

FluidDB API specification

Below is FluidDB API function documentation for `toplevel=`, `verb=`.

General information about the HTTP API can be found [here](#). Typographic conventions used on this page are listed [at bottom](#).

[/about](#): [POST](#), [GET](#), [HEAD](#), [PUT](#), [DELETE](#).
[/namespaces](#): [POST](#), [GET](#), [PUT](#), [DELETE](#).
[/objects](#): [POST](#), [GET](#), [HEAD](#), [PUT](#), [DELETE](#).
[/permissions](#): [POST](#), [GET](#), [PUT](#), [DELETE](#).
[/policies](#): [POST](#), [GET](#), [PUT](#), [DELETE](#).
[/tags](#): [POST](#), [GET](#), [PUT](#), [DELETE](#).
[/users](#): [GET](#).
[/values](#): [GET](#), [PUT](#), [DELETE](#).

About

POST

[link|top](#)

Create a new object.

POST `/about/aboutstr`

Create a new object with the given about value. If there is already a FluidDB object whose `fluiddb/about` tag has the given value, the returned object id will be that of the pre-existing object. In this call, and all others with an about value in the URI, you must convert your about value to UTF-8 and then [percent-encode](#) it before adding it to the request URI. For an example see the PUT request, below.

Response payload

The response payload will be sent in `application/json` format with the following fields:

Field name	Field type	Description
URI	unicode string	The URI of the new object.
id	unicode string	The id of the new object.

Example

Create an object with an about tag of 'chewing-gum'.

Request

```
POST /about/chewing-gum HTTP/1.1
Authorization: Basic XXXXXXXX
Content-Type: application/json
```

Response

```
HTTP/1.1 201 Created
Content-Length: 134
Location: http://fluiddb.fluidinfo.com/about/chewing-gum
Date: Mon, 02 Aug 2010 13:00:29 GMT
Content-Type: application/json

{"id": "9c8e4b12-4b7d-40d2-865b-d5b1945350b1",
 "URI": "http://fluiddb.fluidinfo.com/about/chewing-gum"}
```

HTTP response status codes

The following HTTP response codes may occur:

Condition	Return
The <code>aboutstr</code> argument was not valid UTF-8.	400 (Bad Request)
An error with the request payload. More details.	400 (Bad Request)
An error with the request makes it impossible to respond. More details.	400 (Bad Request)
A new object was created without error.	201 (Created)

GET

[link|top](#)

The GET method for `/about` retrieves information about the FluidDB object (if any) with a particular `fluidddb/about` value, or retrieves the value of a tag on the object with a given `about` tag value.

GET `/about/aboutstr`

To request information on the object whose `fluidddb/about` tag is `aboutstr`, specify the `about` value in the request URI.

Response payload

The response payload will be sent in `application/json` format with the following fields:

Field name	Field type	Description
<code>id</code>	unicode string	The id of the object.
<code>tagPaths</code>	list of unicode strings	The full path names of the tags on this object (for which the user has READ permission), if any.

Example

Retrieve information about the FluidDB object whose `fluidddb/about` value is 'Barcelona'.

Request

```
GET /about/Barcelona HTTP/1.1
Authorization: Basic XXXXXXXXX
```

Response

```
HTTP/1.1 200 OK
Content-Length: 1080
Date: Mon, 02 Aug 2010 13:16:09 GMT
Content-Type: application/json

{"tagPaths": ["esteve\\opinion", "terrycojones\\favorite-cities"]}
```

HTTP response status codes

The following HTTP response codes may occur:

Condition	Return
The <code>aboutstr</code> argument was not valid UTF-8.	400 (Bad Request)
No object with the given <code>about</code> value exists.	404 (Not Found)
An error with the request makes it impossible to respond. More details.	400 (Bad Request)
No error occurred.	200 (OK)

GET `/about/aboutstr/namespace1/namespace2/tag`

Retrieve the value of a tag from the object (if any) whose `fluidddb/about` is `aboutstr`. The value is returned in the payload of the response. Please see [here](#) for more details.

Example

Return a primitive value type from FluidDB. Note that the Content-Type header in the response is `application/vnd.fluiddb.value+json` as given to FluidDB when the value was originally PUT.

Request

```
GET /about/London/geo/latitude HTTP/1.1
Authorization: Basic XXXXXXXXX
```

Response

```
HTTP/1.1 200 OK
Content-Length: 4
Date: Mon, 02 Aug 2010 13:20:31 GMT
Content-Type: application/vnd.fluiddb.value+json

51.5
```

Example

Return an opaque value type from FluidDB. Note that the Content-Type header in the response is application/pdf, because the tag value is of that type, as given to FluidDB when the value was originally PUT.

Request

```
GET /about/Barcelona/lonelyplanet.com/outline HTTP/1.1
Authorization: Basic XXXXXXXX
```

Response

```
HTTP/1.1 200 OK
Content-Length: 6666
Date: Mon, 02 Aug 2010 13:25:28 GMT
Content-Type: application/pdf

%PDF-1.4
1 0 obj
<< /Length 2 0 R
/Filter /FlateDecode
>>
stream
...
```

HTTP response status codes

The following HTTP response codes may occur:

Condition	Return
The <code>aboutstr</code> argument was not valid UTF-8.	400 (Bad Request)
No object with the given <code>about</code> value exists.	404 (Not Found)
If the user does not have READ permission on the tag.	401 (Unauthorized)
An error with the request makes it impossible to respond. More details.	400 (Bad Request)
If you do not specify an <code>Accept</code> header value that allows the tag value to be returned.	406 (Not Acceptable)
If the object has an instance of the tag and the requesting user has READ permission for the tag.	200 (OK)

HEAD

[link|top](#)

The HEAD method on `about` can be used to test whether an object has a given tag or not, without retrieving the value of the tag.

`HEAD /about/aboutstr/namespace1/namespace2/tag`

Test for the existence of a tag on the object (if any) whose `fluiddb/about` is `aboutstr`.

Example

Retrieve information about a tag attached to a specific object.

Request

```
HEAD /about/Barcelona/ntoll/review HTTP/1.1
Authorization: Basic XXXXXXXX
```

Response

```
HTTP/1.1 200 OK
Content-Length: 31288
```

HTTP response status codes

The following HTTP response codes may occur:

Condition	Return
The <code>aboutstr</code> argument was not valid UTF-8.	400 (Bad Request)
No object with the given <code>about</code> value exists.	404 (Not Found)
If the user does not have READ permission on the tag.	401 (Unauthorized)
The object has an instance of the tag.	200 (OK)

PUT

[link|top](#)

Create or update a tag on an object.

`PUT /about/aboutstr/namespace1/namespace2/tag`

Create or update a tag on the object whose `fluiddb/about` is `aboutstr`. If no object with `fluiddb/about` = `aboutstr` exists, it will be created. The tag value is sent in the payload of the request. See [here](#) for more details.

Example

Write a primitive value (in this case 'true') to the FluidDB object whose `fluiddb/about` value is 台北 (Taipei). 台北 are the two unicode codepoints U+53F0 and U+5317, each of which is a 3-byte sequence in UTF-8 (0xE5 0x8F 0xB0 and 0xE5 0x8C 0x97 respectively). Note the Content-Type header in the request is set to `application/vnd.fluiddb.value+json` to indicate to FluidDB that this is a primitive value.

Request

```
PUT /about/%E5%8F%B0%E5%8C%97/ntoll/cities/visited HTTP/1.1
Authorization: Basic XXXXXXXXX
Content-Length: XXXXXXXXX
Content-Type: application/vnd.fluiddb.value+json

true
```

Response

```
HTTP/1.1 204 No Content
Date: Mon, 02 Aug 2010 14:31:47 GMT
Content-Type: text/html
```

Example

Write an opaque value type to FluidDB. Note that the Content-Type header in the request is set to `application/pdf` indicating to FluidDB that this value is opaque. When this value is retrieved the Content-Type header in the response will be `application/pdf`. (The body of the request has been truncated.)

Request

```
PUT /about/Barcelona/lonelyplanet.com/outline HTTP/1.1
Authorization: Basic XXXXXXXXX
Content-Length: XXXXXXXXX
Content-Type: application/pdf

%PDF-1.4
1 0 obj
<< /Length 2 0 R
/Filter /FlateDecode
>>
stream
...
```

Response

```
HTTP/1.1 204 No Content
Date: Mon, 02 Aug 2010 14:33:40 GMT
Content-Type: text/html
```

HTTP response status codes

The following HTTP response codes may occur:

Condition	Return
The <code>aboutstr</code> argument was not valid UTF-8.	400 (Bad Request)
If the user does not have CREATE permission on the tag.	401 (Unauthorized)
An error with the request payload. More details.	400 (Bad Request)
If the tag is successfully created / updated on the object.	204 (No Content)

DELETE

[link|top](#)

Delete a tag from an object. Note that [it is not possible to delete a FluidDB object.](#)

`DELETE /about/aboutstr/namespace1/namespace2/tag`

Delete a tag from an object.

Example

Delete the `ntoll/rating` tag from the object about Barcelona.

Request

```
DELETE /about/Barcelona/ntoll/rating HTTP/1.1
Authorization: Basic XXXXXXXX
```

Response

```
HTTP/1.1 204 No Content
Date: Mon, 02 Aug 2010 14:56:52 GMT
Content-Type: text/html
```

HTTP response status codes

The following HTTP response codes may occur:

Condition	Return
If an intermediate namespace does not exist.	404 (Not Found)
If the tag does not exist.	404 (Not Found)
The <code>aboutstr</code> argument was not valid UTF-8.	400 (Bad Request)
No object with the given <code>about</code> value exists.	404 (Not Found)
If the user does not have DELETE permission on the tag.	401 (Unauthorized)
The tag is successfully deleted from the object.	204 (No Content)

Namespaces

POST

[link|top](#)

`POST /namespaces/namespace1/namespace2`

Create a new namespace.

Request payload

The request payload must be sent in `application/json` format with the following fields:

Field name	Field type	Mandatory	Description
<code>description</code>	unicode string	True	A description of the namespace.
<code>name</code>	unicode string	True	The name of the new namespace.

Response payload

The response payload will be sent in `application/json` format with the following fields:

Field name	Field type	Description
URI	unicode string	The URI of the new namespace.
id	unicode string	The id of the object that corresponds to the new namespace.

Example

Create a new namespace called 'people' in the test user's top-level namespace.

Request

```
POST /namespaces/test HTTP/1.1
Authorization: Basic XXXXXXXX
Content-Length: XXXXXXXX
Content-Type: application/json

{
  "description": "A namespace for tags that I'm using to add to people",
  "name": "people"
}
```

Response

```
HTTP/1.1 201 Created
Content-Length: 110
Location: http://fluiddb.fluidinfo.com/namespaces/test/people
Date: Mon, 02 Aug 2010 12:40:41 GMT
Content-Type: application/json

{"id": "e9c97fa8-05ed-4905-9f72-8d00b7390f9b",
 "URI": "http://fluiddb.fluidinfo.com/namespaces/test/people"}
```

HTTP response status codes

The following HTTP response codes may occur:

Condition	Return
If the namespace already exists.	412 (Precondition Failed)
If a parent namespace does not exist.	404 (Not Found)
If the requesting user does not have CREATE permission on the parent namespace.	401 (Unauthorized)
If the full path of the new namespace is too long. The current maximum path length is 233 characters.	400 (Bad Request)
An error with the request payload. More details.	400 (Bad Request)
An error with the request makes it impossible to respond. More details.	400 (Bad Request)
If the namespace is created successfully.	201 (Created)

Note:

1. The new namespace will have permissions set according to the user's defaults. There is no permission inheritance in FluidDB.

GET

[link|top](#)

GET /namespaces/namespace1/namespace2

Get information about the namespaces contained in a namespace.

URI arguments:

Name	Type	Default	Mandatory	Description
returnDescription	Boolean	False	False	If True, also return the namespace description.
returnNamespaces	Boolean	False	False	If True, also return the names of the namespaces in this namespace.
returnTags	Boolean	False	False	If True, also return the names of the tags in this namespace.

Response payload

The response payload will be sent in `application/json` format with the following fields:

Field name	Field type	Description
<code>description</code>	unicode string	A description of the namespace. This field is only present if <code>returnDescription</code> is <code>True</code> in the request.
<code>id</code>	unicode string	The id of the FluidDB object corresponding to the namespace.
<code>namespaceNames</code>	list of unicode strings	The names of the sub-namespaces in this namespace. This field is only present if <code>returnNamespaces</code> is <code>True</code> in the request.
<code>tagNames</code>	list of unicode strings	The names of the tags in this namespace (only present if <code>returnTags</code> is <code>True</code> in the request).

Example

Retrieve information about the `test/people` namespace.

Request

```
GET /namespaces/test/people?returnDescription=True&returnNamespaces=True&returnTags=True HTTP/
Authorization: Basic XXXXXXXX
```

Response

```
HTTP/1.1 200 OK
Content-Length: 118
Date: Mon, 02 Aug 2010 12:43:05 GMT
Content-Type: application/json

{"tagNames": [], "namespaceNames": [], "id": "e9c97fa8-05ed-4905-9f72-8d00b7390f9b",
"description": "A namespace for tags I'm using to add to people"}
```

HTTP response status codes

The following HTTP response codes may occur:

Condition	Return
If the namespace does not exist.	404 (Not Found)
If an intermediate namespace does not exist.	404 (Not Found)
If the requesting user does not have <code>LIST</code> permission on the namespace.	401 (Unauthorized)
An error with the request makes it impossible to respond. More details .	400 (Bad Request)
No error.	200 (OK)

PUT

[link|top](#)

`PUT /namespaces/namespace1/namespace2`

Update a namespace.

Request payload

The request payload must be sent in `application/json` format with the following fields:

Field name	Field type	Mandatory	Description
<code>description</code>	unicode string	True	A description of the namespace.

Example

Update the description of the `test/people` namespace.

Request

```
PUT /namespaces/test/people HTTP/1.1
Authorization: Basic XXXXXXXX
Content-Length: XXXXXXXX
Content-Type: application/json

{
  "description": "Contains tags used to annotate objects representing people"
```

```
}
```

Response

```
HTTP/1.1 204 No Content  
Date: Mon, 02 Aug 2010 12:46:38 GMT  
Content-Type: text/html
```

HTTP response status codes

The following HTTP response codes may occur:

Condition	Return
If the namespace does not exist.	404 (Not Found)
If an intermediate namespace does not exist.	404 (Not Found)
If the requesting user does not have UPDATE permission on the namespace.	401 (Unauthorized)
An error with the request payload. More details.	400 (Bad Request)
If the namespace is updated successfully.	204 (No Content)

DELETE

[link|top](#)

DELETE /namespaces/namespace1/namespace2

Delete a namespace.

Example

Delete the test/people namespace.

Request

```
DELETE /namespaces/test/people HTTP/1.1  
Authorization: Basic XXXXXXXXX
```

Response

```
HTTP/1.1 204 No Content  
Date: Mon, 02 Aug 2010 12:47:25 GMT  
Content-Type: text/html
```

HTTP response status codes

The following HTTP response codes may occur:

Condition	Return
If the namespace does not exist.	404 (Not Found)
If an intermediate namespace does not exist.	404 (Not Found)
If the requesting user does not have DELETE permission on the namespace.	401 (Unauthorized)
If the namespace is not empty. A namespace is empty if it contains no other namespaces or tags.	412 (Precondition Failed)
If the namespace is deleted successfully.	204 (No Content)

Note:

1. A namespace can only be deleted if it is empty. I.e., it must not contain any sub-namespaces or any tags.

Objects

POST

[link|top](#)

Create a new FluidDB object

Create a new FluidDB object.

POST /objects

Create a new object.

Request payload

The request payload must be sent in `application/json` format with the following fields:

Field name	Field type	Mandatory	Description
about	unicode string	False	The value for the <code>about</code> tag. If you do not want an <code>about</code> tag on this object, omit this field.

Response payload

The response payload will be sent in `application/json` format with the following fields:

Field name	Field type	Description
URI	unicode string	The URI of the new object.
id	unicode string	The id of the new object.

Example

Create an object with an `about` tag that indicates the object represents the book 'Dune'.

Request

```
POST /objects HTTP/1.1
Authorization: Basic XXXXXXXXX
Content-Length: XXXXXXXXX
Content-Type: application/json

{
  "about": "book:Dune"
}
```

Response

```
HTTP/1.1 201 Created
Content-Length: 134
Location: http://fluiddb.fluidinfo.com/objects/
```

Example

Create a new object without an `about` tag.

Request

```
POST /objects HTTP/1.1
Authorization: Basic XXXXXXXXX
Content-Length: XXXXXXXXX
Content-Type: application/json
```

Response

```
HTTP/1.1 201 Created
Content-Length: 134
Location: http://fluiddb.fluidinfo.com/objects/
```

HTTP response status codes

The following HTTP response codes may occur:

Condition	Return
An error with the request payload. More details.	400 (Bad Request)
An error with the request makes it impossible to respond. More details.	400 (Bad Request)
A new object was created without error.	201 (Created)

Notes:

1. The response will also contain a `Location` header giving the URI for the new object.
2. If you pass an `about` value and there is already a FluidDB object whose `about` tag has that value, the returned object id will be that of the pre-existing object.

GET

[link|top](#)

The GET method on objects is used to retrieve objects matching a query (on object tags), to retrieve information about a particular object, or to retrieve the value of a tag on a particular object.

[GET /objects](#)

Search for objects that match a query.

URI arguments:

Name	Type	Default	Mandatory	Description
query	string		True	A query string specifying what sorts of objects to return. The query language is described here . You must convert your query to UTF-8 and then percent-encode it before adding it to the request URI

Response payload

The response payload will be sent in `application/json` format with the following fields:

Field name	Field type	Description
ids	list of unicode strings	A list of object ids matching the query. Each object id is a UUID string (as described here).

Example

Search for objects in FluidDB (note the URL encoded query argument).

Request

```
GET /objects?query=has%20ntoll/
```

Response

```
HTTP/1.1 200 OK
Content-Length: 209
Date: Mon, 02 Aug 2010 13:14:32 GMT
Content-Type: application/json

{"ids": ["5a4823a4-26b4-495c-9a29-a1e830a1b153", "8b07a7ec-e5f7-46cd-9cd1-9a40c3137762", "83f2ad81-43db-421a-adc4-974f3a8bca0d", "ac1e937e-dd76-426e-a2b5-06a439a708cc", "52bc041d-e8f4-4f8d-a66a-a3b5ea5fa156"]}
```

HTTP response status codes

The following HTTP response codes may occur:

Condition	Return
If no query is given.	400 (Bad Request)
The query string was not valid UTF-8.	400 (Bad Request)
If the query string could not be parsed.	400 (Bad Request)
If the query (or any of its sub-parts) results in too many matching objects. The current limit is 1 million objects.	413 (Request Entity Too Large)
If the user does not have READ permission on a tag whose value is needed to satisfy the query.	401 (Unauthorized)
An error with the request makes it impossible to respond. More details.	400 (Bad Request)
No error occurred.	200 (OK)

[GET /objects/id](#)

To request information on a particular object, include the object ID in the request URI. The return information will indicate which tags are present on the object, though not with the tag values.

URI arguments:

Name	Type	Default	Mandatory	Description
------	------	---------	-----------	-------------

Name	Type	Default	Mandatory	Description
showAbout	string	False	False	If True, return the value of the <code>about</code> tag on the object. If the object does not have an <code>about</code> tag, the response payload will still contain an <code>about</code> key, with, in the case of a JSON payload, a <code>null</code> value. If False, the payload will not have an <code>about</code> key.

Response payload

The response payload will be sent in `application/json` format with the following fields:

Field name	Field type	Description
<code>about</code>	unicode string	The <code>about</code> tag on the object, if any.
<code>tagPaths</code>	list of unicode strings	The full path names of the tags on this object (for which the user has <code>READ</code> permission), if any.

Example

Retrieve information about a specific object (the 'tagPaths' list in the response has been truncated)

Request

```
GET /objects/5a4823a4-26b4-495c-9a29-a1e830a1b153
```

Response

```
HTTP/1.1 200 OK
Content-Length: 1080
Date: Mon, 02 Aug 2010 13:16:09 GMT
Content-Type: application/json

{"about": "twitter.com:uid:5893972", "tagPaths": ["twitter.com\\friends\\fxn"]}
```

HTTP response status codes

The following HTTP response codes may occur:

Condition	Return
If no object with the given ID exists.	404 (Not Found)
An error with the request payload. More details .	400 (Bad Request)
An error with the request makes it impossible to respond. More details .	400 (Bad Request)
No error occurred.	200 (OK)

Note:

1. An object ID has the form of a UUID, as described [here](#).

`GET /objects/id/namespace1/namespace2/tag`

Retrieve the value of a tag from an object. The value is returned in the payload of the response. Please see [here](#) for more details.

Example

Return a primitive value type from FluidDB. Note that the Content-Type header in the response is `application/vnd.fluiddb.value+json`, as given to FluidDB when the value was PUT.

Request

```
GET /objects/5a4823a4-26b4-495c-9a29-a1e830a1b153/
```

Response

```
HTTP/1.1 200 OK
Content-Length: 6
Date: Mon, 02 Aug 2010 13:20:31 GMT
Content-Type: application/vnd.fluiddb.value+json

"HD42"
```

Example

Example

Return an opaque value type from FluidDB. Note that the Content-Type header in the response is text/html, because the tag value is of that type, as given to FluidDB when the value was PUT.

Request

```
GET /objects/366375a5-c811-4c59-a469-0a3efb602411/
```

Response

```
HTTP/1.1 200 OK
Content-Length: 6666
Date: Mon, 02 Aug 2010 13:25:28 GMT
Content-Type: text/html

<html>
  <head>
    <title>Hello</title>
  </head>
  <body>
    Hello
  </body>
</html>
```

HTTP response status codes

The following HTTP response codes may occur:

Condition	Return
If the user does not have READ permission for the tag.	401 (Unauthorized)
An error with the request makes it impossible to respond. More details.	400 (Bad Request)
If you do not specify an <code>Accept</code> header value that allows the tag value to be returned.	406 (Not Acceptable)
If the object has an instance of the tag.	200 (OK)

HEAD

[link|top](#)

The HEAD method on objects can be used to test whether an object has a given tag or not, without retrieving the value of the tag.

[HEAD /objects/id/namespace1/namespace2/tag](#)

Test for the existence of a tag on an object.

Example

Retrieve the headers associated with a tag attached to a specific object.

Request

```
HEAD /objects/366375a5-c811-4c59-a469-0a3efb602411/
```

Response

```
HTTP/1.1 200 OK
Content-Length: 28926
Date: Mon, 02 Aug 2010 13:26:25 GMT
Content-Type: text/html
```

HTTP response status codes

The following HTTP response codes may occur:

Condition	Return
If the user does not have READ permission for the tag.	401 (Unauthorized)
The object has an instance of the tag.	200 (OK)

PUT

[link|top](#)

Create or update a tag on an object.

`PUT /objects/id/namespace1/namespace2/tag`

Create or update a tag on an object. The tag value is sent in the payload of the request. See [here](#) for more details.

Example

Write a primitive value type to FluidDB. Note the Content-Type header in the request is set to application/vnd.fluiddb.value+json to indicate to FluidDB that this is a primitive value.

Request

```
PUT /objects/5a4823a4-26b4-495c-9a29-a1e830a1b153/
```

Response

```
HTTP/1.1 204 No Content
Date: Mon, 02 Aug 2010 14:31:47 GMT
Content-Type: text/html
```

Example

Write an opaque value type to FluidDB. Note that the Content-Type header in the request is set to application/pdf indicating to FluidDB that this value is opaque. When this value is retrieved the Content-Type header in the response will be application/pdf. (Note: the body of the request has been truncated.)

Request

```
PUT /objects/5a4823a4-26b4-495c-9a29-a1e830a1b153/
```

Response

```
HTTP/1.1 204 No Content
Date: Mon, 02 Aug 2010 14:33:40 GMT
Content-Type: text/html
```

HTTP response status codes

The following HTTP response codes may occur:

Condition	Return
If the user does not have CREATE permission on the tag.	401 (Unauthorized)
An error with the request payload. More details.	400 (Bad Request)
If the tag is successfully created / updated on the object.	204 (No Content)

DELETE

[link|top](#)

Delete a tag from an object. Note that [it is not possible to delete a FluidDB object.](#)

`DELETE /objects/id/namespace1/namespace2/tag`

Delete a tag from an object.

Example

Delete the test/quz tag from the referenced object.

Request

```
DELETE /objects/5a4823a4-26b4-495c-9a29-a1e830a1b153/
```

Response

```
HTTP/1.1 204 No Content
Date: Mon, 02 Aug 2010 14:56:52 GMT
Content-Type: text/html
```

HTTP response status codes

The following HTTP response codes may occur:

Condition	Return
If the object id is malformed.	404 (Not Found)
If an intermediate namespace does not exist.	404 (Not Found)
If the tag does not exist.	404 (Not Found)
If the user does not have DELETE permission on the tag.	401 (Unauthorized)
The tag is successfully deleted from the object.	204 (No Content)

Note:

1. You will receive a 204 (No Content) status even if the object has no instance of the tag.

Permissions

POST

[link|top](#)

POST is not supported on permissions, because permissions are automatically set from a user's default permissions when a namespace or tag is first created. Use PUT to adjust the permissions on a given namespace or tag.

GET

[link|top](#)

GET /permissions/namespaces/namespace1/namespace2

Get the permissions on a namespace: the open/closed policy, and the set of exceptions to the policy.

URI arguments:

Name	Type	Default	Mandatory	Description
action	string		True	The action whose permissions information is sought. Possible values are: create, update, delete, list, control.

Response payload

The response payload will be sent in application/json format with the following fields:

Field name	Field type	Description
exceptions	list of unicode strings	The names of the users who are exceptions to the policy.
policy	unicode string	The policy (either 'open' or 'closed').

Example

Retrieve the 'create' policy for the test namespace.

Request

```
GET /permissions/namespaces/test?action=create HTTP/1.1
Authorization: Basic XXXXXXXX
```

Response

```
HTTP/1.1 200 OK
Content-Length: 44
Date: Mon, 02 Aug 2010 14:58:28 GMT
Content-Type: application/json

{"policy": "closed", "exceptions": ["test"]}
```

HTTP response status codes

The following HTTP response codes may occur:

Condition	Return
If an intermediate namespace does not exist.	404 (Not Found)
If the namespace does not exist.	404 (Not Found)
If the requesting user does not have CONTROL permission on the namespace.	401 (Unauthorized)
If the action argument is missing or invalid.	400 (Bad Request)
An error with the request makes it impossible to respond. More details.	400 (Bad Request)
No error.	200 (OK)

[GET /permissions/tags/namespace1/namespace2/tag](#)

Get the permissions on a tag: the open/closed policy, and the set of exceptions to the policy.

URI arguments:

Name	Type	Default	Mandatory	Description
action	string		True	The action whose permissions information is sought. Possible values are: update, delete, control.

Response payload

The response payload will be sent in `application/json` format with the following fields:

Field name	Field type	Description
exceptions	list of unicode strings	The names of the users who are exceptions to the policy.
policy	unicode string	The policy (either 'open' or 'closed').

Example

Retrieve the 'update' policy for the test/quz tag.

Request

```
GET /permissions/tags/test/quz?action=update HTTP/1.1
Authorization: Basic XXXXXXXX
```

Response

```
HTTP/1.1 200 OK
Content-Length: 44
Date: Mon, 02 Aug 2010 14:59:22 GMT
Content-Type: application/json

{"policy": "closed", "exceptions": ["test"]}
```

HTTP response status codes

The following HTTP response codes may occur:

Condition	Return
If an intermediate namespace does not exist.	404 (Not Found)
If the tag does not exist.	404 (Not Found)
If the requesting user does not have CONTROL permission on the tag.	401 (Unauthorized)
If the action argument is missing or invalid.	400 (Bad Request)
An error with the request makes it impossible to respond. More details.	400 (Bad Request)
No error.	200 (OK)

[GET /permissions/tag-values/namespace1/namespace2/tag](#)

Get the permissions on the set of tag instances: the open/closed policy, and the set of exceptions to the policy.

URI arguments:

Name	Type	Default	Mandatory	Description
action	string		True	The action whose permissions information is sought. Possible values are: create, read, delete, control.

Response payload

The response payload will be sent in `application/json` format with the following fields:

Field name	Field type	Description
<code>exceptions</code>	list of unicode strings	The names of the users who are exceptions to the policy.
<code>policy</code>	unicode string	The policy (either 'open' or 'closed').

Example

Retrieve the 'delete' policy for values associated with the `test/quz` tag.

Request

```
GET /permissions/tag-values/test/quz?action=delete HTTP/1.1
Authorization: Basic XXXXXXXXX
```

Response

```
HTTP/1.1 200 OK
Content-Length: 36
Date: Mon, 02 Aug 2010 14:59:54 GMT
Content-Type: application/json

{"policy": "open", "exceptions": []}
```

HTTP response status codes

The following HTTP response codes may occur:

Condition	Return
If an intermediate namespace does not exist.	404 (Not Found)
If the tag does not exist.	404 (Not Found)
If the requesting user does not have CONTROL permission on the tag.	401 (Unauthorized)
If the action argument is missing or invalid.	400 (Bad Request)
An error with the request makes it impossible to respond. More details .	400 (Bad Request)
No error.	200 (OK)

PUT

[link|top](#)

`PUT /permissions/namespaces/namespace1/namespace2`

Update the permissions on a namespace: the open/closed policy, and the set of exceptions to the policy.

URI arguments:

Name	Type	Default	Mandatory	Description
<code>action</code>	string		True	The action whose permissions information is to be updated. Possible values are: create, update, delete, list, control.

Request payload

The request payload must be sent in `application/json` format with the following fields:

Field name	Field type	Mandatory	Description
<code>exceptions</code>	list of unicode strings	True	The names of the users who are exceptions to the policy.
<code>policy</code>	unicode string	True	The policy (either 'open' or 'closed').

Example

Update the 'create' policy for the test namespace.

Request

```
PUT /permissions/namespaces/test?action=create HTTP/1.1
Authorization: Basic XXXXXXXXX
Content-Length: XXXXXXXXX
Content-Type: application/json
```

```
{
  "policy": "closed",
  "exceptions": ["test", "ntoll"]
}
```

Response

```
HTTP/1.1 204 No Content
Date: Mon, 02 Aug 2010 15:03:42 GMT
Content-Type: text/html
```

HTTP response status codes

The following HTTP response codes may occur:

Condition	Return
If an intermediate namespace does not exist.	404 (Not Found)
If the namespace does not exist.	404 (Not Found)
If the requesting user does not have CONTROL permission on the namespace.	401 (Unauthorized)
If the policy is not 'open' or 'closed'.	400 (Bad Request)
If the action argument is missing or invalid.	400 (Bad Request)
An error with the request payload. More details.	400 (Bad Request)
If permissions are changed successfully.	204 (No Content)

[PUT /permissions/tags/namespace1/namespace2/tag](#)

Update the permissions on a tag: the open/closed policy, and the set of exceptions to the policy.

URI arguments:

Name	Type	Default	Mandatory	Description
action	string		True	The action whose permissions information is to be updated. Possible values are: update, delete, control.

Request payload

The request payload must be sent in `application/json` format with the following fields:

Field name	Field type	Mandatory	Description
exceptions	list of unicode strings	True	The names of the users who are exceptions to the policy.
policy	unicode string	True	The policy (either 'open' or 'closed').

Example

Update the 'update' policy for the test/quz tag.

Request

```
PUT /permissions/tags/test/quz?action=update HTTP/1.1
Authorization: Basic XXXXXXXXX
Content-Length: XXXXXXXXX
Content-Type: application/json

{
  "policy": "closed",
  "exceptions": ["test", "ntoll"]
}
```

Response

```
HTTP/1.1 204 No Content
Date: Mon, 02 Aug 2010 15:04:39 GMT
Content-Type: text/html
```

HTTP response status codes

The following HTTP response codes may occur:

Condition	Return
-----------	--------

If an intermediate namespace does not exist.	404 (Not Found)
If the tag does not exist.	404 (Not Found)
If the requesting user does not have CONTROL permission on the tag.	401 (Unauthorized)
If the policy is not 'open' or 'closed'.	400 (Bad Request)
If the action argument is missing or invalid.	400 (Bad Request)
An error with the request payload. More details.	400 (Bad Request)
If permissions are changed successfully.	204 (No Content)

PUT /permissions/tag-values/namespace1/namespace2/tag

Update the permissions on a tag's instances: the open/closed policy, and the set of exceptions to the policy.

URI arguments:

Name	Type	Default	Mandatory	Description
action	string		True	The action whose permissions information is to be updated. Possible values are: create, read, delete, control.

Request payload

The request payload must be sent in `application/json` format with the following fields:

Field name	Field type	Mandatory	Description
exceptions	list of unicode strings	True	The names of the users who are exceptions to the policy.
policy	unicode string	True	The policy (either 'open' or 'closed').

Example

Update the 'create' policy for values associated with the test/quz tag.

Request

```
PUT /permissions/tag-values/test/quz?action=create HTTP/1.1
Authorization: Basic XXXXXXXX
Content-Length: XXXXXXXX
Content-Type: application/json

{
  "policy": "closed",
  "exceptions": ["test", "ntoll"]
}
```

Response

```
HTTP/1.1 204 No Content
Date: Mon, 02 Aug 2010 15:08:44 GMT
Content-Type: text/html
```

HTTP response status codes

The following HTTP response codes may occur:

Condition	Return
If an intermediate namespace does not exist.	404 (Not Found)
If the tag does not exist.	404 (Not Found)
If the requesting user does not have CONTROL permission on the tag.	401 (Unauthorized)
If the policy is not 'open' or 'closed'.	400 (Bad Request)
If the action argument is missing or invalid.	400 (Bad Request)
An error with the request payload. More details.	400 (Bad Request)
If permissions are changed successfully.	204 (No Content)

DELETE

[link|top](#)

DELETE is not supported on permissions because namespaces and tags must always have a set of permissions.

Policies

POST

[link|top](#)

POST is not supported on policies because users have a set of policies assigned when they are created.

GET

[link|top](#)

`GET /policies/username/category/action`

Get a user's default permission policy for a given category and action. These are the permissions that will be assigned to new tags and namespaces created by the user.

`category` must be one of 'namespaces', 'tags' or 'tag-values'. The `actions` available by category are as follows.

- For namespaces: 'create', 'update', 'delete', and 'list'.
- For tags: 'update' and 'delete'.
- For tag-values: 'create', 'read', and 'delete'.

Response payload

The response payload will be sent in `application/json` format with the following fields:

Field name	Field type	Description
<code>exceptions</code>	list of unicode strings	The names of the users who are exceptions to the policy.
<code>policy</code>	unicode string	The policy (either 'open' or 'closed').

Example

Retrieve the default 'create' policy for the test user's namespaces.

Request

```
GET /policies/test/namespaces/create HTTP/1.1
Authorization: Basic XXXXXXXXX
```

Response

```
HTTP/1.1 200 OK
Content-Length: 44
Date: Mon, 02 Aug 2010 15:11:35 GMT
Content-Type: application/json

{"policy": "closed", "exceptions": ["test"]}
```

HTTP response status codes

The following HTTP response codes may occur:

Condition	Return
If the user does not exist.	404 (Not Found)
If the category/action pair is invalid or missing.	404 (Not Found)
An error with the request makes it impossible to respond. More details .	400 (Bad Request)
The policy is returned successfully.	200 (OK)

PUT

[link|top](#)

`PUT /policies/username/category/action`

Update a user's default permission policy for a given category and action. These are the permissions that will be assigned to new tags and namespaces created by the user.

The `category` and `action` components of the URI are as for GET.

Request payload

The request payload must be sent in `application/json` format with the following fields:

Field name	Field type	Mandatory	Description
exceptions	list of unicode strings	True	The names of the users who are exceptions to the policy.
policy	unicode string	True	The policy (either 'open' or 'closed').

Example

Update the default 'create' policy for the test user's namespaces.

Request

```
PUT /policies/test/namespaces/create HTTP/1.1
Authorization: Basic XXXXXXXX
Content-Length: XXXXXXXX
Content-Type: application/json

{
  "policy": "closed",
  "exceptions": ["test"]
}
```

Response

```
HTTP/1.1 204 No Content
Date: Mon, 02 Aug 2010 15:12:40 GMT
Content-Type: text/html
```

HTTP response status codes

The following HTTP response codes may occur:

Condition	Return
If the requesting user is not the user in the URI	401 (Unauthorized)
If the user does not exist.	404 (Not Found)
If the category/action pair is invalid or missing.	400 (Bad Request)
An error with the request payload. More details.	400 (Bad Request)
Policy updated successfully.	204 (No Content)

DELETE

[link|top](#)

DELETE is not supported on policies because users must always have a set of policies to be used in the creation of new tags and namespaces.

Tags

POST

[link|top](#)

POST `/tags/namespace1/namespace2`

Add a tag name to a namespace.

Request payload

The request payload must be sent in `application/json` format with the following fields:

Field name	Field type	Mandatory	Description
description	unicode string	True	A description of the tag.
indexed	boolean	True	Whether or not tag values should be indexed.
name	unicode string	True	The name of the new tag.

Response payload

The response payload will be sent in `application/json` format with the following fields:

Field name	Field type	Description
URI	unicode string	The URI of the new object.
id	unicode string	The id of the object that corresponds to the new tag.

Example

Create a new tag called 'rating' in the test users top-level namespace.

Request

```
POST /tags/test HTTP/1.1
Authorization: Basic XXXXXXXX
Content-Length: XXXXXXXX
Content-Type: application/json

{
  "description": "How I rate things on a scale of 1 (worst) to 10 (best).",
  "indexed": false,
  "name": "rating"
}
```

Response

```
HTTP/1.1 201 Created
Content-Length: 110
Location: http://fluiddb.fluidinfo.com/tags/test/rating
Date: Mon, 02 Aug 2010 15:15:00 GMT
Content-Type: application/json

{"id": "56e0c31a-1a4c-4091-8a65-b37af769752a",
 "URI": "http://fluiddb.fluidinfo.com/tags/test/rating"}
```

HTTP response status codes

The following HTTP response codes may occur:

Condition	Return
If the containing namespace or an intermediate namespace does not exist.	404 (Not Found)
If the user does not have CREATE permission on the containing (i.e., deepest) namespace.	401 (Unauthorized)
If the tag already exists.	412 (Precondition Failed)
If the full path of the new tag is too long. The current maximum path length is 233 characters.	400 (Bad Request)
An error with the request payload. More details.	400 (Bad Request)
An error with the request makes it impossible to respond. More details.	400 (Bad Request)
If the tag is created successfully.	201 (Created)

Notes:

1. A `Location` header will be returned containing the URI of the new tag.
2. This method creates a tag itself, *not* the value of a tag on a particular object (for that, use POST on `/objects/id/namespace1/namespace2/tag`).

GET

[link|top](#)

GET `/tags/namespace1/namespace2/tag`

Retrieve information about the tag.

URI arguments:

Name	Type	Default	Mandatory	Description
returnDescription	Boolean	False	False	If True, return the description of the tag.

Response payload

The response payload will be sent in `application/json` format with the following fields:

The response payload will be sent in application/json format with the following fields.

Field name	Field type	Description
description	unicode string	A description of the tag. This field is only present if returnDescription is True in the request.
id	unicode string	The id of the FluidDB object corresponding to the tag.
indexed	boolean	Whether or not tag values are indexed.

Example

Retrieve information about the test/rating tag.

Request

```
GET /tags/test/rating?returnDescription=True HTTP/1.1
Authorization: Basic XXXXXXXXX
```

Response

```
HTTP/1.1 200 OK
Content-Length: 108
Date: Mon, 02 Aug 2010 15:15:59 GMT
Content-Type: application/json

{"indexed": false, "id": "56e0c31a-1a4c-4091-8a65-b37af769752a",
"description": "How I rate things on a scale of 1 (worst) to 10 (best)."}

```

HTTP response status codes

The following HTTP response codes may occur:

Condition	Return
If an intermediate namespace does not exist.	404 (Not Found)
If the tag does not exist.	404 (Not Found)
An error with the request makes it impossible to respond. More details .	400 (Bad Request)
No error.	200 (OK)

Note:

1. This method retrieves information about the tag itself, *not* the value of a tag on an object (to get the value on an object, use GET on [/objects/id/namespace1/namespace2/tag](#)).

PUT

[link|top](#)

[PUT /tags/namespace1/namespace2/tag](#)

Update information about a tag in a namespace.

Request payload

The request payload must be sent in application/json format with the following fields:

Field name	Field type	Mandatory	Description
description	unicode string	True	A description of the tag.

Example

Update the description of the test/rating tag.

Request

```
PUT /tags/test/rating HTTP/1.1
Authorization: Basic XXXXXXXXX
Content-Length: XXXXXXXXX
Content-Type: application/json

{
  "description": "Indicates rating from 1 (bad) -> 10 (good)"
}
```

Response

```
HTTP/1.1 204 No Content
Date: Mon, 02 Aug 2010 15:16:59 GMT
Content-Type: text/html
```

HTTP response status codes

The following HTTP response codes may occur:

Condition	Return
If an intermediate namespace does not exist.	404 (Not Found)
If the tag does not exist.	404 (Not Found)
If the requesting user does not have UPDATE permission on the tag.	401 (Unauthorized)
An error with the request payload. More details.	400 (Bad Request)
If the operation completes successfully.	204 (No Content)

Note:

1. Changing whether a tag is indexed after it has been created is not currently supported.

DELETE

[link|top](#)

[DELETE /tags/namespace1/namespace2/tag](#)

Delete a tag. The tag name is removed from its containing namespace and all occurrences of the tag on objects are deleted.

Example

Delete the test/rating tag.

Request

```
DELETE /tags/test/rating HTTP/1.1
Authorization: Basic XXXXXXXX
```

Response

```
HTTP/1.1 204 No Content
Date: Mon, 02 Aug 2010 15:17:38 GMT
Content-Type: text/html
```

HTTP response status codes

The following HTTP response codes may occur:

Condition	Return
If an intermediate namespace does not exist.	404 (Not Found)
If the tag does not exist.	404 (Not Found)
If the requesting user does not have DELETE permission on the tag.	401 (Unauthorized)
If the operation completes successfully.	204 (No Content)

Users

GET

[link|top](#)

[GET /users/username](#)

Return information about a particular user.

Response payload

The response payload will be sent in `application/json` format with the following fields:

Field name	Field type	Description
------------	------------	-------------

id	unicode string	The id of the object corresponding to the user.
name	unicode string	The user's name.

Example

Retrieve information about the user 'ntoλλ'. Note that to specify unicode 'λ' characters in the request URI, you must first convert them to UTF-8 and then URL-encode them. Unicode is returned in the response, in the manner specified by the JSON specification.

Request

```
GET /users/nto%CE%BB%CE%BB HTTP/1.1
```

Response

```
HTTP/1.1 200 OK
Content-Length: 62
Date: Mon, 02 Aug 2010 15:18:17 GMT
Content-Type: application/json

{"name": "Nicholas To\u03bb\u03bbbervey",
 "id": "42909deb-9854-47ae-a8f8-3f59d4fbe5a5"}
```

HTTP response status codes

The following HTTP response codes may occur:

Condition	Return
If the user does not exist.	404 (Not Found)
An error with the request makes it impossible to respond. More details.	400 (Bad Request)
No error.	200 (OK)

Values

GET

[link|top](#)

The GET method on values is used to retrieve tag values from objects matching a query.

[GET /values](#)

Search for objects matching a FluidDB query, and return the value of the requested tags on the matching objects.

URI arguments:

Name	Type	Default	Mandatory	Description
query	string		True	A query string specifying what objects to match. The FluidDB query language is described here .
tag	string		True	The name of a tag whose value should be returned. Repeat this argument as many times as needed.

Response payload

The response payload will be sent in `application/json` format with the following fields:

Field name	Field type	Description
results	dictionary	A dictionary containing information about the requested tag values on objects that match the query. See example below.

Example

Search for objects in FluidDB matching `mike/rating > 5` and retrieve values for the tags `ntoll/rating`, `ntoll/resume`, `fluiddb/about`, and `bit.ly/url` from those objects.

Request

```
GET /values?query=mike%2Frating>5&tag=ntoll/rating&tag=ntoll/resume&tag=fluiddb/about&tag=bit.
```

Authorization: Basic XXXXXXXX

Response

```
HTTP/1.1 200 OK
Content-Length: 817
Date: Mon, 02 Aug 2010 13:14:32 GMT
Content-Type: application/json

{
  "results" : {
    "id" : {
      "05eee31e-fbd1-43cc-9500-0469707a9bc3" : {
        "fluiddb/about" : {
          "value" : "Hey you"
        },
        "ntoll/rating" : {
          "value" : 5
        },
        "ntoll/resume" : {
          "value-type" : "application/pdf",
          "size" : 179393
        }
      },
      "0521e31e-fbd1-43cc-9500-046974569bc3" : {
        "fluiddb/about" : {
          "value" : "http://www.yahoo.com"
        },
        "ntoll/rating" : {
          "value" : 9
        },
        "bit.ly/url" : {
          "value" : "http://bit.ly/4bYAV2"
        }
      }
    }
  }
}
```

HTTP response status codes

The following HTTP response codes may occur:

Condition	Return
If no query or tag is given.	400 (Bad Request)
If the query string could not be parsed.	400 (Bad Request)
If the query (or any of its sub-parts) results in too many matching objects. The current limit is 1 million objects.	413 (Request Entity Too Large)
If the requesting user does not have READ permission on a tag whose value is needed to satisfy the query.	401 (Unauthorized)
If the requesting user does not have READ permission on a tag whose value is requested.	401 (Unauthorized)
If a tag whose value is requested does not exist.	404 (Not Found)
An error with the request makes it impossible to respond. More details.	400 (Bad Request)
No error occurred.	200 (OK)

PUT

[link|top](#)

The PUT method on values is used to set tag values on objects matching a query.

[PUT /values](#)

Search for objects matching a FluidDB query, and set the values of the tags given in the payload on the matching objects.

URI arguments:

Name	Type	Default	Mandatory	Description
query	string		True	A query string specifying what objects to match. The FluidDB query language is described here .

Example

Search for objects in FluidDB matching `mike/rating > 5` and update the tags `ntoll/rating` and `ntoll/seen` with the values 6 and true (respectively) on those objects.

Request

```
PUT /values?query=mike%2Frating>5 HTTP/1.1
Authorization: Basic XXXXXXXX
Content-type: application/json
Content-length: 104

{
  "ntoll/rating" : {
    "value" : 6
  },
  "ntoll/seen" : {
    "value" : true
  }
}
```

Response

```
HTTP/1.1 204 No Content
Content-Type: text/html
Date: Mon, 02 Aug 2010 13:14:32 GMT
```

HTTP response status codes

The following HTTP response codes may occur:

Condition	Return
If no query is given.	400 (Bad Request)
If the query string could not be parsed.	400 (Bad Request)
If the payload specifying tag values cannot be parsed.	400 (Bad Request)
If the query (or any of its sub-parts) results in too many matching objects. The current limit is 1 million objects.	413 (Request Entity Too Large)
If the requesting user does not have READ permission on a tag whose value is needed to satisfy the query.	401 (Unauthorized)
If the requesting user does not have CREATE permission on a tag whose update is requested.	401 (Unauthorized)
If a tag whose update is requested does not exist.	404 (Not Found)
No error occurred.	204 (No Content)

DELETE

[link|top](#)

The DELETE method on values is used to delete tags from objects matching a query.

`DELETE /values`

Search for objects matching a FluidDB query, and delete the the requested tags from the matching objects.

URI arguments:

Name	Type	Default	Mandatory	Description
query	string		True	A query string specifying what objects to match. The FluidDB query language is described here .
tag	string		True	The name of a tag that should be deleted from objects matching the query. Repeat this argument as many times as needed.

Example

Search for objects in FluidDB matching `mike/rating > 5` and delete the tags `ntoll/rating` and `ntoll/resume` from those objects.

Request

```
DELETE /values?query=mike%2Frating>5&tag=ntoll/rating&tag=ntoll/resume HTTP/1.1
Authorization: Basic XXXXXXXX
```

Response

```
HTTP/1.1 204 No Content
Content-Type: text/html
Date: Mon, 02 Aug 2010 13:14:32 GMT
```

HTTP response status codes

The following HTTP response codes may occur:

Condition	Return
If no query or tag is given.	400 (Bad Request)
If the query string could not be parsed.	400 (Bad Request)
If the query (or any of its sub-parts) results in too many matching objects. The current limit is 1 million objects.	413 (Request Entity Too Large)
If the requesting user does not have READ permission on a tag whose value is needed to satisfy the query.	401 (Unauthorized)
If the requesting user does not have DELETE permission on a tag whose deletion is requested.	401 (Unauthorized)
If a tag whose deletion is requested does not exist.	404 (Not Found)
No error occurred.	204 (No Content)

Typographic conventions

The following typographic conventions are used above:

- URIs look like `/users/john`
- When part of a URI is variable, it will look like `/users/USERNAME` (here USERNAME is variable)
- The names of namespaces and tags in FluidDB look like `john/books/rating`
- Variable parts of namespaces and tags are distinguished as with URIs, e.g., `users/USERNAME`
- Permissions look like **DELETE**