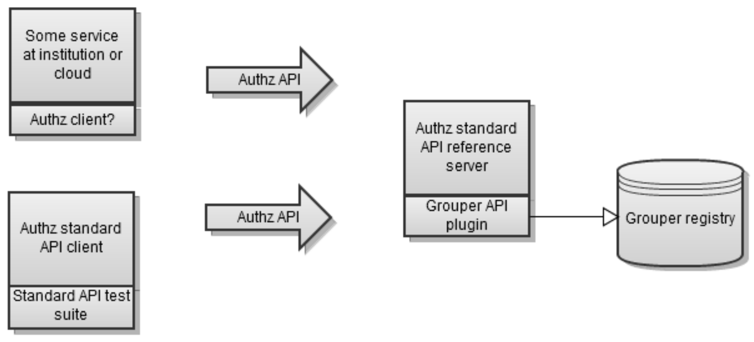
Authorization Standard API

This is a wiki for the standard API for Groups, Permissions

This is a RESTful API which is initially focused on HTTP/JSON.

- Authorization Standard API Calls Discussions
- Authorization Standard API Client
- Authorization Standard API Representations
- Authorization Standard API Test Harness
- Authorization Standard API Use Cases


Some service at institution or cloud
Authz client?
Authz API
Authz standard API client
Standard API test suite
Authz API
Authz standard API reference server
Grouper API plugin
Grouper registry

Open issues

- For lists of resources, e.g. /groups, the first 100 records will be returned if no paging is specified
- If pagingEnabled is false, or the limit is too high, the server's max limit will be imposed. Note, this max limit must be at least 1000.
- Validation on sorting/paging: you cannot send an offset without a limit. If you send pagingEnabled = false, then you shouldnt send in paging settings
- Fields (extra/omit)
- The folder path separator is a colon ":". An actual colon in a name part is represented by slash-colon ("\:").
- Groups and roles will be returned by group resources... yes

Standards

- top level: error (error code), error_description (free form), error_uri, status, warning, success
- Fields can be marked as
  - Mandatory: implementors must implement this field
  - Default: when searching for the resource, these fields will be returned. All fields are returned if the resource is requested. If the user cant access one of the subresources, it is still returned (non admins see the admin URI)
- Timestamps are ISO 8601, e.g.: 2012-10-04T03:10:14.123Z   note: no other 8601 representations are allowed.
- Request/response wrapper. Requests and response are wrapper in an object wrapper. Responses wrap objects which are the real responses, e.g. they are a field in the response wrapper. Responses have top level fields: meta, responseMeta:
  - meta (generic metadata about the resource)
    - lastModified (default, mandatory): timestamp when the resource was modified
    - selfUri (default, mandatory): URI to the current resource, includes URL params which are expected by server (not authn params etc)
    - structureName: name of the struct which is returned. This might be different if there is an error
  - responseMeta : metadata about this particular request/response that goes from server to client
    - responseTimestamp (mandatory): the timestamp that this was sent from the server (at the end of the processing)
    - millis (mandatory): number of milliseconds that the server took in processing this request
    - requestProcessed (mandatory): freeform text about the request that the server processed, when debugging to make sure the server is processing the right params
    - httpStatusCode: number of the HTTP status code
- Fields returned are controlled with the "fields" parameter, the "extraFields" parameter, and the "omitFields" parameter. The values of these params are not case-sensitive.
  - The values to these params can be qualified by object type, or if not qualified, refer to the requested resource object. e.g. for a group resource, or a group search, "id" or "group.id" will refer to the group id field.
  - If you do not specify anything, then all fields will be returned for the resource, and the default fields will be returned for a search (default fields are fields which are not generally difficult to compute, and which are commonly used).
  - If you specify "fields", then only the fields specified will be returned for the objects specified. E.g. fields=”id,name,meta.statusCode”
  - If you send in “extraFields” then in addition to the typical fields in the meta object, the totalCount field will also be supplied.   E.g. extraFields=meta.totalCount

    - If the values of these parameters conflict, there will be an error. You should not refer to the same object in more than one of the params. i.e. if you refer to group in "fields", then dont refer to it in "extraFields" or "omitFields". If you request a field which does not exist, it will be a warning, not an error.
    - If fields has "all" for the object type, then send all fields for that object type. If fields has "default" then send the default fields (e.g. the fields for a search result). If fields has "allEverywhere" then for all object types, send all fields
    - If you ask for a field that doesnt exist you will get a warning
- If unexpected fields are added to the request or reponse, it should not break the processing of the request or response. This is important for versioning. i.e. if you are not expecting the field group.foo, just ignore it (or log it), but do not throw an error.
- Fields should be named with camelcase. e.g. thisIsSomeField   names should be compatible with java variable names
- Any URI fields should be named "uri" or end in "Uri", unless it is a standard (e.g. oauth). e.g. membersUri, Note the URL is relative, though can be converted to absolute by considering the responseMeta.serviceRootUri field (ADD MORE HERE)

- Any link in the responses that is based on a named or id'ed resource, it should use the id'ed URI
- Method is the HTTP method, unless the HTTP ?method=DELETE is set, which has a value of a valid method, e.g. GET, PUT, POST, DELETE. Case insensitive on value
- If there are extra attributes, these in an extensions map in the object
- 401 means you arent authenticated, 403 means you are authenticated and you arent allowed
- Act as can be specified with: actAs=netId:jsmith (or whatever URI for the subject). The server will determine if that is allowed.
- Resources must end in the format requested. Possible formats are json and xml. e.g. https://groups.institution.edu/groupsApp/authzStandardApi/v1/groups.json or https://groups.institution.edu/groupsApp/authzStandardApi/v1/groups.xml
- Charset of the values is utf8
- Names should be uri's. e.g. group name could be name:a:b:c
- HTTP params are single valued, if there are two values for one param it is an error
- If an HTTP param is not expected, it should give a warning
- Fields and http params should not start with underscore, they should be alphanumeric and uncontroversial so they can be processed by various languages/formats
- Responses should be indented if this param is passed: ?indent=true is passed
- Boolean HTTP params can be: true|false or t|f or yes|no or y|n case insensitive
- For sorting and paging, see the section below
- Inserting / updating
    - For inserts use a POST, PUT is insert or update, for an update use PUT with ?saveMode=update
    - If you POST and it already exists, then 409 status "conflict explained"
    - If you PUT with ?saveMode=update and it doesnt already exist, that is a 412 error (precondition failed)
    - A PUT with a If-Match that does not match will return a 412 error (precondition failed)
    - Every PUT or POST needs to have a uri identifier in the URL. e.g. name:test:myGroup
- If you are updating and you do you specify a field, then it will not be updated. If you include the field and it is blank, then it will blank it out. If you use the url param: ?blankMissingFields=true then any missing fields will be blanked out

TODO

Discuss history of memberships

Versioning

The version of the API is an integer which is sent in the URL. If the API is changed in a backwards compatible way (e.g. extra fields which do not change names or structure), then the version number doesn't change. However, there is also a dot version number which includes another integer for the revision number which is changed whenever the API is changed in a backwards compatible way. The server dot version is sent in the response metadata.

Query params: paging, sorting

Clients can specify records on search requests to be sorted and/or paged. Note, if not specified, lists will be sorted based on a default sort field.

To put paging in the request, use these params: resource?offset=5&sortField=name&ascending=false&limit=200

Param name	Example value(s)	Description
pagingEnabled	true, false	To request that there shouldnt be paging, use this url param, pagingEnabled=false. pagingEnabled can be true or false. It doesnt have to be sent for "true", you can send limit instead. If you send pagingEnabled=false and limit set to something, that is an error. If pagingEnabled=false, the server can override this with an arbitrary limit and sortField. However, the server should support at least a count of 1000
limit	100	To set the size of the page, use limit. This number must be greater than 0. Note: if the count is too high the server can override the request and lower the count. However, the server must support at least a limit of 1000
offset	0	offset is zero indexes as to which page is returned. If it is not set, and there is no offsetFieldValue, it defaults to 0. Must be an integer greater than or equal to 0.
offsetFieldValue	"jsmith"	this is mutually exclusive with offset. If this is set, then the results will return everything after this value in the result list, until the resultlist is blank, then there is nothing else. This is more reliable than offset which can miss more records due to data churn while the paging is taking place. Note, it is imperative that the sort field be unique and non-null. If you want all records, you could sort on "id" and be safe. After receiving the first batch, then send the last value of the sorted field in this param to get the next batch
sortField	"displayName"	sortField is the resource specific sort field to sort on. Note, there could be more than one value that relates to the same field to make things easier to use. This should not be case-sensitive. If it is blank, and there is sorting, then a default sortField will be selected. You can set the sort field without using paging if you like.
ascending	true|false	ascending is a boolean true or false. This defaults to true.

extraFields=meta.totalCount		if the server should do a total count of records with the request and return in meta.totalCount. If it is not specified, then it will not be returned. This might affect the performance of the request if requested

When a search is returned, if it is paged or sorted or not cached, fields in the meta object will be returned with the total count (is extraFields=meta.totalCount is set) and a summary of what was done inside the meta object

```
{
  "meta":{
    "offset": 4,
    "limit": 100,
    "sortField":"displayName",
    "ascending":true,
    "totalCount":468
    "lastModified":"2012-11-04T09:57:03.541Z", ... etc
  },
  "responseMeta":{...  etc
}
```

- For lists of resources, e.g. /groups, the first 100 records will be returned if no paging is specified
- If pagingEnabled is false, or the limit is too high, the server's max limit will be imposed. Note, this max limit must be at least 1000.

Validations (will throw error):

- You cannot send an offset without a limit
- If you send pagingEnabled = false, then you shouldnt send in paging settings