be returned in the response.
includePolicyIds	Yes	false	A boolean value. A value of `true` specifies the repository must return the policy ids for the node.
includeAcl	Yes	false	A boolean value. A value of `true` specifies the repository must return the Access Control List (ACL) for the node.

Getting the children of a node

You can get the children of a specific node in the repository by using its `id`. The resulting AtomPub XML document describes children of the node. You can tailor the information returned by providing HTML parameters. You can use this method to navigate a folder tree in the repository.

URL format

Here is an example of a URL to retrieve information on a specific node:

```
https://api.alfresco.com/yourcompany.com/public/cmis/versions/1.0/atom/
children?id=5dba1525-44a6-45ed-a42e-4a1a1a3f0539
```

The response body is an AtomPub XML document which describes the child nodes in a standard way. See the CMIS specification for more details.

Parameters

You can add the following optional HTTP parameters to the URL:

Parameter	Optional?	Default value	Description
filter	Yes	Repository specific	A comma-separated list of query names that defines which properties must be returned by the repository.
orderBy	Yes	Repository specific	A comma-separated list of query names that defines the order of the results set. Each query name in the list must be followed by the string `ASC` or `DESC` to specify the direction of the order, ascending or descending.
includeAllowableActions	Yes	false	A boolean value. A value of `true` specifies that the repository must return the allowable actions for each node.
includeRelationships	Yes	IncludeRelationships.NONE	The relationships in which each node participates that must be returned in the response.
renditionFilter	Yes	cmis:none	A filter describing the set of renditions that must be returned in the response.
includePathSegment	Yes	false	A boolean value. A value of `true` returns a path segment in the response for each child object that can be used to construct that object's path.
maxItems	Yes	Repository specific	The maximum number of items to return in the response.
skipCount	Yes	0	The number of objects to skip over before returning any results.

Getting the contents of a document

You can get the contents of a specific document in the repository by using its `id`. The format of the URI and the parameters that you can use are detailed in the service document.

URL format

Here is an example of a URL to retrieve the contents of a specific document:

```
https://api.alfresco.com/yourcompany.com/public/cmis/versions/1.0/atom/content?
id=824ba7cd-dcee-4908-8917-7b6ac0611c97
```

The response body is the content of the document. The format is specific to the type of content, so for example, getting the contents of a text document returns a text response body.

Updating the contents of a document

You can replace the contents of a specific document in the repository by using its `id`. The format of the URl and the parameters that you can use are detailed in the service document.

URL format

Here is an example of a URL to retrieve the contents of a specific document:

```
https://api.alfresco.com/yourcompany.com/public/cmis/versions/1.0/atom/content?
id=824ba7cd-dcee-4908-8917-7b6ac0611c97
```

Request Header

The request Content-Type must be of the same mime-type as the target document. In this example, we are updating a plain text document.

```
Content-Type: text/plain; charset=utf-8
```

Request body

The request body is the new content of the document.

```
Some updated text.
```

Response

If the request is successful an HTTP CREATED response (status 201) is returned.

Alfresco REST API

The API gives your application access to Alfresco Networks, Sites, Containers, Comments, Ratings, and tags. Unlike CMIS, response and request bodies are all specified with simple JSON.

Getting Started

To get you started with the API, this section explains the format of the URL you will use, and what to expect in responses.

What is an entity?

The generic term used in the API for any object in an Alfresco repository is entity. An entity is of a specific entity type, and has a unique entity id.

The Alfresco REST API operates on the following entity types:

sites
An Alfresco site is a project area where you can share content and collaborate with other site members.

containers
A container is a folder or space in a site.

members
Members are the people who collaborate on a site.

people
People are the users of Alfresco. A person entity describes the user as they are known to Alfresco.

favoriteSites
The sites that a person has marked as favorite in Alfresco.

preferences
A person's preferences in Alfresco.

networks
A network is the group of users and sites that belong to an organization. Networks are organized by email domain. When a user signs up for an Alfresco account , their email domain becomes their Home Network.

activities
Activities describe any past activity in a site, for example creating an item of content, commenting on a node, liking an item of content.

nodes
A node is an overall term for an item of content or a folder.

comments
A person can comment on folders and individual items to give other users information or notes specific to that content.

tags
Any item of Alfresco content can be tagged.

ratings
A person can rate an item of content by liking it. They can also remove their like of an item of content.

favorites
A favorite describes an Alfresco entity that a person has marked as a favorite.

site membership request
A site membership request describes a request for a person to join a site in Alfresco.

A logical group of entities is termed a collection.

What does a request look like?

You call a method on the API by issuing an authenticated HTTP request with a URL.

The four HTTP methods are used in the following ways:-

POST
is used to create a new entity in a collection of entities

GET
is used to retrieve information on a single entity or to retrieve a list of entities

PUT
is used to update a single entity

DELETE
is used to delete a single entity

Request URL format

Each request is a URL with a specific format.

This is an example of a request URL

```
https://api.alfresco.com/example.com/public/alfresco/versions/1/sites
```

Each request URL is made up of the following elements:-

1. The protocol, which will always be `https`
2. The hostname which will always be `api.alfresco.com`
3. Your network id, which in this case is `yourcompany.com`
4. The API you want to call. In this case it is the Alfresco REST API identified as `/public/alfresco`.
5. `/versions/n`. This specifies the version of the API you are using. Currently `n` will always be `1`.
6. The API method itself. In this case the request is for all instances of the entity type `sites`.

API method format

The method itself consists of at least one entity type, or an entity type and an entity id, or concatenations of entity type and id pairs, optionally followed by HTTP parameters that filter the results.

For example the following API method will return a list of all site entities:

```
sites
```

The entity type can be followed by an entity id, so for example the following API method will return on the site entity with the id `fred-bloggs-yourcompany-com`.

```
sites/fred-bloggs-yourcompany-com
```

Entity types and ids can be concatenated, so for example the following API method will get site membership information for a specific person from a specific site

```
sites/fred-bloggs-yourcompany-com/members/fred.bloggs@yourcompany.com
```

Specifying the current user

When making an Alfresco REST API call, your application may not know the userId of the currently authenticated user. You can use the string `-me-` to represent that user in request URLs, and PUT and POST request bodies.

For example, assuming the currently authenticated user is `fred.bloggs@yourcompany.com`the following URL will return a list of site memberships for the currently authenticated user:

```
https://api.alfresco.com/yourcompany.com/public/alfresco/versions/1/
people/fred.bloggs@yourcompany.com/sites
```

Using the current user `-me-`, the following URL will return the same list of site memberships:

```
https://api.alfresco.com/yourcompany.com/public/alfresco/versions/1/
people/-me-/sites
```

HTTP Parameters

The API provides several HTTP parameters that you can append to any API method URL to filter the returned results. The parameters are optional and can be use in combination with each other. There are also parameters that are used in a specific API method. You will find those documented in the API reference. Parameters listed here are applicable to any API method.

Pagination

As a developer, the REST API gives you control on how much of a returned collection you want to receive.

The collection returned by a simple request can contain a large number of entities. You can control the size of the list using pagination. So for example if a node with an id of `e8680e58-0701-4b64-950d-66cce277fbc7` has 100 comments the following request will return a list of 100 entities:

```
nodes/e8680e58-0701-4b64-950d-66cce277fbc7/comments
```

You can get just the first 10 items using the maxItems parameter:

```
nodes/e8680e58-0701-4b64-950d-66cce277fbc7/comments?maxItems=10
```

You can then get the second page of 10 items using the skipCount parameter:

```
nodes/e8680e58-0701-4b64-950d-66cce277fbc7/comments?
maxItems=10&skipCount=10
```

A returned list object will always contain a pagination object which has the following properties:

skipCount
An integer describing how many entities exist in the collection before those included in this list.

maxItems
The maxItems parameter used to generate this list, or if there was no maxItems parameter the default value, 10.

count
The number of objects in the entries array.

hasMoreItems
A boolean value which is true if there are more entities in the collection beyond those in this response. A true value means request with a larger value for the skipCount or the maxItems parameter will return more entities.

totalItems
An integer describing the total number of entities in the collection. The API may not be able to determine this value, in which case this property will not be present.

Filtering properties

You may only be interested in a subset of properties in a returned entity. You can use the properties parameter to restrict the returned properties.

The properties parameter is a comma-separated list of property names:-

```
properties=property1,property2...
```

For example if you invoked the following API method using the HTTP GET method:-

```
        sites
```

Alfresco would return a list of site objects each with four properties; `id`, `title`, `visibility`, `description`. Your application may only interested in say two properties, `title` and `description`. You can filter the returned site objects like this :-

```
        sites?properties=title,description
```

The collection returned will look like this:-

```
{
  "list" : {
    "pagination" : {
      "count" : 2,
      "hasMoreItems" : false,
      "totalItems" : 2,
      "skipCount" : 0,
      "maxItems" : 100
    },
    "entries" : [ {
      "entry" : {
        "title" : "Test Site",
        "description" : "A site for testing"
      }
    }, {
      "entry" : {
        "title" : "Fred Bloggs's Home",
        "description" : "Fred Blogs's private home site."
      }
    } ]
  }
}
```

Each entry in the list is a site object filtered to include just the `title` and `description` properties.

Including relations

Use the `relations` parameter to include one or more related entities in a single response.

The entity types in Alfresco are organized in a tree structure. So for example, the `sites` entity has two children, `containers` and `members`. You can reduce network traffic by using the `relations` parameter to include one or more child entities in a single response. The parameter is a comma separated list of entity types

```
relations=entity1,entity2,...
```

If you you invoked the following API method using the HTTP GET method:-

```
        sites?relations=containers,members
```

Alfresco returns a list of site objects, and retrieves the child container and member objects for each site in the returned collection, and returns them in a peer object of the entry object containing the site. Here is an example of the returned JSON:-

```
{
  "list" : {
    "pagination" : {
      "count" : 2,
      "hasMoreItems" : false,
      "totalItems" : 2,
      "skipCount" : 0,
      "maxItems" : 100
    },
    "entries" : [ {
      "entry" : {
        "id" : "test",
        "title" : "test",
        "visibility" : "PUBLIC"
      },
      "relations" : {
        "containers" : {
          "list" : {
            "pagination" : {
              "count" : 1,
              "hasMoreItems" : false,
              "totalItems" : 1,
              "skipCount" : 0,
              "maxItems" : 100
            },
            "entries" : [ {
              "entry" : {
                "id" : "5b0d84c8-0749-4fee-bd4f-9134d6990e5b",
                "folderId" : "documentLibrary"
              }
            } ]
          }
        },
        "members" : {
          "list" : {
            "pagination" : {
              "count" : 2,
              "hasMoreItems" : false,
              "skipCount" : 0,
              "maxItems" : 100
            },
            "entries" : [ {
              "entry" : {
                "id" : "fred-bloggs@yourcompany.com",
                "person" : {
                  "enabled" : true,
                  "lastName" : Bloggs",
                  "id" : "fred.bloggs@yourcompany.com",
                  "email" : "fred.bloggs@yourcompany.com",
                  "company" : {
                  },
                  "firstName" : "Fred"
```

```
            },
            "role" : "SiteManager"
          }
        }, {
          "entry" : {
            "id" : "joe-bloggs@yourcompany.com",
            "person" : {
              "enabled" : true,
              "lastName" : "Bloggs",
              "id" : "joe.bloggs@yourcompany.com",
              "email" : "joe.bloggs@yourcompany.com",
              "company" : {
              },
              "firstName" : "Joe"
            },
            "role" : "SiteConsumer"
          }
        } ]
      }
    }
  }
}, {
  "entry" : {
    "id" : "fred-bloggs-yourcompany-com",
    "title" : "Fred Bloggs's Home",
    "visibility" : "PRIVATE",
    "description" : "Fred Bloggs's private home site."
  },
  "relations" : {
    "containers" : {
      "list" : {
        "pagination" : {
          "count" : 1,
          "hasMoreItems" : false,
          "totalItems" : 1,
          "skipCount" : 0,
          "maxItems" : 100
        },
        "entries" : [ {
          "entry" : {
            "id" : "289f9030-eef6-421f-bdb6-1e6d2da165b6",
            "folderId" : "documentLibrary"
          }
        } ]
      }
    },
    "members" : {
      "list" : {
        "pagination" : {
          "count" : 1,
          "hasMoreItems" : false,
          "skipCount" : 0,
          "maxItems" : 100
        },
        "entries" : [ {
          "entry" : {
            "id" : "fred.bloggs@alfresco.com",
            "person" : {
              "enabled" : true,
              "lastName" : "Raff",
              "location" : "Somewhere",
              "avatarId" : "85d45e64-eb02-44e1-b989-dbf571ab0704",
              "instantMessageId" : "fredb",
              "googleId" : "fredb@gmail.com",
              "id" : "fred.bloggs@alfresco.com",
              "skypeId" : "fredb",
              "email" : "fred.bloggs@alfresco.com",
              "description" : "Been with company for n years",
```

```
                  "company" : {
                    "organization" : "Your Company",
                    "address1" : "Some place",
                    "address2" : "Somewhere",
                    "postcode" : "Z99 9Z9",
                    "telephone" : "01234 123456",
                    "fax" : "01234 123457",
                    "email" : "info@yourcompany.com"
                  },
                  "firstName" : "Fred",
                  "telephone" : "01234 567890",
                  "jobTitle" : "VP of something",
                  "mobile" : "07777 567890"
                },
                "role" : "SiteManager"
              }
            } ]
          }
        }
      }
    } ]
  }
}
```

Filtering relations

You can include just those properties in which you are interested in objects returned in a relations object.

You can include an optional comma-separated list of properties in any entity specified in a relations parameter to filter the resulting objects to include just those properties you are interested in.

```
relations=relation1(property1,property2,...),relation2(propertya,propertyb,...),relatio
```

For example, if you wanted to retrieve a list of all sites, their top-level containers, and their members, but you only needed the id of the sites, the folderId of the containers, and the id and role of the members, you would invoke the following API method using the HTTP GET method:-

```
        sites?properties=id&relations=containers(folderId),members(id,role)
```

Alfresco returns a list of filtered site objects, with the child container and member objects filtered to contain just the requested properties. Here is an example of the returned JSON:-

```
{
  "list" : {
    "pagination" : {
      "count" : 2,
      "hasMoreItems" : false,
      "totalItems" : 2,
      "skipCount" : 0,
      "maxItems" : 100
    },
    "entries" : [ {
      "entry" : {
        "id" : "test"
      },
      "relations" : {
        "containers" : {
          "list" : {
            "pagination" : {
              "count" : 1,
              "hasMoreItems" : false,
              "totalItems" : 1,
              "skipCount" : 0,
              "maxItems" : 100
            },
```

```
          "entries" : [ {
            "entry" : {
              "folderId" : "documentLibrary"
            }
          } ]
        }
      },
      "members" : {
        "list" : {
          "pagination" : {
            "count" : 2,
            "hasMoreItems" : false,
            "skipCount" : 0,
            "maxItems" : 100
          },
          "entries" : [ {
            "entry" : {
              "id" : "fred-bloggs@yourcompany.com"
              "role" : "SiteManager"
            }
          }, {
            "entry" : {
              "id" : "joe-bloggs@yourcompany.com"
              "role" : "SiteConsumer"
            }
          } ]
        }
      }
    }
  }, {
    "entry" : {
      "id" : "fred-bloggs-yourcompany-com"
    },
    "relations" : {
      "containers" : {
        "list" : {
          "pagination" : {
            "count" : 1,
            "hasMoreItems" : false,
            "totalItems" : 1,
            "skipCount" : 0,
            "maxItems" : 100
          },
          "entries" : [ {
            "entry" : {
              "folderId" : "documentLibrary"
            }
          } ]
        }
      },
      "members" : {
        "list" : {
          "pagination" : {
            "count" : 1,
            "hasMoreItems" : false,
            "skipCount" : 0,
            "maxItems" : 100
          },
          "entries" : [ {
            "entry" : {
              "id" : "fred.bloggs@alfresco.com"
              "role" : "SiteManager"
            }
          } ]
        }
      }
    }
  } ]
```

```
  }
}
```

What does a response look like?

All responses are JSON objects. The format of the response object depends on the request. The object may contain an entry object, an entry and a relations object, a list object, or an error object. Note that if a property or an entire object has no value, then it is not returned in the parent object.

Entry object

An API call which returns information about a single entity will return in an entry object. Here is an example response from a request for information on a site with a specific site-id:-

```
{
   "entry":{
      "title":"Fred Blogg's Home",
      "description":"Fred Blogg's private home site.",
      "visibility":"PRIVATE",
      "id":"fred-bloggs-yourcompany-com"
   }
}
```

Note that the entry object's properties are variable and depend on the API call made.

Relations object

If an API method specifies the Including relations on page 17 parameter, then any included children will be returned in a relations object. Here is an example of a relations object :-

```
    "relations" : {
      "containers" : {
        "list" : {
          "pagination" : {
            "count" : 1,
            "hasMoreItems" : false,
            "totalItems" : 1,
            "skipCount" : 0,
            "maxItems" : 100
          },
          "entries" : [ {
            "entry" : {
              "id" : "b9f8c112-66b9-4733-a77d-46e61c395706",
              "folderId" : "documentLibrary"
            }
          } ]
        }
      }
    }
```

List object

An API call which returns information about a several entities will return in a list object. A list will always have two properties, `pagination` and `entries` . The pagination object is described in Pagination on page 16 . The entries object is an array of entry objects. Here is an example response from a request for information on all sites:-

```
{
   "list":{
      "pagination":{
         "count":1,
         "hasMoreItems":false,
         "totalItems":1,
```

```
         "skipCount":0,
         "maxItems":10
      },
      "entries":[
         {
            "entry":{
               "title":"Fred Blogg's Home",
               "description":"Fred Blogg's private home site.",
               "visibility":"PRIVATE",
               "id":"fred-bloggs-yourcompany-com"
            }
         }
      ]
   }
}
```

Error object

An API call which fails for some reason will return an error object containing these properties:-

errorKey
A unique string identifier

statusCode
The HTTP status code for the type of error. The same code is returned in the HTTP response.

briefSummary
description of the cause of the error

descriptionUrl
A URL to a detailed description of the error

stackTrace
If an exception was thrown, this contains the Java stack trace as a string

additionalState
This optional property if it is present contains a free-form JSON object with additional information on the state of the server and/or the request

Here is an example of an error object from a request for a specific site-id that does not exist on the server:

```
{
 "error" : {
   "statusCode" : 404,
   "briefSummary" : "07220488 The entity with id: frank-bloggs-yourcompany-com
was not found",
   "stackTrace" :
"[org.alfresco.rest.api.impl.SitesImpl.validateSite(SitesImpl.java:111),
org.alfresco.rest.api.impl.SitesImpl.getSite(SitesImpl.java:137), ... ,java.lang.Threa
   "descriptionURL" : "http://someError?id=null"
 }
}
```

Note that the stack trace has been truncated for this example.

Date and Time format

Dates in the JSON response object are encoded as character strings using the extended format defined by ISO 8601:2004 They are always in UTC.

You can find the ISO 8601:2004 here. Here's an example of what to expect in a date/time string in a JSON response:-

```
"createdAt" : "2012-07-20T21:46:09.659+0000"
```

Using HTTP OPTIONS to get entity metadata

The Alfresco REST API supports the use of the HTTP OPTIONS method to retrieve structured information on the methods available on an entity and its relations.

Method

For example, to get information on the `nodes` entity, the methods you can use on it, its children (or relations), and the methods you can use on those, you can invoke the following API method using the HTTP OPTIONS method:-

```
nodes
```

Example request URL

```
https://api.alfresco.com/yourcompany.com/public/alfresco/versions/1/nodes
```

Response

- If the request is successful an HTTP OK is returned (status 200).

Example response body

```
{
  "list" : {
    "pagination" : {
      "count" : 4,
      "hasMoreItems" : false,
      "totalItems" : 4,
      "skipCount" : 0,
      "maxItems" : 100
    },
    "entries" : [ {
      "entry" : {
        "metaData" : {
          "uniqueId" : "/nodes",
          "type" : "ENTITY"
        }
      }
    }, {
      "entry" : {
        "metaData" : {
          "uniqueId" : "/nodes/{entityId}/tags",
          "type" : "RELATIONSHIP",
          "operations" : [ {
            "httpMethod" : "POST",
            "title" : "Add the tag to the node with id 'nodeId'.",
            "parameters" : [ {
              "name" : "entityId",
              "required" : true,
              "title" : "The unique id of the entity being addressed",
              "description" : "The unique id must be a String. It is returned
 as an 'id' from the entity",
              "dataType" : "java.lang.String",
              "allowMultiple" : false,
              "paramType" : "TEMPLATE"
            }, {
              "name" : "TAG",
              "required" : true,
              "title" : "The entity",
              "description" : "What shall we say?",
              "dataType" : "org.alfresco.rest.api.model.Tag",
              "allowMultiple" : false,
              "paramType" : "OBJECT"
            } ]
          }, {
```

```
          "httpMethod" : "GET",
          "title" : "A paged list of tags on the node 'nodeId'.",
          "parameters" : [ {
            "name" : "entityId",
            "required" : true,
            "title" : "The unique id of the entity being addressed",
            "description" : "The unique id must be a String. It is returned
as an 'id' from the entity",
            "dataType" : "java.lang.String",
            "allowMultiple" : false,
            "paramType" : "TEMPLATE"
          } ]
        }, {
          "httpMethod" : "DELETE",
          "title" : "Remove the tag from the node with id 'nodeId'.",
          "parameters" : [ {
            "name" : "entityId",
            "required" : true,
            "title" : "The unique id of the entity being addressed",
            "description" : "The unique id must be a String. It is returned
as an 'id' from the entity",
            "dataType" : "java.lang.String",
            "allowMultiple" : false,
            "paramType" : "TEMPLATE"
          } ]
        } ],
        "parentResource" : "/nodes"
      }
    }
  }, {
    "entry" : {
      "metaData" : {
        "uniqueId" : "/nodes/{entityId}/ratings",
        "type" : "RELATIONSHIP",
        "operations" : [ {
          "httpMethod" : "POST",
          "title" : "Apply a rating for node 'nodeId'.",
          "parameters" : [ {
            "name" : "entityId",
            "required" : true,
            "title" : "The unique id of the entity being addressed",
            "description" : "The unique id must be a String. It is returned
as an 'id' from the entity",
            "dataType" : "java.lang.String",
            "allowMultiple" : false,
            "paramType" : "TEMPLATE"
          }, {
            "name" : "NODERATING",
            "required" : true,
            "title" : "The entity",
            "description" : "What shall we say?",
            "dataType" : "org.alfresco.rest.api.model.NodeRating",
            "allowMultiple" : false,
            "paramType" : "OBJECT"
          } ]
        }, {
          "httpMethod" : "GET",
          "title" : "A paged list of ratings for node 'nodeId'.",
          "parameters" : [ {
            "name" : "entityId",
            "required" : true,
            "title" : "The unique id of the entity being addressed",
            "description" : "The unique id must be a String. It is returned
as an 'id' from the entity",
            "dataType" : "java.lang.String",
            "allowMultiple" : false,
            "paramType" : "TEMPLATE"
          } ]
```

```
        }, {
          "httpMethod" : "GET",
          "title" : "Get the rating with id 'ratingSchemeId' for node
'nodeId'.",
          "parameters" : [ {
            "name" : "entityId",
            "required" : true,
            "title" : "The unique id of the entity being addressed",
            "description" : "The unique id must be a String. It is returned
as an 'id' from the entity",
            "dataType" : "java.lang.String",
            "allowMultiple" : false,
            "paramType" : "TEMPLATE"
          }, {
            "name" : "relationshipId",
            "required" : true,
            "title" : "The unique id of the entity relationship being
addressed",
            "description" : "The unique id must be a String. It is only valid
in the scope of the relationship",
            "dataType" : "java.lang.String",
            "allowMultiple" : false,
            "paramType" : "TEMPLATE"
          } ]
        }, {
          "httpMethod" : "DELETE",
          "title" : "Missing @WebApiDescription annotation",
          "description" : "This method should be annotated with
@WebApiDescription",
          "parameters" : [ {
            "name" : "entityId",
            "required" : true,
            "title" : "The unique id of the entity being addressed",
            "description" : "The unique id must be a String. It is returned
as an 'id' from the entity",
            "dataType" : "java.lang.String",
            "allowMultiple" : false,
            "paramType" : "TEMPLATE"
          } ]
        } ],
        "parentResource" : "/nodes"
      }
    }
  }, {
    "entry" : {
      "metaData" : {
        "uniqueId" : "/nodes/{entityId}/comments",
        "type" : "RELATIONSHIP",
        "operations" : [ {
          "httpMethod" : "POST",
          "title" : "Create a comment for the node 'nodeId'.",
          "parameters" : [ {
            "name" : "entityId",
            "required" : true,
            "title" : "The unique id of the entity being addressed",
            "description" : "The unique id must be a String. It is returned
as an 'id' from the entity",
            "dataType" : "java.lang.String",
            "allowMultiple" : false,
            "paramType" : "TEMPLATE"
          }, {
            "name" : "COMMENT",
            "required" : true,
            "title" : "The entity",
            "description" : "What shall we say?",
            "dataType" : "org.alfresco.rest.api.model.Comment",
            "allowMultiple" : false,
            "paramType" : "OBJECT"
```

```
            } ]
          }, {
            "httpMethod" : "GET",
            "title" : "Returns a paged list of comments for the document/folder
 identified by nodeId, sorted chronologically with the newest first.",
            "parameters" : [ {
              "name" : "entityId",
              "required" : true,
              "title" : "The unique id of the entity being addressed",
              "description" : "The unique id must be a String. It is returned
 as an 'id' from the entity",
              "dataType" : "java.lang.String",
              "allowMultiple" : false,
              "paramType" : "TEMPLATE"
            } ]
          }, {
            "httpMethod" : "PUT",
            "title" : "Updates the comment with the given id.",
            "parameters" : [ {
              "name" : "entityId",
              "required" : true,
              "title" : "The unique id of the entity being addressed",
              "description" : "The unique id must be a String. It is returned
 as an 'id' from the entity",
              "dataType" : "java.lang.String",
              "allowMultiple" : false,
              "paramType" : "TEMPLATE"
            }, {
              "name" : "relationshipId",
              "required" : true,
              "title" : "The unique id of the entity relationship being
 addressed",
              "description" : "The unique id must be a String. It is only valid
 in the scope of the relationship",
              "dataType" : "java.lang.String",
              "allowMultiple" : false,
              "paramType" : "TEMPLATE"
            }, {
              "name" : "COMMENT",
              "required" : true,
              "title" : "The entity",
              "description" : "What shall we say?",
              "dataType" : "org.alfresco.rest.api.model.Comment",
              "allowMultiple" : false,
              "paramType" : "OBJECT"
            } ]
          }, {
            "httpMethod" : "DELETE",
            "title" : "Delete the comment with the given commentNodeId.",
            "parameters" : [ {
              "name" : "entityId",
              "required" : true,
              "title" : "The unique id of the entity being addressed",
              "description" : "The unique id must be a String. It is returned
 as an 'id' from the entity",
              "dataType" : "java.lang.String",
              "allowMultiple" : false,
              "paramType" : "TEMPLATE"
            } ]
          } ],
          "parentResource" : "/nodes"
        }
      }
    } ]
  }
}
```

API Reference

This reference material has a description of each of the Alfresco entities operated on by the REST API, and of the methods that are available for each of those entities.

Networks

A network is the group of users and sites that belong to an organization. Networks are organized by email domain. When a user signs up for an Alfresco account , their email domain becomes their Home Network.

Network object

Property	Type	JSON Type	Description
id	id	string	This network's unique id
type	enumerated string	string	This network's type
isEnabled	boolean	boolean	Is this network active?
createdAt	Date Time	String	The date time this network was created
quotas	array	array	Limits and usage of each quota. A network will have quotas for File space, the number of sites in the network, the number of people in the network, and the number of network administrators.
paidNetwork	boolean	boolean	Is this a paid network?
subscriptionLevel	enumerated string	string	The type of subscription for this network. Possible values are `Free`, `Standard`, and `Enterprise`

Example of a network object

```
"entry" : {
  "id" : "yourcompany.com",
  "createdAt" : "2012-06-07T10:22:28.000+0000",
  "quotas" : [ {
    "limit" : 52428800,
    "id" : "fileUploadQuota"
  }, {
    "limit" : 5368709120,
    "usage" : 149102356,
    "id" : "fileQuota"
  }, {
    "limit" : -1,
    "usage" : 29,
    "id" : "siteCountQuota"
  }, {
    "limit" : -1,
    "usage" : 33,
    "id" : "personCountQuota"
  }, {
    "limit" : -1,
    "usage" : 15,
    "id" : "personInternalOnlyCountQuota"
  }, {
    "limit" : 0,
    "usage" : 0,
    "id" : "personNetworkAdminCountQuota"
```

```
    } ],
    "paidNetwork" : false,
    "isEnabled" : true,
    "subscriptionLevel" : "Free"
  }
```

List order

Lists of these entities are returned ordered by ascending `id`.

Get a specific network

Method

Using the HTTP GET method:-

```
networks/<networkId>
```

Example request URL

```
https://api.alfresco.com/yourcompany.com/public/alfresco/versions/1/networks/
yourcompany.com
```

Response

- If the networkId does not exist in this network, an HTTP `Not Found` (status 404) is returned.
- If the request is successful an HTTP OK is returned (status 200).

Example response body

```
{
    "id" : "yourcompany.com",
    "createdAt" : "2012-06-07T10:22:28.000+0000",
    "quotas" : [ {
      "limit" : 52428800,
      "id" : "fileUploadQuota"
    }, {
      "limit" : 5368709120,
      "usage" : 149102356,
      "id" : "fileQuota"
    }, {
      "limit" : -1,
      "usage" : 29,
      "id" : "siteCountQuota"
    }, {
      "limit" : -1,
      "usage" : 33,
      "id" : "personCountQuota"
    }, {
      "limit" : -1,
      "usage" : 15,
      "id" : "personInternalOnlyCountQuota"
    }, {
      "limit" : 0,
      "usage" : 0,
      "id" : "personNetworkAdminCountQuota"
    } ],
    "paidNetwork" : false,
    "isEnabled" : true,
    "subscriptionLevel" : "Free"
  }
```

Get networks for the current authenticated person

Method

Using the HTTP GET method on the root URL.

Example request URL

```
https://api.alfresco.com/
```

Response

- If the request is successful an HTTP OK is returned (status 200).

Example response body

```
{
  "list" : {
    "pagination" : {
      "count" : 1,
      "hasMoreItems" : false,
      "totalItems" : 1,
      "skipCount" : 0,
      "maxItems" : 100
    },
    "entries" : [ {
      "entry" : {
          "id" : "yourcompany.com",
          "homeNetwork" : true,
          "createdAt" : "2012-06-07T10:22:28.000+0000",
          "quotas" : [ {
            "limit" : 52428800,
            "id" : "fileUploadQuota"
          }, {
            "limit" : 5368709120,
            "usage" : 149102356,
            "id" : "fileQuota"
          }, {
            "limit" : -1,
            "usage" : 29,
            "id" : "siteCountQuota"
          }, {
            "limit" : -1,
            "usage" : 33,
            "id" : "personCountQuota"
          }, {
            "limit" : -1,
            "usage" : 15,
            "id" : "personInternalOnlyCountQuota"
          }, {
            "limit" : 0,
            "usage" : 0,
            "id" : "personNetworkAdminCountQuota"
          } ],
          "paidNetwork" : false,
          "isEnabled" : true,
          "subscriptionLevel" : "Free"
        }
      }
    ]
  }
}
```

Sites

An Alfresco site is a project area where you can share content and collaborate with other site members. There are API calls for getting a list of sites, and for getting information on a single site.

Site object

Property	Type	JSON Type	Description
title	string	string	The site's name (used in the site's list and on the sites dashboard).
description	string	string	The description of the site
visibility	string	string	The visibility of the site, `PRIVATE`, `PUBLIC`, or `MODERATED`.
id	id	string	The site identifier. An opaque string which uniquely identifies this site.

Example of a site object

```
{
      "title":"Fred Bloggs's Home",
      "description":"Fred Bloggs's private home site.",
      "visibility":"PRIVATE",
      "id":"fred-bloggs-yourcompany-com"
}
```

Methods

Methods for Site objects.

Get a list of sites

Method

Using the HTTP GET method:-

```
sites
```

Example request URL

```
https://api.alfresco.com/yourcompany.com/public/alfresco/versions/1/sites
```

Response

- If the request is successful an HTTP OK is returned (status 200).

Example response body

```
{
   "list":{
      "pagination":{
         "count":1,
         "hasMoreItems":false,
         "totalItems":1,
         "skipCount":0,
         "maxItems":10
      },
      "entries":[
         {
            "entry":{
```

```
               "title":"Fred Blogg's Home",
               "description":"Fred Blogg's private home site.",
               "visibility":"PRIVATE",
               "id":"fred-bloggs-yourcompany-com"
            }
         }
      ]
   }
}
```

Get information for a site

Method

Using the HTTP GET method:-

```
sites/<siteId>
```

Example request URL

```
https://api.alfresco.com/yourcompany.com/public/alfresco/versions/1/sites/fred-bloggs-yourcompany-com
```

Response

- If the siteId does not exist in this network, an HTTP `Not Found` (status 404) is returned.
- If the request is successful an HTTP OK is returned (status 200).

Example response body

```
{
      "entry":{
         "title":"Fred Blogg's Home",
         "description":"Fred Blogg's private home site.",
         "visibility":"PRIVATE",
         "id":"fred-bloggs-yourcompany-com"
      }
}
```

Containers

A container is a folder or space in a site. There are API calls for getting a list of top-level containers in a site, and for getting a container by its `containerId`.

Container object

Property	Type	JSON Type	Description
folderId	string	string	The container's descriptive name.
id	id	string	The container identifier. An opaque string which uniquely identifies this container.

Example of a container object

```
{
     "folderId":"documentLibrary",
     "id":"7fb6c69b-f462-429a-a168-87762f660c65"
}
```

List order

Lists of these entities are returned ordered by ascending `folderId`.

Methods

Methods for Container objects.

Get a list of containers

Method

Using the HTTP GET method:-

```
sites/<siteId>/containers
```

Example request URL

```
https://api.alfresco.com/yourcompany.com/public/alfresco/versions/1/sites/fred-bloggs-yourcompany-com/containers
```

Response

- If the siteId does not exist in this network, an HTTP `Not Found` (status 404) is returned.
- If the request is successful an HTTP OK is returned (status 200).

Example response body

```
{
   "list":{
      "pagination":{
         "count":1,
         "hasMoreItems":false,
         "skipCount":0,
         "maxItems":100
      },
      "entries":[
         {
            "entry":{
               "folderId":"documentLibrary",
               "id":"7fb6c69b-f462-429a-a168-87762f660c65"
            }
         }
      ]
   }
}
```

Get information for a container

Method

Using the HTTP GET method:-

```
sites/<siteId>/containers/<containerId>
```

Example request URL

```
https://api.alfresco.com/yourcompany.com/public/alfresco/versions/1/sites/fred-bloggs-yourcompany-com/containers/7fb6c69b-f462-429a-a168-87762f660c65
```

Response

- If the siteId or containerId do not exist in this network, an HTTP `Not Found` (status 404) is returned.

- If the request is successful an HTTP OK is returned (status 200).

Example response body

```
{
      "entry":{
         "folderId":"documentLibrary",
         "id":"7fb6c69b-f462-429a-a168-87762f660c65"
      }
}
```

Members

Members are the people who collaborate on a site. There are API calls for getting a list of the members of the site, getting the site membership information for a person, adding a person to a site, and updating a person's site membership information.

Member object

Property	Type	JSON Type	Description
role	enumerated string	string	The member's role. Possible values are `SiteManager`, `SiteContributor`, and `SiteCollaborator`.
id	email id	string	The person's personId - the email address with which the person registered
People on page 42	person object	object	An embedded person object describing this member.

Example of a member object

```
{
      "role":"SiteManager",
      "id":"fred.bloggs@yourcompany.com",
      "person":{
            "enabled":true,
            "lastName":"Bloggs",
            "location":"Somewhere",
            "avatarId":"6be34757-5764-4a4b-a86c-f5f0878b9700",
            "instantMessageId":"fred",
            "googleId":"fred@google.com",
            "id":"fred.bloggs@yourcompany.com",
            "skypeId":"fredbloggs",
            "email":"fred.bloggs@yourcompany.com",
            "description":"a person",
            "company":{
               "organization":"alfresco",
               "address1":"somewhere",
               "postcode":"fff fff",
               "telephone":"01234 456789",
               "fax":"01234 456789",
               "email":"info@yourcompany.com"
            },
            "firstName":"Fred",
            "telephone":"01234 99229922",
            "jobTitle":"Chief Bottle Washer",
            "mobile":"07777 012345"
      }
  }
```

List order

Lists of these entities are returned ordered by ascending (`lastName`, `firstName`, `role`).

Methods

Methods for Member objects.

Get a list of members

Method

Using the HTTP GET method:-

```
sites/<siteId>/members
```

Example request URL

```
https://api.alfresco.com/yourcompany.com/public/alfresco/versions/1/sites/fred-bloggs-yourcompany-com/members
```

Response

- If the siteId does not exist in this network, an HTTP `Not Found` (status 404) is returned.
- If the request is successful an HTTP OK is returned (status 200).

Example response body


```
{
   "list":{
      "pagination":{
         "count":1,
         "hasMoreItems":false,
         "totalItems":-1,
         "skipCount":0,
         "maxItems":10
      },
      "entries":[
         {
            "entry":{
               "role":"SiteManager",
               "id":"fred.bloggs@yourcompany.com",
               "person":{
                     "enabled":true,
                     "lastName":"Bloggs",
                     "location":"Somewhere",
                     "avatarId":"6be34757-5764-4a4b-a86c-f5f0878b9700",
                     "instantMessageId":"fred",
                     "googleId":"fred@google.com",
                     "id":"fred.bloggs@yourcompany.com",
                     "skypeId":"fredbloggs",
                     "email":"fred.bloggs@yourcompany.com",
                     "description":"a person",
                     "company":{
                        "organization":"alfresco",
                        "address1":"somewhere",
                        "postcode":"fff fff",
                        "telephone":"01234 456789",
                        "fax":"01234 456789",
                        "email":"info@yourcompany.com"
                     },
                     "firstName":"Fred",
                     "telephone":"01234 99229922",
                     "jobTitle":"Chief Bottle Washer",
                     "mobile":"07777 012345"
               }
            }
         ]
   }
```

```
}
```

Get information for a member of a site

Method

Using the HTTP GET method:-

```
sites/<siteId>/members/<personId>
```

Example request URL

```
https://api.alfresco.com/yourcompany.com/public/alfresco/versions/1/sites/fred-bloggs-yourcompany-com/members/fred.bloggs@yourcompany.com
```

Response

- If the siteId or personId do not exist in this network, an HTTP `Not Found` (status 404) is returned.
- If the request is successful an HTTP OK is returned (status 200).

Example response body

```
{
   "entry":{
      "role":"SiteManager",
      "id":"fred.bloggs@yourcompany.com",
      "person":{
            "enabled":true,
            "lastName":"Bloggs",
            "location":"Somewhere",
            "avatarId":"6be34757-5764-4a4b-a86c-f5f0878b9700",
            "instantMessageId":"fred",
            "googleId":"fred@google.com",
            "id":"fred.bloggs@yourcompany.com",
            "skypeId":"fredbloggs",
            "email":"fred.bloggs@yourcompany.com",
            "description":"a person",
            "company":{
               "organization":"alfresco",
               "address1":"somewhere",
               "postcode":"fff fff",
               "telephone":"01234 456789",
               "fax":"01234 456789",
               "email":"info@yourcompany.com"
            },
            "firstName":"Fred",
            "telephone":"01234 99229922",
            "jobTitle":"Chief Bottle Washer",
            "mobile":"07777 012345"
      }
}
```

Create a member of a site

Method

Using the HTTP POST method:-

```
sites/<siteId>/members
```

Example request URL

```
https://api.alfresco.com/yourcompany.com/public/alfresco/versions/1/sites/fred-bloggs-yourcompany-com/members
```

POST body

Property	Type	JSON Type	Description
id	email id	string	The id of the person.
role	enumerated type	string	The role for this person. Possible values are `SiteConsumer` and `SiteManager`.

Example POST body

```
{
  'id': 'joe.bloggs@yourcompany.com',
  'role': 'SiteConsumer'
}
```

Response

- If the siteId or personId do not exist in this network, an HTTP `Not Found` (status 404) is returned.
- If the request is successful a HTTP CREATED response (status 201) is returned.

Example response body

```
{
   "entry":{
      "id":"fred.bloggs@yourcompany.com",
      "role":"SiteConsumer"
      }
}
```

Update a member of a site

Method

Using the HTTP PUT method:-

```
sites/<siteId>/members/<personId>
```

Example request URL

```
https://api.alfresco.com/yourcompany.com/public/alfresco/versions/1/sites/fred-bloggs-yourcompany-com/members/joe.bloggs@yourcompany.com
```

PUT body

Property	Type	JSON Type	Description
role	enumerated type	string	The new role for this person. Possible values are `SiteConsumer`, `SiteCollaborator`, and `SiteManager`.

Example PUT body

```
{
  'role': 'SiteManager'
}
```

Response

- If siteId, personId, or role do not exist an HTTP Not Found (status 404) is returned with a Not Found.
- If the personId supplied is not a member of the site an HTTP Bad Request (status 400) is returned with an Bad Request.

- If the request is successful an HTTP OK is returned (status 200).

Example response body

```
{
   "entry":{
      "id":"joe.bloggs@yourcompany.com",
      "role":"SiteManager"
      }
}
```

Remove a member of a site

Method

Using the HTTP DELETE method:-

```
sites/<siteId>/members/<personId>
```

A personID is always the email address that they registered with

Example request URL

```
https://api.alfresco.com/yourcompany.com/public/alfresco/versions/1/sites/fred-bloggs-yourcompany-com/members/fred.bloggs@yourcompany.com/
```

Response

- If the siteId or personId do not exist in this network, an HTTP `Not Found` (status 404) is returned.
- If the person is not a member of the site, an Bad Request (status 400) is returned.
- If the request is successful an HTTP No Content is returned (status 204), and the person's site membership is removed.

Site membership requests

A site membership request describes a request for a person to join a site in Alfresco. There are API calls for getting a list of a user's site membership requests, for joining a site, for modifying a request to join a site, and for deleting a site membership request.

Site membership request object

Property	Type	JSON Type	Description
id	string	string	The site id.
site	object	object	The target site.
message	string	string	An optional message from the requester explaining why access is being requested.
createdAt	date time	string	The time this site membership request was made.
modifiedAt	date time	string	The time this site membership request was modified.

Example of a site membership request object

```
{
   "entry": {
     "id" : "the-secret-site",
     "createdAt" : "2012-07-20T21:46:09.659+0000",
     "modifiedAt" : "2012-07-20T21:46:09.659+0000",
     "message" : "I need this access for national security reasons!",
     "site": {
```

```
        "id" : "the-secret-site",
        "guid" : "8ac18731-601b-4bb4-be1a-cd5d252cce3f",
        "title" : "The Company's Secret Site",
        "visibility" : "MODERATED",
        "description" : "The Company's Secret Site"
        }
    }
}
```

List order

Lists of these entities are returned ordered by ascending site `title`.

Get a list of site membership requests

Method

Using the HTTP GET method:-

```
people/>personId>/site-membership-requests
```

Example request URL

```
https://api.alfresco.com/yourcompany.com/public/alfresco/versions/1/people/
fred.bloggs@yourcompany.com/site-membership-requests
```

Response

- If the request is successful an HTTP `OK` is returned (status 200).
- If the **personId** does not exist in this network an HTTP `Not Found` is returned (status 404).
- If the current user does not have permission to access the site membership requests of the **personId**, an HTTP `Not Found` is returned (status 404).

Example response body

```
{
   "list" : {
      "pagination" : {
        "count" : 2,
        "hasMoreItems" : false,
        "skipCount" : 0,
        "maxItems" : 100
      },
      "entries": [  {
         "entry": {
           "id" : "fred-bloggs-yourcompany-com",
           "createdAt" : "2012-07-20T21:46:09.659+0000",
           "site": {
               "id" : "fred-bloggs-yourcompany-com",
               "guid" : "9de68812-720c-5ed4-de2d-fe4a364ddb2e",
               "title" : "Fred Bloggs's Site",
               "visibility" : "MODERATED",
               "description" : "Fred Bloggs's Site"
           }
         }
       },
       {
         "entry": {
           "id" : "the-secret-site",
           "createdAt" : "2012-08-20T21:46:09.659+0000",
           "modifiedAt" : "2012-09-20T21:46:09.672+0000",
           "message" : "I need this access for national security reasons!",
           "site": {
               "id" : "the-secret-site",
```

```
                "guid" : "8ac18731-601b-4bb4-be1a-cd5d252cce3f",
                "title" : "The Company's Secret Site",
                "visibility" : "MODERATED",
                "description" : "The Company's Secret Site"
             }
          }
       }
    ]
  }
}
```

Join a site

Method

Using the HTTP POST method:-

```
people/<personId>/site-membership-requests
```

Example request URL

```
https://api.alfresco.com/yourcompany.com/public/alfresco/versions/1/people/-me-/site-membership-requests
```

POST body

Property	Type	JSON Type	Description
id	string	string	The id of the site to be joined.
message	string	string	An optional message describing why site membership is being requested.

Example POST body

```
{
    "id" : "secret-site",
    "message" : "I need this access for national security reasons!"
}
```

Response

- If the request is successful an HTTP Created is returned (status 201).
- If the **personId** is already a member of **siteId** an HTTP `Bad Request` is returned (status 400).
- If an existing site membership request by **personId** for **siteId** exists, an HTTP `Bad Request` is returned (status 400).
- If the **personId** does not exist in this network an HTTP `Not Found` is returned (status 404).
- If the **siteId** does not exist in this network an HTTP `Not Found` is returned (status 404).
- If the **siteId** is private an HTTP `Not Found` is returned (status 404).
- If the current user does not match the **personId**, the user does not have permission to create this site membership request, and an HTTP `Not Found` is returned (status 404).

Example response body

```
{
  "entry" : {
     "targetGuid" : "8ac18731-601b-4bb4-be1a-cd5d252cce3f",
     "createdAt" : "2012-07-20T21:46:09.659+0000",
     "target": {
         "site" : {
```

```
            "id" : "foo",
            "guid" : "8ac18731-601b-4bb4-be1a-cd5d252cce3f",
            "title" : "The Foo Site",
            "visibility" : "PRIVATE",
            "description" : "The Foo Site",
            "role" : "SiteManager"
         }
      }
   }
}
```

Modifying a site membership request

Method

Using the HTTP PUT method:-

```
people/<personId>/site-membership-requests/<siteId>
```

Example request URL

```
https://api.alfresco.com/yourcompany.com/public/alfresco/versions/1/people/-
me-/site-membership-requests/secret-site
```

PUT body

Property	Type	JSON Type	Description
message	string	string	An optional message describing why site membership is being requested.

Example PUT body

```
{
    "message" : "I need this access for national security reasons!"
}
```

Response

- If the request is successful an HTTP `Created` is returned (status 200).
- If the **personId** does not exist in this network an HTTP `Not Found` is returned (status 404).
- If the **siteId** does not exist in this network an HTTP `Not Found` is returned (status 404).
- If the **siteId** does match a site membership request for **personId**, an HTTP `Not Found` is returned (status 404).
- If the current user does not match the **personId**, the user does not have permission to modfiy this site membership request, and an HTTP `Not Found` is returned (status 404).

Example response body

```
{
  "entry": {
    "id" : "the-secret-site",
    "createdAt" : "2012-07-20T21:46:09.659+0000",
    "modifiedAt" : "2012-08-20T21:46:09.659+0000",
    "message" : "I need this access for national security reasons!",
    "site": {
        "id" : "the-secret-site",
        "guid" : "8ac18731-601b-4bb4-be1a-cd5d252cce3f",
        "title" : "The Company's Secret Site",
        "visibility" : "MODERATED",
        "description" : "The Company's Secret Site"
```

```
    }
  }
}
```

Delete a site membership request

Method

Using the HTTP DELETE method:-

```
people/<personId>/site-membership-requests/<siteId>
```

Example request URL

```
https://api.alfresco.com/yourcompany.com/public/alfresco/versions/1/people/-me-/site-membership-requests/secret-site
```

Response

- If the request is successful the site membership request is removed and an HTTP `No Content` (status 204) is returned.
- If the personId does not exist in this network, an HTTP `Not Found` (status 404) is returned.
- If the specified siteId does not exist in this network, an HTTP `Not Found` (status 404) is returned.
- If the specified siteId does not does not match a site request for personId, an HTTP `Not Found` (status 404) is returned.
- If the current user does not match the personID, the user making the API call does not have sufficient permission to withdraw this site request, and an HTTP `Not Found` is returned (status 404).

People

People are the users of Alfresco. A person entity describes the user as they are known to Alfresco. There are API methods to get the sites a person is a member of, to get the the details of a person, their favorite sites, preferences, and networks they are a member of. Methods are also available to process activities related to a person.

Person object

Property	Type	JSON Type	Description
enabled	boolean	boolean	Is this person currently enabled?
lastName	string	string	the person's last name
location	string	string	The person's location or address
avatarId	id	string	The id of the person's avatar
instantMessageId	string	string	The person's instant message Id
googleId	string	string	The person's Google Id
id	email id	string	The person's personId - the email address with which the person registered
skypeId	string	string	The person's Skype Id
description	string	string	The person's description
company	company	object	An embedded company object describing the person's company
firstName	string	string	The person's first name
telephone	string	string	The person's telephone number
jobTitle	string	string	The person's job title
mobile	string	string	The person's mobile number

Example of a person object

```
{
  "entry" : {
    "enabled" : true,
    "lastName" : "Bloggs",
    "location" : "Somewhere",
    "avatarId" : "85d45e64-eb02-44e1-b989-dbf571ab0704",
    "instantMessageId" : "fredb",
    "googleId" : "fredb@gmail.com",
    "id" : "fred.bloggs@yourcompany.com",
    "skypeId" : "fredb",
    "email" : "fred.bloggs@yourcompany.com",
    "description" : "Been with company for n years",
    "company" : {
      "organization" : "Your Company",
      "address1" : "Some place",
      "address2" : "Somewhere",
      "postcode" : "Z99 9Z9",
      "telephone" : "01234 123456",
      "fax" : "01234 123457",
      "email" : "info@yourcompany.com"
    },
    "firstName" : "Fred",
    "telephone" : "01234 567890",
    "jobTitle" : "VP of something",
    "mobile" : "07777 567890"
  }
}
```

Methods

Methods for Person objects.

Get information about a person

Method

Using the HTTP GET method:-

```
people/<personId>
```

A personID is always the email address that they registered with

Example request URL

```
https://api.alfresco.com/yourcompany.com/public/alfresco/versions/1/people/
fred.bloggs@yourcompany.com
```

Response

- If the personId does not exist in this network, an HTTP `Not Found` (status 404) is returned.
- If the request is successful an HTTP OK is returned (status 200).

Example response body

```
{
   "entry":{
      "enabled":true,
      "lastName":"Bloggs",
      "location":"Someplace",
      "avatarId":"93df15f5-dee2-4bfe-8f1d-f0026d548f86",
      "instantMessageId":"fredb",
      "googleId":"fred.bloggs@gmail.com",
      "id":"fred.bloggs@yourcompany.com",
```

```
      "skypeId":"fredb",
      "description":"A generic person",
      "company":{
         "organization":"Alfresco",
         "address1":"address",
         "postcode":"post code",
         "telephone":"0123 456789",
         "fax":"0123 456789",
         "email":"enquiries@yourcompany.com"
      },
      "firstName":"Fred",
      "telephone":"0123 456777",
      "jobTitle":"VP of something",
      "mobile":"0777 456777"
   }
}
```

Note that the response object is an entry containing a People on page 42 entity with an embedded company entity.

Sites

An Alfresco site is a project area where you can share content and collaborate with other site members. There are API calls for getting a list of sites that a person is a member of, and for getting information about a person's membership of a single site.

For more information on the site object, see Sites on page 31.

Get a list of a person's site memberships

Method

Using the HTTP GET method:-

```
people/<personId>/sites
```

A personID is always the email address that they registered with.

Example request URL

```
https://api.alfresco.com/yourcompany.com/public/alfresco/versions/1/people/
fred.bloggs@yourcompany.com/sites
```

Response

- If the personId or the siteId does not exist in this network, an HTTP `Not Found` (status 404) is returned.
- If the request is successful an HTTP OK is returned (status 200).

Example response body

```
{
  "list" : {
    "pagination" : {
      "count" : 2,
      "hasMoreItems" : false,
      "skipCount" : 0,
      "maxItems" : 100
    },
    "entries" : [ {
      "entry" : {
        "site" : {
          "id" : "general-test-site",
```

```
          "title" : "General Test Site",
          "visibility" : "PRIVATE",
          "description" : "Test Site"
        },
        "id" : "general-test-site",
        "role" : "SiteCollaborator"
      }
    }, {
      "entry" : {
        "site" : {
          "id" : "fred-bloggs-yourcompany-com",
          "visibility" : "PRIVATE",
          "description" : "Fred Bloggs's private home site."
        },
        "id" : "fred-bloggs-yourcompany-com",
        "role" : "SiteManager"
      }
    } ]
  }
}
```

Note that each entry in the response list is a Members on page 34 entity with an embedded Sites on page 31 entity.

Get information about a person's site membership

Method

Using the HTTP GET method:-

```
people/<personId>/sites/<siteId>
```

A personID is always the email address that they registered with

Example request URL

```
https://api.alfresco.com/yourcompany.com/public/alfresco/versions/1/people/
fred.bloggs@yourcompany.com/sites/fred-bloggs-yourcompany-com
```

Response

- If the siteId or personId do not exist in this network, an HTTP `Not Found` (status 404) is returned.
- If the request is successful an HTTP OK is returned (status 200).

Example response body

```
{
  "entry" : {
    "site" : {
      "id" : "fred-bloggs-yourcompany-com",
      "title" : "Fred Bloggs's Home",
      "visibility" : "PRIVATE",
      "description" : "Fred Bloggs's private home site."
    },
    "id" : "fred-bloggs-yourcompany-com",
    "role" : "SiteManager"
  }
}
```

Note that the response object is an entry containing a Members on page 34 entity with an embedded Sites on page 31 entity.

Favorite sites

The sites that a person has marked as favorite in Alfresco.

Favorite-sites object

Property	Type	JSON Type	Description
id	email id	string	The person's personId. The email address the person registered with.
site	Sites on page 31	object	An embedded site object.

Example of a favorite-sites object

```
{
  "list" : {
    "pagination" : {
      "count" : 1,
      "hasMoreItems" : false,
      "totalItems" : 1,
      "skipCount" : 0,
      "maxItems" : 100
    },
    "entries" : [ {
      "entry" : {
          "id" : "fred-bloggs-yourcompany-com",
          "title" : "Fred Bloggs's Home",
          "visibility" : "PRIVATE",
          "description" : "Fred Bloggs's private home site."
      }
    } ]
  }
}
```

List order

Lists of these entities are returned ordered by ascending `title`.

Get a person's favorite sites

Method

Using the HTTP GET method:-

```
people/<personId>/favorite-sites
```

A personID is always the email address that they registered with.

Example request URL

```
https://api.alfresco.com/yourcompany.com/public/alfresco/versions/1/people/
fred.bloggs@yourcompany.com/favorite-sites
```

Response

- If the personId does not exist in this network, an HTTP `Not Found` (status 404) is returned.
- If the request is successful an HTTP OK is returned (status 200).

Example response body

```
{
    "list":{
        "pagination":{
            "count":1,
            "hasMoreItems":false,
            "skipCount":0,
```

```
            "maxItems":100AlfrescoRESTAPITheAlfrescoAPI43
        },
        "entries":[
            {
                "entry":{
                    "id":"fred-bloggs-yourcompany-com",
                    "site":{
                        "title":"Fred Bloggs's Home",
                        "description":"Fred Bloggs's private home site.",
                        "visibility":"PRIVATE",
                        "id":
                        "fred-bloggs-yourcompany-com"
                    }
                }
            }
        ]
    }
}
```

Note that each entry in the response list is a Favorite sites on page 45 entity with an embedded Sites on page 31 entity.

Preferences

A person's preferences in Alfresco.

Preferences object

Property	Type	JSON Type	Description
id	id	string	The unique preference id.
value	Any JSON primitive value	Any JSON primitive value	The value of the preference.

Example of a preferences object

```
{
       "value":true,
       "id":"org.alfresco.share.sites.favourites.fred-bloggs-yourcompany-com"
}
```

List order

Lists of these entities are returned ordered by ascending `id`.

Get a person's preferences

Method

Using the HTTP GET method:-

```
people/<personId>/preferences
```

A personID is always the email address that they registered with.

Example request URL

```
https://api.alfresco.com/yourcompany.com/public/alfresco/versions/1/people/
fred.bloggs@yourcompany.com/preferences
```

Response

- If the personId does not exist in this network, an HTTP `Not Found` (status 404) is returned.
- If the request is successful an HTTP OK is returned (status 200).

Example response body

```
{
  "list" : {
    "pagination" : {
      "count" : 3,
      "hasMoreItems" : false,
      "skipCount" : 0,
      "maxItems" : 100
    },
      "entries":[
        {
            "entry":{
              "value":"4452493d-675f-42f2-9fb9-50ee97c8c5c9,b8a10d93-
b383-4127-9f36-ff0ec5f2c450",
              "id":"org.alfresco.share.documents.favourites"
            }
        },
        {
            "entry":{
              "value":true,
              "id":"org.alfresco.share.sites.favourites.fred-bloggs-
yourcompany-com"
            }
        },
        {
            "entry":{
              "value":true,
              "id":"org.alfresco.share.sites.favourites.test-site-1"
            }
        }
      ]
   }
}
```

Get a preference

Method

Using the HTTP GET method:-

```
people/<personId>/preferences/<preferenceId>
```

A personID is always the email address that they registered with.

Example request URL

```
https://api.alfresco.com/yourcompany.com/public/alfresco/versions/1/people/
fred.bloggs@yourcompany.com/preferences/org.alfresco.share.documents.favourites
```

Response

- If the personId or the preferenceId does not exist in this network, an HTTP `Not Found` (status 404) is returned.
- If the request is successful an HTTP OK is returned (status 200).

Example response body

```
{
   "entry":{
      "value":"4452493d-675f-42f2-9fb9-50ee97c8c5c9,b8a10d93-b383-4127-9f36-
ff0ec5f2c450",
      "id":"org.alfresco.share.documents.favourites"
   }
```

```
}
```

Networks

A network is the group of users and sites that belong to an organization. Networks are organized by email domain. When a user signs up for an Alfresco account , their email domain becomes their Home Network.

Network object

See Networks on page 28 for information on the network entity.

Get a specific network

Method

Using the HTTP GET method:-

```
people/<personId>/networks/<networkId>
```

A personID is always the email address that they registered with.

Example request URL

```
https://api.alfresco.com/yourcompany.com/public/alfresco/versions/1/people/
fred.bloggs@yourcompany.com/networks/yourcompany.com
```

Response

- If the personId or the networkId does not exist in this network, an HTTP `Not Found` (status 404) is returned.
- If the request is successful an HTTP OK is returned (status 200).

Example response body

```
{
    "id" : "alfresco.com",
    "createdAt" : "2012-06-07T10:22:28.000+0000",
    "quotas" : [ {
      "limit" : 52428800,
      "id" : "fileUploadQuota"
    }, {
      "limit" : 5368709120,
      "usage" : 149102356,
      "id" : "fileQuota"
    }, {
      "limit" : -1,
      "usage" : 29,
      "id" : "siteCountQuota"
    }, {
      "limit" : -1,
      "usage" : 33,
      "id" : "personCountQuota"
    }, {
      "limit" : -1,
      "usage" : 15,
      "id" : "personInternalOnlyCountQuota"
    }, {
      "limit" : 0,
      "usage" : 0,
      "id" : "personNetworkAdminCountQuota"
    } ],
    "paidNetwork" : false,
    "isEnabled" : true,
```

```
    "subscriptionLevel" : "Free"
  }
```

Get a person's networks

Method

Using the HTTP GET method:-

```
people/<personId>/networks
```

A personID is always the email address that they registered with.

Example request URL

```
https://api.alfresco.com/yourcompany.com/public/alfresco/versions/1/people/
fred.bloggs@yourcompany.com/networks
```

Response

- If the personId does not exist in this network, an HTTP `Not Found` (status 404) is returned.
- If the request is successful an HTTP OK is returned (status 200).

Example response body

```
{
  "list" : {
    "pagination" : {
      "count" : 1,
      "hasMoreItems" : false,
      "totalItems" : 1,
      "skipCount" : 0,
      "maxItems" : 100
    },
    "entries" : [ {
      "entry" : {
        "id" : "yourcompany.com",
        "homeNetwork" : true,
        "network" : {
          "id" : "alfresco.com",
          "createdAt" : "2012-06-07T10:22:28.000+0000",
          "quotas" : [ {
            "limit" : 52428800,
            "id" : "fileUploadQuota"
          }, {
            "limit" : 5368709120,
            "usage" : 149102356,
            "id" : "fileQuota"
          }, {
            "limit" : -1,
            "usage" : 29,
            "id" : "siteCountQuota"
          }, {
            "limit" : -1,
            "usage" : 33,
            "id" : "personCountQuota"
          }, {
            "limit" : -1,
            "usage" : 15,
            "id" : "personInternalOnlyCountQuota"
          }, {
            "limit" : 0,
            "usage" : 0,
            "id" : "personNetworkAdminCountQuota"
          } ],
```