HTTP authentication API NG

Motivation

`org.clazzes.login.http` is a the HTTP based implementation of DomainPasswordLoginService.

While the old HTTP authentication request is satisfying for user/password checks, new optional features like group membership queries require new handshakes for the HTTP backend API.

This document speficies the next-gen HTTP authentication API.

Contents

- Basic Handshake Pattern
  - Basic Request Pattern
  - Basic Response pattern
- Required operations
  - tryLogin
  - getSupportedOperations
- Optional Operations
  - changePassword
  - deactivateUser
  - getDefaultDomain
  - getGroups
  - getGroupMembers
  - sendPassword
  - searchUser

Basic Handshake Pattern

Basic Request Pattern

A request to an authentication URL is a HTTPS POST request like this:

```
POST /my/authentication/service HTTP/1.1
Host: auth.my.domain
Content-Type: application/x-www-form-urlencoded

op=<op>&param1=<value1>&param2=<value2>
```

<op> is the operation requested, usually the name of the method in DomainPasswordLoginService.java.

To provide backwards compatibility, the `op` parameter is optional and defaults to `tryLogin`.

See below for detailed examples.

Basic Response pattern

Every respond to an authentication request is answered with a HTTP response with

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

and on of the following status codes:

```
200 OK - login is ok, or other operation was completed successfully
403 Forbidden - the login is invalid or the operation is not permitted
404 Not found - if a user could not be found during a search operation
406 Not Acceptable - too many unsuccessful authentications, or other reason to suspect a brute force attack
```

The response body must not be empty and must be UTF-8 encoded, it's content is specified differently for each operation.

For most operations the reponse is either

- a short message for logging (not more than 1024 bytes)
- or a list of values separated by '`,`'
- or '`-`' for "empty list"/"no data"

- or '`--`' for "not supported by backend"

The server may enforce the use of HTTP basic authentication in order to keep offending servers away from dictionary attacks.

JSON variants

A backend may support to return the response in the form of small JSON documents.

To trigger json response, add the parameter `json=1` to the request, like this:

```
POST /my/authentication/service HTTP/1.1
Host: auth.my.domain
Content-Type: application/x-www-form-urlencoded

op=<op>&json=1&param1=<value1>&param2=<value2>
```

To explicitly disable JSON response, use `json=0` instead.

Backends might choose to support only one variant, only with or only without JSON response.

With JSON reponses on, the repsonse is either

- a short info message, like

  ```
  { "info" : "Some message to use in log files" }
  ```

- or a list of named values, for examples scroll down to the operation chapters
- or a empty list if no data can be found
- or an error message for "not supported by backend" or similar problems, like

  ```
  { "error" : "Operation not supported by backend for specified domain" }
  ```

Required operations

tryLogin

Request body (new format, preferred)

```
op=tryLogin&user=<user>&domain=<domain>&passwd=<passwd>
```

The `domain` parameter is optional.

Request body in old format, supported for backward compatibility reasons

```
user=<user>&passwd=<passwd>
```

Response body (plain non-JSON variant)

Non-empty information text, not more than 1024 bytes. The message may go into logfiles and should not be displayed to the user.

Response body (JSON variant)

Successful:

```
{ "user" : "jdoe", "prettyName" : "John Doe", "eMailAddress" : "jdoe@foo.bar" }
```

Not found or problem: See documentation of "searchUser".

getSupportedOperations

Request body

```
op=getSupportedFeatures
```

Response body (plain non-JSON variant)

List of suppored operations, separated by ','.

Example showing minimal feature set:

```
getSupportedOperations,tryLogin
```

Example specifying maximum feature set:

```
getSupportedOperations,tryLogin,changePassword,deactivateUser,getDefaultDomain,getGroups,sendPassword,searchUser
```

Response body (JSON variant)

```
[ "getSupportedOperations", "tryLogin" ]
```

Optional Operations

changePassword

Changes the password of the user.

Request body

```
op=changePassword&user=<user>&domain=<domain>&oldPassword=<oldPassword>&newPassword=<newPassword>&newPasswordConfirmed=<newPassword>
```

The `domain` parameter is optional.

The `newPasswordConfirmed` parameter is optional and available only to simplify writing web interfaces. If it is specified and does not match `newPassword`, the password is not changed.

Response body

Non-empty information text, not more than 1024 bytes. The message may go into logfiles and should not be displayed to the user.

deactivateUser

Deactivates a user, prevents him for logging in again.

Request body

```
op=deactivateUser&user=<user>&domain=<domain>
```

The `domain` parameter is optional.

Response body

Non-empty information text, not more than 1024 bytes. The message may go into logfiles and should not be displayed to the user.

getDefaultDomain

Returns the default domain, if there is any.

Request body

```
op=getDefaultDomain
```

Response body (plain non-JSON variant)

Default authentication domain, or '`-`' if there is no default domain, or '`--`' if there is no domain support at all.

Response body (JSON variant)

```
[ "SOMEDOMAIN" ]
```

getGroups

Returns the groups the user is a member of.

Request body

```
op=getGroups&user=<user>&domain=<domain>
```

The `domain` parameter is optional.

Response body (plain non-JSON variant)

List of group names, separated by '`,`' or just '`-`' if the user is not member of any group, or '`--`' if there is no group support.

Response body (JSON variant)

The following example shows a list of 2 groups, one with maximum details, one wiht miniimal details:

```
[
  { "group" : "users", "prettyName" : "Human users of this system", "domain" : "MYDOMAIN" } ,
  { "group" : "dialout" }
]
```

getGroupMembers

Returns the users the are a member of the specified group.

Request body

```
op=getGroupMembers&group=<group>&domain=<domain>
```

The `domain` parameter is optional.

Response body (plain non-JSON variant)

List of group names, separated by '`,`' or just '`-`' if the user is not member of any group, or '`--`' if there is no group support.

Response body (JSON variant)

```
[
  { "user" : "leonard", "prettyName" : "Leonard Hofstaetter", "eMailAddress" : "lh@tbbt.foo.bar" } ,
  { "user" : "penny" } ,
  { "user" : "sheldon" }
]
```

sendPassword

Generates a new password or send a "new password" link to the user.

Request body

```
op=sendPassword&user=<user>&domain=<domain>
```

The `domain` parameter is optional.

Response body

Non-empty information text, not more than 1024 bytes. The message may go into logfiles and should not be displayed to the user.

searchUser

Searches a user in the database, sets response code to 200 if the user is there, 404 if the user could not be found.

Request body

```
op=searchUser&user=<user>&domain=<domain>
```

The `domain` parameter is optional.

Response body

Non-empty information text, not more than 1024 bytes. The message may go into logfiles and should not be displayed to the user.

Response body (JSON variant)

Successful, with response code 200:

```
{ "user" : "jdoe", "prettyName" : "John Doe", "eMailAddress" : "jdoe@foo.bar" }
```

Not found, with response code 404:

```
{ "error" : "user not found" }
```

Problem, with repsonse code 500:

```
{ "error" : "Operation not supported by backend for specified domain" }
```