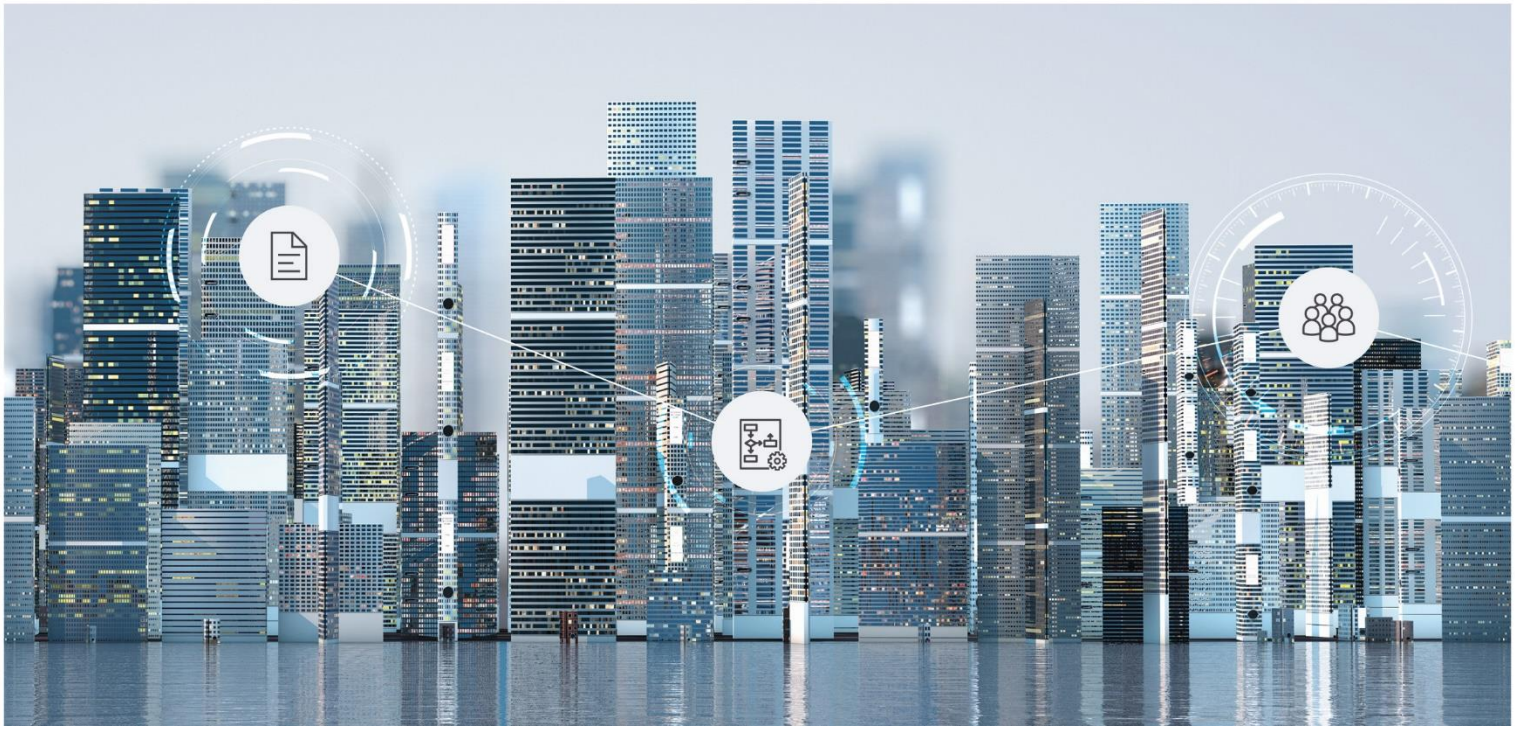
Fabasoft



White Paper

Fabasoft Folio Authentication Methods

Fabasoft Folio 2023 Update Rollup 2

Contents

1 Introduction

Fabasoft Folio offers a wide range of authentication methods. This white paper describes the necessary configuration steps for the following methods:

- Authentication with Kerberos (Basic, Cookie)
- Authentication with LDAP (Basic)
- Authentication with LDAP (Basic, Cookie)
- Authentication with SAML
- Authentication with External (REMOTE_USER)
- Authentication with OAuth

2 Software Requirements

System environment: All information contained in this document implicitly assumes a Microsoft Windows or Linux environment.

Supported platforms: For detailed information on supported operating systems and software see the software product information on the Fabasoft distribution media.

Descriptions in this document are based on the following software:

Authentication with SAML

- Shibboleth IdP 4.2.1

3 Preparation

Fabasoft Folio supports the following authentication methods in addition to those provided by the web server:

Authentication Method	Environment Variable Value
Default Represents the configured web server authentication method (e.g. basic authentication).	`DEFAULT`
Kerberos (Basic, Cookie)	`KERBEROS_BASIC`
LDAP (Basic)	`LDAP`
LDAP (Basic, Cookie)	`LDAP_COOKIE`
SAML	`SAML`
External (REMOTE_USER)	`REMOTE_USER`
OAuth	`OAUTH`

To change an authentication method, perform the following steps:

1. Set the authentication method for a domain, a host, or a web service.
2. Enable anonymous access for all affected web services.
3. Configure the authentication method (see the next chapters for details).
4. Restart all affected web services.

3.1 Set an Authentication Method for a Domain

To set an authentication method for a domain, perform the following steps:

1. Open the *Virtual Application Configuration*, which is referenced in the *Current Domain* or *Domain Type*.
2. Click the "Authentication" tab.
3. In the *Authentication Method* field define the desired authentication method.

3.2 Set an Authentication Method for a Host

To set an authentication method for a host and all its web services, perform the following steps:

1. Open the relevant update file.
   Microsoft Windows Environment: `%ProgramData%\Fabasoft\coomk.upd`
   Linux Environment: `/var/opt/fabasoft/coo.upd`
2. Add the following entry to the file:
   `HKEY_ENVIRONMENT\FSCVEXT_AUTHMETH=<environment-variable-value>`

Other environment variables can be set in a similar manner.

3.3 Set an Authentication Method for a Web Service

In a Microsoft Windows environment, the service specific environment variable can be set by adding the following keys to the registry, where `<id>` represents the virtual directory of the web service:

```
Windows Registry Editor Version 5.00

[HKEY_LOCAL_MACHINE\SOFTWARE\Fabasoft\FscWeb\Modules\<id>]
@="%ProgramFiles%\\Fabasoft\\Components\\Web\\<id>\\ASP\\content\\bin\\fscvext.dll
"
[HKEY_LOCAL_MACHINE\SOFTWARE\Fabasoft\FscWeb\Modules\<id>\FSCVEXT_AUTHMETH]
@="<environment-variable-value>"
```

`<id>` represents the directory name in `%ProgramFiles%\Fabasoft\Components\Web` that corresponds to the target web service.

In a Linux environment, the registry keys can be created as follows:

```
su – fscsrv
mkdir –p
/etc/fabasoft/settings/machine/SOFTWARE/Fabasoft/FscWeb/Modules/<id>
cd /etc/fabasoft/settings/machine/SOFTWARE/Fabasoft/FscWeb/Modules/<id>
echo -n /opt/fabasoft/share/web/<id>/asp/content/bin/libfscvext.so >
registry.default
mkdir FSCVEXT_AUTHMETH
echo -n "<environment-variable-value>" > FSCVEXT_AUTHMETH/registry.default
```

`<id>` represents the directory name in `/var/opt/fabasoft/instances` that corresponds to the target web service.

3.4 Enable Anonymous Access

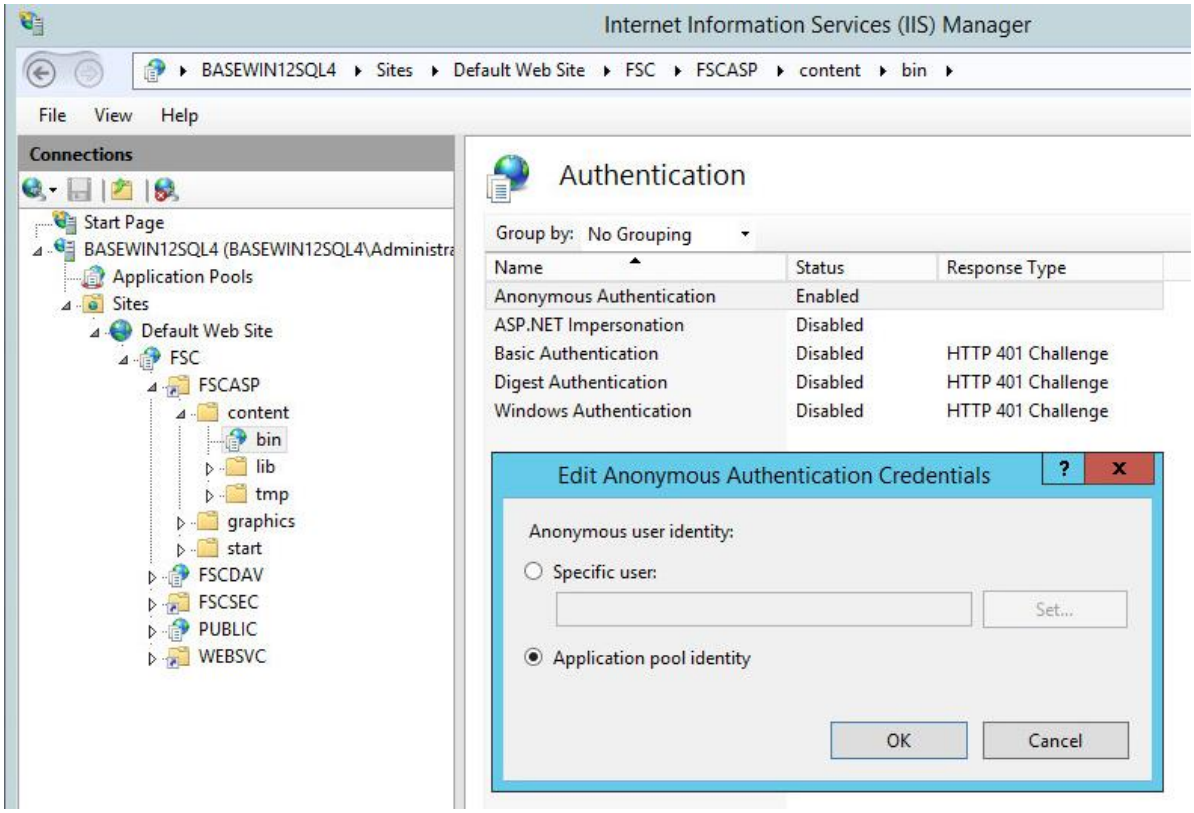
3.4.1 Linux Environment

On Linux, install a Kerberos-enabled Fabasoft Folio Web Service. In case of existing web services with basic authentication, this can be done by commenting out the authentication relevant entries in the file `/etc/fabasoft/web/WebService_<id>.conf` like shown below. `<id>` is the number of the web service.

```
<Location /fsc>
  SetHandler fscvext
  #AuthName "Fabasoft Components"
  #Require valid-user
  #AuthType Basic
  #AuthUserFile /etc/fabasoft/web/htpasswd
  #AuthGroupFile /etc/fabasoft/web/htgroups
</Location>
```

3.4.2 Microsoft Windows Environment

On Microsoft Windows environments, enable anonymous access to the Fabasoft Folio Web Service and disable all other authentication methods. Ensure that the *Application pool identity* is used for the *Anonymous Authentication* method.



4 Authentication with Kerberos (Basic, Cookie)

The following chapters describe a basic authentication method using Kerberos and HTTP cookies. This authentication method authenticates using Kerberos credentials entered by the user and then stores a cookie with the authentication information, so that credentials are only required and

validated during the initial request. Consequently, clients such as the Fabasoft Folio Client do not require credentials once the cookie is available.

Performance Note: This authentication method should only be used for web services that are accessed interactively via web browsers. Otherwise, HTTP requests from non-browser clients that ignore cookies set by the server (e.g. conversion service requests) may cause significant performance problems because every single HTTP request has to create a new Fabasoft Folio session in that scenario. Use the environment variable `FSCVEXT_AUTHMETH` to configure the authentication method for specific hosts or web services.

4.1 Configuration

The following settings are necessary for the configuration of Kerberos (basic, cookie):

1. Open the *Virtual Application Configuration*, which is referenced in the *Current Domain* or *Domain Type*.
2. Click the "Authentication" tab.

The following relevant properties are available:

- *Cookies*
  - *Session Cookie*
    Set the value of this property to "Yes", if the authentication should only be valid during a user session.
  - *Authentication Expires After Minutes*
    Define after how many minutes the authentication of a user expires, who is actively working with the system.
  - *Authentication of an Idle Session Expires After Minutes*
    Define after how many minutes the authentication of a user expires, who is not actively working with the system.

Via the environment variable `FSCVEXT_AUTHBASICDOMAIN` it is possible to define a default domain for authentication used when only a user name without a domain name was applied. Additionally, on Linux, this variable will be used to resolve short domain names. Therefore it is possible to define more domain names separately by a ';'. Doing so, the first specified domain of the list will be used as default domain.

Example: `FSCVEXT_AUTHBASICDOMAIN = "default.test.com; eng.test.com; sq.test.com"`

5 Authentication With LDAP (Basic)

LDAP (Lightweight Directory Access Protocol) is a widely used network protocol to access directory services. A directory consists of a tree of directory entries (also known as objects), that usually reflect organizational structures. Regardless of the overall complexity of a directory infrastructure and contents, the information contained will, eventually, branch down to an entry for a user, which is what we are interested in, in terms of authentication.

Entries in the directory consist of a "Distinguished Name" (DN), serving as basic unique identifier in the tree, plus a set of attributes, possessing a unique name and one or more values, each.

Note: The DN can change over the lifetime of a particular entry in the directory, for instance, when personnel are reassigned to another department. It is thus not suited for the purpose of external references that point to an entry in the directory.

The set of attributes is defined in a schema. These are part of the configuration of the directory server, and are referenced via the `objectClass` attribute of an entry.

User entry objects are usually identified by having the object class `posixAccount` assigned. Examples for attributes defined in the `posixAccount` object class: `uid`, `loginShell`, `password`, `homeDirectory`.

LDAP supports a set of actions that can be issued against the directory service; these actions include (but are not limited to):

- Search (search and, optionally, retrieve entries)
- Bind (authenticate)
- Add
- Delete
- Modify

5.1 Principle of Operation

Via LDAP, searches can be conducted for entries in the tree, depending on one or more of the attributes in order to find the desired entry and retrieve its DN.

A sample search expression to find a certain user could look like:

```
(&(objectClass=posixAccount)(uid=Cadence.Jones))
```

The ampersand denotes a conjunctive operation, thus only entries whose attribute `objectClass` contain `posixAccount` and attribute `uid` equals `Cadence.Jones` will be returned from the search.

Now, when a user login name is received by the Fabasoft Folio Web Service in the course of a HTTP authentication process, the user entry can be found in the LDAP service, using a search filter similar to the above example.

The resulting DN is used to issue a bind operation against the entry in the directory. Bind additionally requires a password, which has been supplied by the user via the HTTP authentication dialogue, too. If the LDAP bind operation succeeds, the user is granted access to the Fabasoft Folio Web Service.

For additional information, please refer to the documentation provided with your directory server software of choice and start off by reading http://en.wikipedia.org/wiki/Ldap.

More information on HTTP basic authentication is available in http://en.wikipedia.org/wiki/Basic_authentication_scheme.

5.2 Requirements

- A well-configured directory server, accessible via LDAP
- Anonymous search access to the directory
- Fabasoft Folio Web Service
- User entries have their attribute `objectClass` contain `posixAccount` (from `nis.schema`)
- Valid user entry and password to test proper operation.

- Verify that the package `pam_ldap` is installed.

5.3 Configuration

Create a new *Service Data Source* object. Alternatively, the default Fabasoft Folio `LDAPDataSource` object can be used. Add the following (minimal) set of parameters:

- *Parameters*
  - *Data Source Type*
    "LDAP" or "LDAPS" to define the LDAP protocol to be used.
  - *Server*
    The hostname of the LDAP server.
  - *Searchbase*
    The location in the LDAP where the search begins.
    Example: `ou=org,dc=comp,dc=com`
  - *Filterkey*
    The LDAP filter used for the user lookup.
    Example: `(&(objectClass=posixAccount)(uid=%u))`

Notes:

- The sequence %u will be replaced with the user logon name as retrieved from the HTTP basic authentication data entered by the user.
- The parameter *Server* may contain a whitespace-separated list of hosts to try to connect to, and each host may optionally be of the form `host:port`. If present, the `:port` overrides the *Port* parameter.

By default, anonymous directory bind access is required. Alternatively, the parameters *Username* and *Password* can be configured to authenticate before the directory bind operation.

To enable LDAPS, add the parameter *Transport* with the value `ldaps`. Additionally, make sure that the CA certificate of the LDAP server is trusted by the web server. Import the CA certificate into the *Trusted Root Certificate Authorities* store.

The created LDAP data source can be configured in the *System Configuration* tab of the *Primary Domain.*

5.4 Configuration (Linux Environment)

If LDAP authentication should also be used for signatures and/or the Fabasoft Folio Web Management, additional configuration steps are required.

5.4.1 Operating System

In `/etc/pam_ldap.conf`, change host and base to reflect your directory server settings. Verify proper operation by issuing `ldapsearch`, which should retrieve all entries under the search base in `ldap.conf`.

```
ldapsearch -x "objectClass=*"
```

5.4.2 Signatures

To allow password signatures for objects in Fabasoft Folio, a named PAM facility needs to exist.

Create the file `/etc/pam.d/fscweb` containing:

```
#%PAM-1.0
auth        required pam_ldap.so
account     required pam_ldap.so
```

Signatures will now be authenticated via LDAP against the directory service, except when Kerberos authentication is in use, because that will take precedence.

5.4.3 Web Management

Create `/etc/pam.d/fscwmc` identical to `fscweb` above. If the authentication scheme is identical to that of the web service, a symbolic link. (`ln -s`) can be created as well, pointing to the file `fscweb`. The web management service uses a different facility to enable utilization of different authentication mechanisms for it.

Ensure that users permitted to authenticate on the web management service are members of the group `fsc`, either locally or via LDAP (attribute `gid`).

5.5 Recommendations

- All attributes that are searched for frequently (such as `uid`), should be defined as indices in the directory server's configuration, the LDAP backend database shall be re-indexed frequently, if so required.
- Confine LDAP search operations to the smallest possible sub tree in the directory; use `ou=subcomp,dc=comp,dc=com` instead of only `dc=comp,dc=com` as a search base.
- Using the unambiguous entry identifiers (i.e. `entryUUID` in OpenLDAP) as foreign keys when referring to the LDAP tree enables you to fearlessly adapt the layout of the tree frequently in order to reflect changes in your organizational structure.

6 Authentication with LDAP (Basic, Cookie)

The following chapters describe a basic authentication method using LDAP and HTTP cookies. This authentication method authenticates using LDAP credentials entered by the user and then stores a cookie with the authentication information, so that credentials are only required and validated during the initial request. Consequently, clients such as the Fabasoft Folio Client do not require credentials once the cookie is available.

Performance Note: This authentication method should only be used for web services that are accessed interactively via web browsers. Otherwise, HTTP requests from non-browser clients that ignore cookies set by the server (e.g. conversion service requests) may cause significant performance problems because every single HTTP request has to create a new Fabasoft Folio session in that scenario. Use the environment variable `FSCVEXT_AUTHMETH` to configure the authentication method for specific hosts or web services.

6.1 Configuration LDAP

Follow the steps to configure for Authentication With LDAP (Basic).

6.2 Configuration Cookie

The following additional settings are necessary for the configuration of LDAP (basic, cookie):

3. Open the *Virtual Application Configuration*, which is referenced in the *Current Domain* or *Domain Type*.
4. Click the "Authentication" tab.

The following relevant properties are available:

- *Cookies*
  - *Session Cookie*
    Set the value of this property to "Yes", if the authentication should only be valid during a user session.
  - *Authentication Expires After Minutes*
    Define after how many minutes the authentication of a user expires, who is actively working with the system.
  - *Authentication of an Idle Session Expires After Minutes*
    Define after how many minutes the authentication of a user expires, who is not actively working with the system.

7 Authentication With SAML

SAML (Security Assertion Markup Language) is an XML-based standard for exchanging authentication and authorization data between an identity provider (IdP) and a service provider (SP). Whereas Fabasoft Folio is the service provider and Shibboleth - an open source SAML implementation - is used as identity provider. The following chapters describe how to install and configure the Shibboleth IdP and how to configure Fabasoft Folio to use single sign-on authentication with SAML.

7.1 Shibboleth Identity Provider

The chosen identity provider is Shibboleth, deployed on Apache Tomcat.

The installer of Shibboleth is available for download at http://shibboleth.internet2.edu/downloads/shibboleth/idp/. Apache Tomcat is available at http://tomcat.apache.org/download-55.cgi.

The Shibboleth IdP is a standard Java web application based on the Servlet 2.4 specification and can be deployed on several servlet containers. For Apache Tomcat, JBoss Tomcat, or Weblogic, detailed documentation about installing and deploying Shibboleth on them is available at https://spaces.internet2.edu/display/SHIB2/IdPInstall.

However, for using the Shibboleth IdP as an identity provider for Fabasoft Folio, Apache Tomcat is recommended. The installation tasks for the Shibboleth IdP described in this document are referred to Apache Tomcat as web server.

7.1.1 Installation Steps

Shibboleth cannot be installed and run by using the GNU Java compiler and VM. For using the IdP another Java virtual machine and compiler must be installed. The recommended JVM is the Sun HotSpot. This fact must be kept in perspective for deploying Shibboleth on operating systems that are using OpenJDK by default (e.g. Red Hat based Linux distributions).

After assuring that a proper JVM is available, the Shibboleth IdP can be installed by performing the following steps:

1. Download the IdP software package (the shibbolethidentityprovider-*-bin.zip file).
2. Unzip the downloaded archive.
3. Prepare the servlet container (Apache Tomcat, JBoss Tomcat or Weblogic).
4. Change into the newly created IdP distribution directory.
5. Run `ant.bat` on windows or `ant.sh` on Linux. This script will ask for the location of the identity provider's home directory that it will create. The user executing this script must have the permissions to create this directory. The output of the script may contain the message "JARs are never empty, they contain at least a manifest file", but this does not indicate a problem.
6. Deploy the IdP WAR file, located in `IDP_HOME/war/`, after you have properly prepared your servlet container.

7.1.2 The Identity Provider Directory Structure

The installation scripts create the identity provider's home directory which contains the following subdirectories:

- `bin/`
  Contains various tools for running, testing, and deploying the IdP.
- `conf/`
  Contains all the configuration files for the IdP.
- `credentials/`
  This is where the IdP's signing and encryption credentials (`idp.key`, and `idp.crt`) are stored.
- `lib/`
  Contains various code libraries used by the tools in `bin/`.
- `logs/`
  Contains the log files of the IdP.
- `metadata/`
  Contains the metadata of the IdP, in a file called `idp-metadata.xml`. It is recommended to store any other retrieved metadata here as well.
- `war/`
  Contains the web application archive (war) file that needs to be deployed into the servlet container.

7.1.3 Shibboleth Configuration

The Shibboleth configuration files are located in the folder `<Shibboleth Home>/conf`. For using the Shibboleth IDP with Fabasoft Folio, the default configuration files `relying-party.xml`, `attributefilter.xml`, `attribute-resolver.xml` and `metadata.xml` have to be adapted. The necessary modifications are described in the next chapters.

7.1.4 Tomcat Configuration

Detailed configuration of Apache Tomcat for hosting Shibboleth can be found at https://spaces.internet2.edu/display/SHIB2/IdPApacheTomcatPrepare.

The main configuration steps are:

- Copy the .jar files included in the identity provider's source endorsed directory into Apache Tomcat's `$CATALINA_HOME/common/endorsed` directory.

Note: Some versions of Apache Tomcat ship with some XML related JARs in this directory. If you see JAR files related to JAXP or XML, please remove these as well as adding the aforementioned Xerces and Xalan JARs.

- Install Shibboleth security provider
  - Copy the library `shibboleth-jce-1.0.0.jar`, located in the identity provider source's lib directory into `JAVA_HOME/jre/lib/ext`. If this directory does not exist it has to be created.
  - Edit the `java.security` file, located in the JRE's `lib/security` directory by adding the following line after the last `security.provider` entry:
    `security.provider.#=edu.internet2.middleware.shibboleth.DelegateToApplicationProvider`
    "#" is the number of the previous provider in the list incremented by one.
- Configure the Apache Tomcat connector for Shibboleth:

```
<Connector port="8443"
maxHttpHeaderSize="8192"
maxSpareThreads="75"
scheme="https"
secure="true"
clientAuth="want"
SSLEnabled="true"
sslProtocol="TLS"
keystoreFile="IDP_HOME/credentials/IdentityProvider.jks"
keystorePass="PASSWORD"
trustStoreFile="IDP_HOME/credentials/IdentityProvider.jks"
trustStorePass="PASSWORD"
trustStoreAlgorithm="DelegateToApplication"
/>
```

`IDP_HOME` is the Shibboleth home directory and `PASSWORD` is the password for the identity provider's keystore, which was specified during the installation process.

Possible installation issue on Microsoft Windows platforms:

o.s.web.context.ContextLoader – Context initialization failed at the first start.

The stack trace of this error probably contains an UnknownHostException for a host "c". This is actually the "c" from the IdP home directory path (e.g. `C:\Shibboleth-2.1`). The error is caused by wrong path expressions in `SHIBSRC\identityprovider\resources\WEB-INF\web.xml`. Replace every occurrence of "`file://$IDP_HOME$`" to "`file:///$IDP_HOME`" in the path expressions. Then the installer (`ant.bat` on Microsoft Windows) has to be run again with "not new installation" option. The created `idp.war` has to be deployed again.

7.1.5 Configuring Relying Parties

For interaction of the identity provider with Fabasoft Folio, the service provider must provide the standardized `metadata.xml` document. This file contains so called entity descriptors, which can be grouped in entity descriptor XML elements. The location of an entity descriptor can be defined by specifying a new metadata provider in the file `relying-party.xml`:

```
<MetadataProvider
  id="<id>"
  xsi:type="FileBackedHTTPMetadataProvider"
  xmlns="urn:mace:shibboleth:2.0:metadata"
  metadataURL="<folio_url>/saml2/metadata"
  backingFile="<backing_file_path>"
/>
```

In the example above, the type attribute of the `<MetadataProvider>` element is `FileBackedHTTPMetadataProvider`, which specifies that the entity descriptor can be downloaded via HTTP and the service provider's metadata gets locally cached in the identity provider's file system.

For each `MetadataProvider`, `<id>` is a unique string. The attribute `metadataURL` is the URL where the metadata can be downloaded whereas `<folio_url>` is the base URL of the Fabasoft Folio service. The attribute `backingFile` specifies the file system location where the metadata gets cached. To reload the specified metadata, the identity provider has to be restarted.

Also some attributes of the XML element `DefaultRelyingParty` have to be adjusted to set some default values for the IDP like shown in the following:

```
<DefaultRelyingParty  provider="<idp_entity_id>"
  defaultSigningCredentialRef="IdPCredential"
defaultAuthenticationMethod="urn:oasis:names:tc:SAML:2.0:ac:classes:PasswordProtec
tedTransport">
...
<ProfileConfiguration xsi:type="saml:SAML2SSOProfile"
  includeAttributeStatement="true"
  assertionLifetime="300000"
  assertionProxyCount="0"
  signResponses="conditional"
  signAssertions="never"
  encryptAssertions="never"
  encryptNameIds="never" />
...
</DefaultRelyingParty>
```

The attribute `provider` is set to `<idp_entity_id>` which is the `entity-id` of the used IdP. To enable password authentication, the attribute `defaultAuthenticationMethod` must be set to `urn:oasis:names:tc:SAML:2.0:ac:classes:PasswordProtectedTransport`. Because assertions should not be encrypted, it is also mandatory to set `encryptAssertions` to `never`.

7.1.6 Using Principal Name for Assertion Subject

By default the subject of the assertions generated by the identity provider is the transient id of the authenticated user. For single sign-on with Fabasoft Folio it is necessary that the principal of the user is contained in the assertion. This can be accomplished by performing changes in the `attribute-resolver.xml`, respectively `attributefilter.xml` file. In the following, an example configuration is given where LDAP is used as authentication backend and the user principal name gets embedded in the `NameID` tag of the assertion. This example includes attribute encoders which entails that the name identification gets sent the `NameID` tag of the response and additionally in an attribute with the name `urn:oid:1.2.840.113556.1.4.656`.

Note: The `AttributeDefinition` identifier `id` must be unique in the attribute configuration.

```
<resolver:AttributeDefinition
  id="userPrincipalName"
  xsi:type="PrincipalName"
  xmlns="urn:mace:shibboleth:2.0:resolver:ad"
  sourceAttributeID="userPrincipalName">
  <resolver:Dependency ref="myLDAP" />

  <!-- for including attribute urn:oid:1.2.840.113556.1.4.656 in the response -->

  <resolver:AttributeEncoder xsi:type="SAML1String"
    xmlns="urn:mace:shibboleth:2.0:attribute:encoder"
    name="urn:mace:dir:attribute-def:userPrincipalName" />
```

```
  <resolver:AttributeEncoder xsi:type="SAML2String"
    xmlns="urn:mace:shibboleth:2.0:attribute:encoder"
    name="urn:oid:1.2.840.113556.1.4.656"
    friendlyName="userPrincipalName" />

  <!-- for including the persistent NameID in the response -->

  <resolver:AttributeEncoder
    xsi:type="SAML1StringNameIdentifier"
    xmlns="urn:mace:shibboleth:2.0:attribute:encoder"
    nameFormat="urn:mace:shibboleth:1.0:nameIdentifier" />

  <resolver:AttributeEncoder xsi:type="SAML2StringNameID"
    xmlns="urn:mace:shibboleth:2.0:attribute:encoder"
    nameFormat="urn:oasis:names:tc:SAML:2.0:nameid-format:persistent" />

</resolver:AttributeDefinition>

<resolver:DataConnector
  id="myLDAP"
  xsi:type="LDAPDirectory"
  xmlns="urn:mace:shibboleth:2.0:resolver:dc"
  ldapURL="ldap://sqd1.sq.fabasoft.com"
  baseDN="ou=Setup-Test
  Accounts,ou=SQ,dc=sq,dc=fabasoft,dc=com"
  principal="DOMAINUSER" principalCredential="CREDENTIALS">
  <FilterTemplate>
    <![CDATA[
      (userPrincipalName=${requestContext.principalName})
      ]]>
  </FilterTemplate>
</resolver:DataConnector>


<AttributeFilterPolicy id="userPrincipalNameToAnyone">
  <PolicyRequirementRule xsi:type="basic:ANY" />
  <AttributeRule attributeID="userPrincipalName">
   <PermitValueRule xsi:type="basic:ANY" />
  </AttributeRule>
</AttributeFilterPolicy>
```

7.2 Configuration

The following settings are necessary for the configuration of SAML:

1. Open the *Virtual Application Configuration*, which is referenced in the *Current Domain* or *Domain Type*.
2. Click the „Authentication" tab.

The following relevant properties are available:

- *Cookies*
  - *Session Cookie*
    Set the value of this property to "Yes", if the authentication should only be valid during a user session.
  - *Authentication Expires After Minutes*
    Define after how many minutes the authentication of a user expires, who is actively working with the system.
  - *Authentication of an Idle Session Expires After Minutes*
    Define after how many minutes the authentication of a user expires, who is not actively working with the system.

- *Activate SAML*
  This property defines whether SAML is activated.
- *Identity Provider Metadata URL*
  This property defines the URL, where the IdP's metadata can be accessed. If defined, Fabasoft Folio attempts to download and update the IdP metadata on startup and ignores the *Identity Provider Metadata* property.
- *Identity Provider Metadata*
  A file containing the metadata can be uploaded. If the *Identity Provider Metadata URL* is not defined, the IdP metadata is initialized based on the provided XML file. For retrieving the IdP metadata, access the IdP metadata URL (Shibboleth: `https://myserver:8443/idp/shibboleth`) with a browser and download the file. In case of Shibboleth, the metadata can also be retrieved from the Shibboleth metadata directory (`metadata/idp-metadata.xml`).
  This property is particularly useful for the initial configuration when both parties, Fabasoft Folio and the IdP, require each other's metadata and the metadata URLs are inaccessible until the metadata is available on both sides.
- *Require Encrypted Assertions*
  This property defines that only encrypted SAML assertions are accepted.
  To use this feature, a private key must be configured.
- *Private Key Path (.p12, Assertion Encryption)*
  This property defines the file system path of the PKCS #12 private key used for SAML assertion encryption.
- *Private Key Password (Assertion Encryption)*
  This property defines the password of the private key used for SAML assertion encryption.
- *Enable Single Logout*
  This property defines whether the single logout feature is enabled.
  To use this feature, a private key must be configured.
- *Sign AuthnRequests*
  This property defines whether AuthnRequests are signed.
  To use this feature, a private key must be configured.
- *Signature Algorithm*
  This property defines the algorithm use to sign AuthnRequests and Single Logout SOAP calls. If not configured RSA-SHA1 is used.
- *Name ID Format for User Identification*
  Depending on the configuration of the identity provider, the name ID format can be specified here if the user identification should be contained in the `NameID` tag.
  Example: `urn:oasis:names:tc:SAML:2.0:nameid-format:persistent`
- *SAML Attribute Name for User Identification*
  If the user identification should be done with an assertion attribute, the name of the assertion attribute has to be specified here.
- *SAML Attribute Name for Authentication Details*
  If additional information provided in an assertion attribute should be logged in *Account Activity* in the *Account* portal as a result of successful authentication, the name of the assertion attribute has to be specified here. The value is stored in the property *Authentication Details* (`COOSYSTEM@1.1:siauthdetails`).

- *Alternative Base URL*
  Usually the base URL can be determined through the initial request. However, in some situations this might not be possible (i.e. if a load balancer is used). Therefore, the base URL can be specified in this field with the format `PROTO://HOST`, where `PROTO` is either http or https and `HOST` is the hostname respectively the domain name of the service provider.
- *Mappings for Authentication Methods*
  By default, the authentication method recorded by the Fabasoft Folio webservice is "SAML". The method can be changed based on the authentication method contained in the SAML assertion.
  Example:
    - *SAML Authentication Method*
      urn:left:rfc:1510
    - *Authentication Method*
      Kerberos (SPNEGO)
  - *SAML Authentication Method*
    Specifies the SAML authentication method contained in the SAML assertion (`/samlp:Response/saml:Assertion/saml:AuthnStatement/saml:AuthnContext/saml:AuthnContextClassRef`).
  - *Authentication Method*
    Specifies the Fabasoft Folio authentication method the SAML authentication method should be mapped to.
    By doing so, the Fabasoft Folio account activity contains the correct authentication method.
- Additionally, the user's security clearance can be extended based on the current authentication method by configuring a specific security class for a given authentication method.
  Create an entry in the *Security Clearances for Authentication Methods* field and edit the properties.
  - *Authentication Method*
    Specifies the Fabasoft Folio authentication method a security clearance is configured for.
  - *Security Clearance*
    Specifies an additional security class to be added to the security clearance of the authenticated user.

Providing credentials for cookie signing

For signing the cookie, the path to a file containing the `pkcs12` credentials must be configured. This also can be done on the "Authentication" tab via the *Private Key Path* field. If the key is password protected, also the password has to be defined.

Furthermore the following properties can be specified:

- *Session Cookie*
  Set the value of this property to "Yes", if the authentication should only be valid during a user session.
- *Cookie Domain*
  Set the value of this property to set the scope of the session cookie to a specific domain (e.g. "myserver.com") or subdomains of a domain (e.g. ".myserver.com").
- *Authentication Expires After Minutes*
  Define after how much minutes the authentication of a user expires, who is actively working

with the system.
Note: Specify a value greater than 0. Otherwise the authentication will expire immediately.

- *Authentication of an Idle Session Expires After Minutes*
  Define after how much minutes the authentication of a user expires, who is not actively working with the system.
  Note: Let this property unspecified or specify the value 0 if idle sessions should not expire. In this case the authentication will expire after the time specified in the previous property.

Configuring proxy settings

If a proxy is used, the configuration can be defined in this section.

- *Proxy Host*
  The host on which the proxy is running.
- *Proxy User*
  The user to be used for the proxy.
- *Proxy Password*
  The proxy user password.

Optional: Setting the environment variable FSCVEXT_ALTAUTHMETH

If LDAP authentication should be used as the alternative authentication method while using SAML as the primary one, the environment variable `FSCVEXT_ALTAUTHMETH` has to be set. This alternative authentication method will only be used if the username and password are already attached to the HTTP request header. However the web service will not send a challenge response if the HTTP request headers does not exist.

Additionally a regular expression can be defined in the environment variable `FSCVEXT_ALTAUTHUSERAGENT` which will be applied to the HTTP header variable `User-Agent`. If the regular expression matches and the HTTP request does not contain the username and password in the request header, a challenge response will be sent to the client. So it is possible to use LDAP authentication for defined user agents.

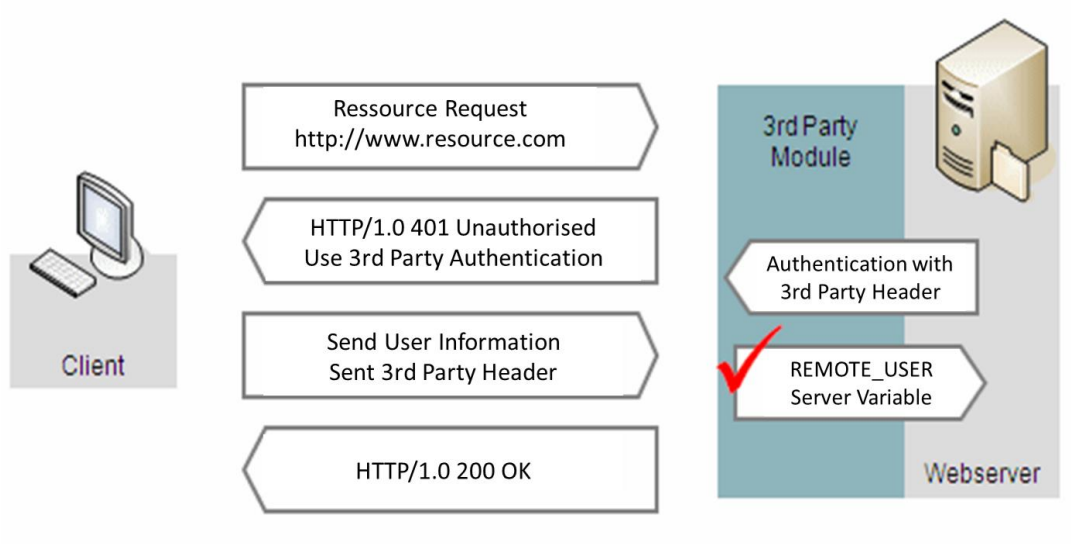
8 Authentication With External (REMOTE_USER)

The following chapters describe an authentication method using the server variable `REMOTE_USER`.

8.1 Authentication “External (REMOTE_USER)”

The *Authentication Method* “External (REMOTE_USER)“ that can be selected in the *Virtual Application Configuration* is based on the principle that a third party product module handles the authentication of a user on the Fabasoft Folio web server and that the information is passed on to the application in the server variable `REMOTE_USER`.

Using the content of the server variable `REMOTE_USER`, under the operating system account of the Fabasoft Folio Web Service the web server module `fscvext` logs on as the specified user.



8.2 Requirements for a Successful Login

There are following requirements for a successful login:

1. The third party product module has successfully passed on the request as authenticated and has set the server variable `REMOTE_USER`.
2. The user assigned to the operating system account of the Fabasoft Folio Web Service has to be entered in the *Privileged Users* field of the *Current Domain* to be able to log on as another user.
3. The content of the server variable `REMOTE_USER` has to match the login name of an active user in the Fabasoft Folio Domain.
4. If the server variable `REMOTE_USER` is empty, the operating system account is used to log on.

8.3 Type of Authentication, Re-Authentication and Signing

The third party product module on the Fabasoft Folio web server is not restricted to a certain type of authentication (regarding e.g. type of header variables or cookies) and is determined by the project context. The content of the server variable `REMOTE_USER` is a blind trust (principle of delegation). There is no interface that enables a re-authentication of the user name that was passed. In this respect it is not possible to sign with entry of the password.

8.4 Server Variable REMOTE_USER in the HTTP Header

The server variable `REMOTE_USER` cannot be passed in a HTTP header. However, it is possible that a third party product module copies a defined header variable to the server variable `REMOTE_USER` if the infrastructure guarantees security for this unsecure practice (e.g. because the Fabasoft Folio web server is only available via determined proxy server that enables secure authentication).

9 Authentication With OAuth

The following chapters describe the configuration of OAuth. This service can be used to authenticate at a Fabasoft Folio Web Service.

9.1 General

OAuth is an open standard for authentication and access control. It allows users to share their private resources stored on one site with another site without having to hand out their credentials, typically supplying username and password tokens instead. Further information can be found on the web (http://oauth.net/ and http://oauth.net/2/).

The implementation is based on OAuth 2.0 Draft 25 and Bearer Tokens Draft 18. The grant type "authorization code" is supported. Refresh tokens are not supported.

9.2 Preparation

9.2.1 Web Service Configuration

To be able to log in using an OAuth access token, the Fabasoft Folio Web Service must be configured to require SAML authentication.

9.2.2 Client Configuration

An OAuth client requires a client identifier and a client secret for being able to request an OAuth access token. These variables can be allocated by creating an OAuth Client object (`FSCOWS@1.1001:OAuthClient`). An OAuth client has the following properties:

- *Name*
  Defines a name that represents the client and its use cases for accessing the service via OAuth.
- *Client ID*
  Represents the OAuth Client ID to use for OAuth requests.
- *Client Secret*
  Represents the OAuth Client Secret to use for OAuth token requests.
- *Redirect URI*
  Defines an optional redirect URI. If specified, the service will redirect to the URI in the context of authorization requests. If an authorization request specifies a redirect URI and a redirect URI is configured, the values must match.
- *OAuth Token Expires After*
  Defines the expiration time of an OAuth access token issued for the client. If not defined, the expiration time specified in the domain configuration is used.
- *Allowed Web Services*
  Defines a list of web services (`FSCOWS@1.1001:WebServiceDefinition`) that can be accessed via OAuth for this client. The scope specified in an OAuth authorization request must be contained in this list, otherwise the request will be denied. If not specified, OAuth tokens cannot be requested or used for this client. If valid tokens exist at this point, these tokens cannot be used for as long as the scope is missing from this list.

9.2.3 Access Configuration

In the scope argument of an OAuth authorization request, clients can specify the desired access for a token. In case of Fabasoft Folio, a scope is represented by a list of full references of *Web Service Definition* (`FSCOWS@1.1001:WebServiceDefinition`) instances separated by a space character.

A *Web Service Definition* contains a list of actions that represent web service implementations (SOAP, JSON, Friendly URLs).

The following well-known web service instances are available for use with OAuth:

- WebDAV (`FSCOWS@1.1001:WebDAVWebService`)
- CMIS (`FSCCMIS@1.1001:CMISWebService`)

The following properties are available:

- *Multilingual Name*
  A list of language-specific names presented to the user during access confirmation. In the context of OAuth, the multilingual name represents a permission that can be granted as part of the confirmation process.
- *Web Service Actions*
  This property defines a list of actions (`COOSYSTEM@1.1:Action`) of web service implementations (SOAP, JSON, FriendlyURL). In the context of OAuth, the list defines which web service implementations can be accessed via OAuth, if the OAuth scope contains the full reference of the *Web Service Definition*. An OAuth client can only specify *Web Service Definitions* that are explicitly allowed in the *OAuth Client* instance.

9.3 Configuration

The following settings are available for the configuration of OAuth:

1. Open the *Web Service Configuration*, which is referenced in the *Current Domain* or *Domain Type*.
2. Click the „OAuth" tab.
   The following properties are available:
   - *OAuth Code Expires After Minutes*
     Defines the expiration time of an OAuth authorization code in minutes (default: 10).
   - *OAuth Token Expires After*
     Defines the expiration time of an OAuth access token (default: "1 Day").
   - *Trusted OAuth Clients*
     Defines a list of trusted OAuth clients. In case of a trusted client, the access confirmation user interface is skipped.

OAuth can only be enabled as an alternative authentication method in combination with SAML. To enable OAuth, the environment variable `FSCVEXT_ALTAUTHMETH` must be set for a host or web service.

9.4 Client Example

This sample is based on Apache Amber, an OAuth2 client for Java, and should serve as a guide for getting started with the Fabasoft Folio OAuth implementation.

9.4.1 Configuration

Before being able to execute the sample, static variables must be initialized as follows:

```
private static final String BASEURI = "https://<hostname>/fsc";
```

Specify the Fabasoft Folio base URL comprised of URL scheme, hostname, port and virtual directory.

```
private static final String CLIENTID = "<client-id>";
```

Specify the value of the property *Client ID* of an OAuth Client (`FSCOWS@1.1001:OAuthClient`) instance in the target domain.

```
private static final String CLIENTSECRET = "<client-secret>";
```

Specify the value of the property *Client Secret* of an OAuth Client (`FSCOWS@1.1001:OAuthClient`) instance in the target domain.

```
private static final String REDIRECTURI = "http://localhost/";
```

The redirect URI can remain unchanged for the sample. The redirect URI specifies, which URL the Fabasoft Folio OAuth implementation redirects to if the user confirms requested access permissions.

```
private static final String SCOPE = "<wsdef-full-reference>";
```

Specify the full reference of one or more Web Service Definition (`FSCOWS@1.1001:WebServiceDefinition`) objects, separated by the space character.

```
// Object address of a NOTE@1.1:NoteObject instance
private static final String TESTOBJECT = "<noteobject-address>";
```

Specify the object address of a Note Object (`NOTE@1.1:NoteObject`) in your domain

9.4.2 Compilation

The sample depends on Apache Amber (http://incubator.apache.org/amber/) as well as Codehaus Jettison (http://jettison.codehaus.org/).

Download the latest snapshot of oauth2-common.jar and oauth2-client.jar from

https://repository.apache.org/content/groups/snapshots/org/apache/amber/oauth2-common/

https://repository.apache.org/content/groups/snapshots/org/apache/amber/oauth2-client

Download the latest binary distribution containing jettision.jar from http://jettison.codehaus.org/.

Create a file OAuthTest.java, copy the contents of 9.4.4 into the file, and compile the sample as follows:

Windows

```
set CLASSPATH=.;jettison.jar;oauth2-common.jar;oauth2-client-jar
javac OAuthTest.java
```

Linux

```
export CLASSPATH=.:jettison.jar:oauth2-common.jar:oauth2-client-jar
javac OAuthTest.java
```

9.4.3 Execution

Execute the sample as follows:

Windows

```
set CLASSPATH=.;jettison.jar;oauth2-common.jar;oauth2-client-jar

java OAuthTest
```

Linux

```
export CLASSPATH=.:jettison.jar:oauth2-common.jar:oauth2-client-jar

java OAuthTest
```

The sample prints an URL that must be accessed via a GUI browser. The query argument `code` of the URL the browser redirects to contains the OAuth authorization code. Copy the value and paste it into the console prompt. The application then requests a token based on the authorization code and attempts to access the configured note object using the token:

```
Confirm access via the following URL:

http://localhost/fsc/oauth2/authorize?scope=TEST%401.506%3AReadContents&redirect_u
ri=http%3A%2F%2Flocalhost%2F&client_id=COO.1.506.3.1001983&response_type=code

Enter 'code' query argument value: COO.1.506.1.15-0211a5cfcf00314daa3e0bdc73865257
Token:      COO.1.506.1.15-cc1166d3fb7ddc409e8b0d0d1dad88e9
Expiration: 86400 seconds
Data:

Network Working Group                                          E. Hammer, Ed.
Internet-Draft
Obsoletes: 5849 (if approved)                                     D. Recordon
Intended status: Standards Track                                     Facebook
Expires: November 2, 2012                                           D. Hardt
                                                                    Microsoft
                                                                   May 1, 2012

                The OAuth 2.0 Authorization Framework
                        draft-ietf-oauth-v2-25
```

9.4.4 Code

OAuthTest.java:

```
import java.io.*;
import java.net.*;
import org.apache.amber.oauth2.client.*;
import org.apache.amber.oauth2.common.exception.*;
import org.apache.amber.oauth2.common.message.types.*;
import org.apache.amber.oauth2.client.request.*;
import org.apache.amber.oauth2.client.response.*;

public class OAuthTest
{
  // Fabasoft Folio base URL (e.g. http://localhost/fsc)
  private static final String BASEURI = "https://<hostname>/fsc";
  // Client ID as defined by FSCOWS@1.1001:OAuthClient instance
  private static final String CLIENTID = "<client-id>";
  // Client secret as defined by FSCOWS@1.1001:OAuthClient instance
  private static final String CLIENTSECRET = "<client-secret>";
  // Arbitrary redirect URI required for the protocol
  private static final String REDIRECTURI = "http://localhost/";
  // Full reference of an FSCOWS@1.1001:WebServiceDefinition instance
  // containing the action FSCASP@1.1001:ReadContentFriendlyURL
  private static final String SCOPE = "<wsdef-full-reference>";
  // Object address of a NOTE@1.1:NoteObject instance
  private static final String TESTOBJECT = "<noteobject-address>";

  public static void main(String[] args) throws IOException, OAuthSystemException
  {
    try {

      //
      // Build authorization code request
      //
      OAuthClientRequest request = OAuthClientRequest
        .authorizationLocation(BASEURI + "/oauth2/authorize")
        .setClientId(CLIENTID)
        .setScope(SCOPE)
        .setRedirectURI(REDIRECTURI)
        .setResponseType("code")
        .buildQueryMessage();
```

```
      System.out.println("Confirm access via the following URL:");
      System.out.println();
      System.out.println(request.getLocationUri());
      System.out.println();
      System.out.print("Enter 'code' query argument value: ");
      BufferedReader reader =
        new BufferedReader(new InputStreamReader(System.in));
      String code = reader.readLine();

      //
      // Build and send token request
      //
      request = OAuthClientRequest
        .tokenLocation(BASEURI + "/oauth2/token")
        .setGrantType(GrantType.AUTHORIZATION_CODE)
        .setClientId(CLIENTID)
        .setClientSecret(CLIENTSECRET)
        .setRedirectURI(REDIRECTURI)
        .setCode(code)
        .buildBodyMessage();

      OAuthClient client = new OAuthClient(new URLConnectionClient());
      OAuthAccessTokenResponse response = client.accessToken(request);
      String accesstoken = response.getAccessToken();
      System.out.println("Token:      " + accesstoken);
      System.out.println("Expiration: " + response.getExpiresIn() + " seconds");

      //
      // Access resource
      //
      URL url = new URL(BASEURI + "/read/" + TESTOBJECT);
      URLConnection conn = url.openConnection();
      conn.setRequestProperty("Authorization", "Bearer " + accesstoken);
      System.out.println("Data:");
      System.out.println();
      reader = new BufferedReader(new InputStreamReader(conn.getInputStream()));
      String line = reader.readLine();
      while (line != null) {
        System.out.println(line);
        line = reader.readLine();
      }
    }
    catch (OAuthProblemException e)
    {
      System.out.println("OAuth error:             " + e.getError());
      System.out.println("OAuth error description: " + e.getDescription());
    }
  }
}
```