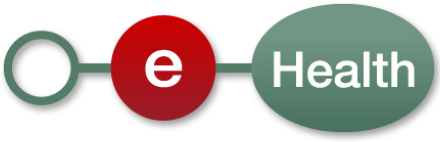

Cookbook
Identity & Authorization Management (I.AM)
SP Shibboleth
Version 1.0

This document is provided to you free of charge by

The eHealth platform

Willebroekkaai 38 – Quai de Willebroeck 38

1000 BRUSSELS

Table of contents

1 Document management

1.1 Document history

Version	Date	Author	Description of changes / remarks
1.0	11/07/2013	eHealth	Initial version

2 Introduction

2.1 Goal of the service

The eHealth Identity Provider (IDP), one of the key components of the eHealth I.AM Federation, is the Identity Service of eHealth that provides Cross Enterprise Web Browser Single Sign On.

The IDP supports multiple authenticationmethods, provides the possibility to choose from multiple principals that represent different aspects of one's identity and facilitates integration for a wide range of partner systems by supporting multiple protocols and profiles widely used in the international community.

Using the Shibboleth SP Software, you can easily integrate with the IDP using one of these protocols and profiles.

2.2 Goal of the document

This document describes how to install and configure a Shibboleth SP (Service Provider) to protect a webapplication and integrate it in the eHealth I.AM Federation.

Installation, basic configuration and launching is described in step by step instructions.

A HOWTO section explains additional configuration thay you might find useful.

For information on Shibboleth itself, see the website at http://shibboleth.net/.

There is also a WIKI, maintained by the Shibboleth team, which contains lots of useful information on installation, configuration of Shibboleth components, security advisories and much more.

We strongly advise you to take a look at the WIKI and read the sections that concern Shibboleth Service Provider components.

You can find it at https://wiki.shibboleth.net/confluence/display/SHIB2/Home.

The Shibboleth community also maintains a few mailing lists, free of charge and actively used.
If you run into problems, we sugges you post your question to the 'Users' mailing list.
You can subscribe to the lists on the Shibboleth site at http://shibboleth.net/community/lists.html

2.3 eHealth document references

All the document references can be found in the support section of the eHealth portal[1]. These versions or any following versions can be used for the eHealth service.

ID	Title	Version	Date	Author
1	Glossary.pdf	1.0	01/01/2013	eHealth

[1] www.ehealth.fgov.be

2.4 External document references

All documents can be found through the internet. They are available to the public, but not supported by eHealth.

ID	Title	Source	Date	Author
1			DD/MM/YYYY	

2.5 Service history

This chapter contains the list of changes made to the service with respect to the previous version.

Previous version	Previous release date	changes
none	DD/MM/YYYY	

Remark: = “None” when the major version = 1

3 Business and privacy requirements

3.1 Certificates

An eHealth certificate is not always necessary between a (Shibboleth) SP and (eHealth) IDP.
It depends on the used SSO Profile.
See document 'eHealth I.AM - IDP' for an overview of the supported SSO Profiles.

An eHealth certificate is used to identify the initiator of the request. If you don't have one, see:

Dutch version:
https://www.ehealth.fgov.be/nl/page_menu/website/home/platform/basicservices/certificates.html

French version:
https://www.ehealth.fgov.be/fr/page_menu/website/home/platform/basicservices/certificates.html

3.2 eHealth contact

eHealth ContactCenter:

02 / 788 51 55

- Mail: support@ehealth.fgov.be
- Web form:
  - Dutch version:
    https://www.ehealth.fgov.be/nl/contactform/website/home/contactform.html
  - French version:
    https://www.ehealth.fgov.be/fr/contactform/website/home/contactform.html

4 Global overview

The eHealth I.AM Identity Provider (IDP) provides Identity and Authorization Information to Service Providers (SP) on behalf of users who are authenticated at the IDP and wish to get access to a service hosted at the SP.

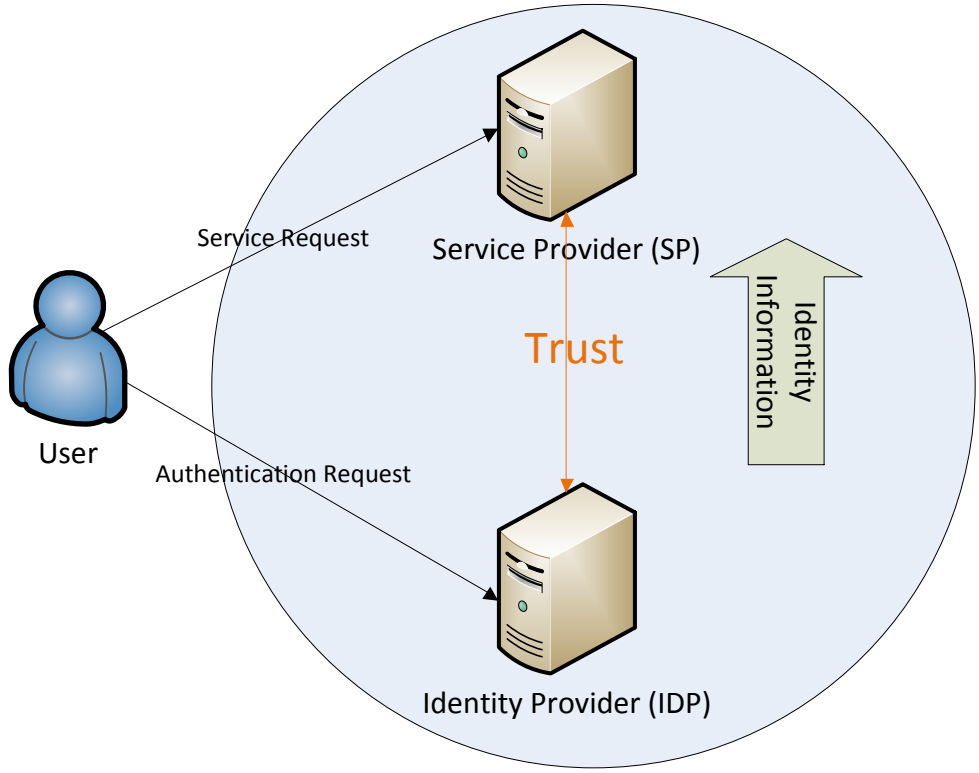

Figure 1 : Single Sign-On

5 Technical specifications

5.1 Introduction

Shibboleth is an open-source project that provides Single Sign-On capabilities and allows sites to make informed authorization decisions for individual access of protected online resources in a privacy-preserving manner.

The project consists of multiple products, free of charge. One of them is the Service Provider (SP). This component implements all of the SAML 1.1 and SAML 2.0 Web SSO profiles and is therefore a good product to be used in client systems that want to integrate with the eHealth I.AM Federation IDP which also supports those profiles.

5.2 Install

Note

As mentioned on the Shibboleth WIKI, we advise you to subscribe to the 'Shibboleth Announcement mailing list'[2]. This is where announcements about new releases, end-of-life of past releases, and security advisories are distributed.

To get started, a suitable version of the Shibboleth SP Software must be installed on a webserver in the infrastructure of the partner hosting the application to protect.

There are distributions for multiple operating systems and webservers.

Webservers

- Apache
- FastCGI Authenticator
- IIS

OS

- OpenSUSE 11+
- OS X 10.6+
- RedHat 5+
- SLES 10+
- Solaris 2.9+
- Windows XP/2003/2008

For a complete and up to date list, see the Shibboleth site[3].

For the purpose of this document, a Shibboleth SP (2.5) and an Apache HTTP Server (2.2) was installed on a Windows (XP) machine using the installer available on the Shibboleth site[4].

[2] http://shibboleth.net/community/lists.html

[3] http://shibboleth.net/products/service-provider.html

If your environment is different, you can still use this cookbook to configure Shibboleth after installation. For most parts, it will be the same, independent of your installation.

For the installation itself, we advise you to visit the website where you can find links to detailed installation instructions for every available distribution[5].

5.2.1 Verify setup

The Shibboleth SP distribution contains a few packages/libraries that are commonly used in other software as well, such as xerces-c, xml-security-c, xmltooling, …

Therefore, it is possible, your environment contains older versions of those packages.

When running the shibd daemon, please make sure it is loaded with the correct packages/libraries as distributed with the software.

5.2.2 Initial Testing

You can test to ensure that the SP is running properly and the surrounding environment is correct by accessing https://localhost/Shibboleth.sso/Status **from the actual web server machine**. You **MUST** use "localhost" as the hostname or it WILL NOT WORK by default. If this test is successful, then the software is ready for further configuration.

5.3 Configure

After installation was successful, some configuration files must be updated to protect your application and integrate with the eHealth I.AM Federation to receive identity and attribute information for your users from the Identity Provider (IDP).

The steps below give basic configuration instructions building on the installation as described in previous section.

Further in this document, you can find a '**Error! Reference source not found.**' section with additional configuration options.

On the Shibboleth WIKI you can find more detailed information. A good place to start is the 'NativeSPGettingStarted[6]' page.

5.3.1 Update Apache configuration

If you are not using Apache as your Webserver, skip this section and go to section 5.3.2 for configuration of the Shibboleth SP modules.

Before you do, make sure your webserver is supported by the Shibboleth software and look on the Shibboleth site for instructions on preparing your server for Shibboleth.

As mentioned on the Shibboleth WIKI, a few changes must be done to the Apache Webserver before it can work together with a Shibboleth SP.

Please read the part of the 'Basic Configuration' on the WIKI[7] before continuing.

This section contains an example configuration.

---

[4] https://wiki.shibboleth.net/confluence/display/SHIB2/NativeSPWindowsInstall

[5] https://wiki.shibboleth.net/confluence/display/SHIB2/Installation

[6] https://wiki.shibboleth.net/confluence/display/SHIB2/NativeSPGettingStarted

[7] https://wiki.shibboleth.net/confluence/display/SHIB2/NativeSPWindowsApacheInstaller

Text in red will need to be updated according to your personal configuration.

5.3.1.1 Update <apache>\conf\httpd.conf

- Include the Shibboleth SP configuration file for Apache (2.2).

```
# Include configuration for the Shibboleth module
Include C:\opt\shibboleth-sp\etc\shibboleth\apache22.config
```

- Ensure that SSL is enabled (your application should not be available over a plain HTTP connection).

```
LoadModule ssl_module modules/mod_ssl.so
...
# Secure (SSL/TLS) connections
Include conf/extra/httpd-ssl.conf
```

- Configure a proxy for the location(s) of your application(s)

If the application you want to protect is not in the DocumentRoot of your server, make sure a Proxy is configured for the path to the location of your application[8].

As this not only depends on the webserver you have installed but also on what is supported by the (application)server being used to run the webapplication on, there are no default instructions here.

An example can be found in the HOWTO 'Protect Java Servlets' using the *Apache JServ Protocol* (*AJP*).

5.3.1.2 Update <apache>\conf\extra\httpd-default.conf

- Ensure `UseCanonicalName` is set

This makes sure that the Apache server always uses the serverName as defined in the Apache configuration files (see following step) instead of relying on what the client entered in the browser. This is needed to avoid security leaks, especially if you plan to use the Shibboleth RequestMap[9] element to protect locations instead of Apache Location directives.

```
UseCanonicalName On
```

5.3.1.3 Update <apache>\conf\extra\httpd-ssl.conf

- Ensure that the ServerName directive is properly set

```
ServerName www.partner.be:443
```

- Ensure that the certificate and private key used to setup the SSL connection are set.

[8] If you don't know how to do this, you should contact your network administrator or check the documentation of your webserver (Apache: http://httpd.apache.org/docs/2.2/mod/mod_proxy.html) and applicationserver.

[9] https://wiki.shibboleth.net/confluence/display/SHIB2/NativeSPRequestMapHowTo

```
#   Server Certificate:
#   Point SSLCertificateFile at a PEM encoded certificate.  If
#   the certificate is encrypted, then you will be prompted for a
#   pass phrase.  Note that a kill -HUP will prompt again.  Keep
#   in mind that if you have both an RSA and a DSA certificate you
#   can configure both in parallel (to also allow the use of DSA
#   ciphers, etc.)
SSLCertificateFile "C:/opt/apache/conf/server.crt"


#   Server Private Key:
#   If the key is not combined with the certificate, use this
#   directive to point at the key file.  Keep in mind that if
#   you've both a RSA and a DSA private key you can configure
#   both in parallel (to also allow the use of DSA ciphers, etc.)
SSLCertificateKeyFile "C:/opt/apache/conf/server.pem"
```

5.3.1.4 Update <shib-sp>\etc\shibboleth\apache22.config

Configuration of this file is well explained on the Shibboleth WIKI[10].

Please read this section before continuing.

- Loading the Shib module

Ensure the Shib module is loaded in the configuration file.

```
LoadModule mod_shib c:/opt/shibboleth-sp/lib/shibboleth/mod_shib_22.so
```

- Route Location(s) for module mod_shib

Make sure the locations mentioned for the mod_shib module are handled by Shibboleth.

```
<Location /Shibboleth.sso>
        SetHandler shib
</Location>
```

The location(s) may vary depending on your configuration of the shibboleth2.xml file. See 'Update <shib-sp>\etc\shibboleth\shibboleth2.xml'.

- Activate Shibboleth as authenticator for your application(s)

If you are not using apache, we advise you to look on the Shibboleth WIKI[11] how to protect resources on the webserver of your choice.

[10] https://wiki.shibboleth.net/confluence/display/SHIB2/NativeSPApacheConfig

[11] https://wiki.shibboleth.net/confluence/display/SHIB2/NativeSPProtectContent

```
<Location /secure>
        AuthType shibboleth
        ShibRequestSetting requireSession On
        Require valid-user
</location>
```

Configuration of this part may vary.

A basic configuration could be as below:

- *<Location …>*: If the URL used by the client matches the Location, apache will apply the configuration inside the directive.
- *AuthType shibboleth*: Partially activates the shib_mod module when type is set to "shibboleth". Must be accompanied by a `Require` command. For more information, see the Apache documentation[12].
- *ShibRequestSetting* requireSession *On*: requires to activate a session when an unauthenticated user tries to access the location.
- *Require valid-user*: A rule that requires an authenticated session, but nothing else. No information of any kind about the user is required in order to satisfy this rule and it should NEVER be used in the absence of additional application logic to perform authorization.

To use more advanced configuration available in Apache or use the Shibboleth SP `<AccessControl>` plugin instead, see HOWTO 'Attach an Access Control Policy'.

5.3.2 Update Shibboleth SP configuration

To protect your application, updates must be done in a few xml files that were provided with the Shibboleth SP distribution you downloaded.

An overview of what can be configured can be found on the Shibboleth WIKI[13].

This section contains an example configuration.

Text in red will need to be updated according to your personal configuration.

5.3.2.1 Update <shib-sp>\etc\shibboleth\attribute-map.xml

Before transort, Identity information on your users will be encapsulated into SAML Attributes.

By default, a Shibboleth SP will remove all received attributes in order not to process unnecessary or even unwanted information.

Therefore, you must specify which attributes you need to receive in the backend.

You need to specify everyone of those attributes in the file attribute-map.xml.

[12] http://httpd.apache.org/docs/2.2/mod/core.html#authtype

[13] https://wiki.shibboleth.net/confluence/display/SHIB2/NativeSPConfigurationElements

All attributes received by the SP but not mentioned in this configuration file will be neglected by the SP and not available to your application.

If you are updating the default xml-file from the Shibboleth distribution, you can remove all of the ones present and add those known to the eHealth I.AM Federation that you expect to receive.

A complete list of eHealth Attributes will not be given here.

Below is just an example specifying two of the basic attributes that are always present in a successfull authentication request.

Please consult the separate document on eHealth attributes[14] for the list of attributes that you may receive in response to authentication requests for your application.

```
<Attributes xmlns="urn:mace:shibboleth:2.0:attribute-map"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <Attribute id="Shib-Authz-Decision" name="urn:be:fgov:ehealth:1.0:authz-decision"
nameFormat="environment" />
    <Attribute id="Shib-eHealth-Ref" name="urn:be:fgov:ehealth:1.0:ehealth-ref"
nameFormat="environment" />
</Attributes>
```

Note

- **Id**: You can choose the `id` attribute as you like. These ids will be used by the Shibboleth SP as names of the request headers or environment attributes (depends on what you configured to make the information available). You will be able to retrieve them after the Shibboleth SP has processed the response of the IDP. See 'Pass on identity information from the Shibboleth SP'.
- **Name**: URI of the attribute as received from the eHealth IDP. This must match the name of the attribute as defined in the eHealth I.AM Federation.
- **NameFormat**: The value of the nameFormat depends on whether you are using a SAML 1.1 or SAML 2.0 profile.
    - SAML 1.1: SAML 1.x Attributes come with namespaces. You'll need to set each nameFormat to the correct namespace of the specified eHealth attribute. See the separate document on eHealth attributes to know which namespace to specify for each attribute.
    - SAML 2.0: the nameFormat for any eHealth attribute will always be 'urn:oasis:names:tc:SAML:2.0:attrname-format:uri'. You can specify this or remove the nameFormat as it will be used as default.

More information on how to handle attributes can be found on the Shibboleth WIKI[15].

5.3.2.2 Update <shib-sp>\etc\shibboleth\attribute-policy.xml

By default, an attribute-policy.xml file is available in the Shibboleth distribution.

You can use this to filter out **values** of attributes (not to be mistaken with filtering out attributes themselves using attribute-map.xml) that your Shibboleth SP received from the IDP.

[14] Document 'eHealth I.AM - Federation Attributes'

[15] https://wiki.shibboleth.net/confluence/display/SHIB2/NativeSPAddAttribute

If you are using the default file from the distribution, you can clean up all the rules present as they do not apply to eHealth attributes and replace the contents with the code below.

```
<afp:AttributeFilterPolicyGroup
    xmlns="urn:mace:shibboleth:2.0:afp:mf:basic"
    xmlns:saml="urn:mace:shibboleth:2.0:afp:mf:saml"
    xmlns:basic="urn:mace:shibboleth:2.0:afp:mf:basic"
    xmlns:afp="urn:mace:shibboleth:2.0:afp"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

    <afp:AttributeFilterPolicy>
        <!-- This policy is in effect in all cases. -->
        <afp:PolicyRequirementRule xsi:type="ANY"/>

        <!-- Catch-all that passes everything else through unmolested. -->
        <afp:AttributeRule attributeID="*">
            <afp:PermitValueRule xsi:type="ANY"/>
        </afp:AttributeRule>

    </afp:AttributeFilterPolicy>

</afp:AttributeFilterPolicyGroup>
```

This policy does not filter any values from the received attributes (defined in the attribute-map.xml), thanks to the ANY xsi:types and the wildcard for the attributeID.

You can add policies and rules for specific attributes if you like.

See the Shibboleth WIKI[16] for more information.

If you don't want to filter anything, you can also configure your Shibboleth SP not to use this file at all.

See configuration of the shibboleth2.xml file below.

5.3.2.3 Update <shib-sp>\etc\shibboleth\shibboleth2.xml

This file is the main configuration file for the service provider. Together with the webserver config file 'apache22.config' (see previous section), it will protect your web resources.

The sections in this config file are listed below with a small description and code fragments if specific settings are required.

For a full description and examples, see the Shibboleth WIKI[17] or the HOWTO section for some specific settings.

- **OutOfProcess**

The OutOfProcess section contains properties affecting the shibd daemon.

```
<OutOfProcess logger="shibd.logger" />
```

Ensure the shibd.logger is set.

This text-based file can be altered to change log configuration.

See HOWTO 'Trace problems' for more information.

[16] https://wiki.shibboleth.net/confluence/display/SHIB2/NativeSPAttributeFilter

[17] https://wiki.shibboleth.net/confluence/display/SHIB2/NativeSPShibbolethXML

On Version 2.4 and above, the OutOfProcess can be omitted, and the default attribute values are used (and no extensions are loaded).

More information on the Shibboleth WIKI[18].

- **InProcess**

The InProcess section contains settings affecting web server modules.

Required for IIS, but can be removed when using other web servers.

More information on the Shibboleth WIKI[19].

- **Listener**

Configures the communication layer between the in-process and out-of-process components.

Can be omitted on Version 2.4 and above, in which case TCP (on Windows) or Unix (everywhere else) plugins will be used, with the default settings described on their respective topic pages.

More information on the Shibboleth WIKI[20].

- **StorageService**

Configures the storage of information that must persist across requests, like the SessionCache and ReplayCache.

Can be omitted on Version 2.4 and above, in which case the default in-memory plugin will be activated, for the Memory StorageService.

More information on the Shibboleth WIKI[21].

- **SessionCache**

Configures the caching of typically cookie-based sessions that bind attributes and SAML assertions for use by web requests.

On Version 2.4 and above, this element can be omitted, resulting in the "StorageService" cache being used, on top of the default StorageService, with other options defaulted.

More information on the Shibboleth WIKI[22].

- **ReplayCache**

Configures the caching of message identifiers for short periods to prevent replay attacks.

---

[18] https://wiki.shibboleth.net/confluence/display/SHIB2/NativeSPOutOfProcess

[19] https://wiki.shibboleth.net/confluence/display/SHIB2/NativeSPInProcess

[20] https://wiki.shibboleth.net/confluence/display/SHIB2/NativeSPListener

[21] https://wiki.shibboleth.net/confluence/display/SHIB2/NativeSPStorageService

[22] https://wiki.shibboleth.net/confluence/display/SHIB2/NativeSPSessionCache

On Version 2.4 and above, can be omitted, and a default/arbitrary StorageService will be used.

More information on the Shibboleth WIKI[23].

- **ArtifactMap**

Configures the short-term storage of XML messages bound to artifacts for communication to partner sites by reference.

On Version 2.4 and above, can be omitted and an in-memory version with default settings will be used.

More information on the Shibboleth WIKI[24].

- **RequestMapper**

Maps incoming web requests to configuration settings and the Application to associate with them. Also connects the portable configuration layer with server-specific mechanisms like .htaccess files.

Generally needed only with IIS as you can use Apache's Location Directive on Apache HTTPServer.

You can think of it as a portable equivalent of the Apache <Location> feature, which associates Apache directives with specific URLs, but with more functionalities that you might find useful.

On Version 2.4 and above, omitting this element will result in a "Native" plugin with an empty/default configuration. This empty configuration maps all requests to default application settings and adds no other settings, unless overridden by web server-specific options.

See HOWTO 'Use Apache Content Settings' to override these default settings.

See HOWTO 'Use the RequestMapper' if you plan to use it anyway.

- **ApplicationDefaults**

Provides for configuration of protocol behavior, session policy, error handling, and attribute handling. All requests into a server using Shibboleth are mapped by the RequestMapper, or web server commands, to the settings in the corresponding <ApplicationDefaults> or <ApplicationOverride> element for processing.

The default configuration of the <ApplicationDefaults> element and most of its subelements will work for most circumstances with a few modifications which are listed below.

For more detailed information on this section, please consult the Shibboleth WIKI[25].

- entityID

The entityID attribute on the <ApplicationDefaults> element needs to be changed to the SAML entityID you'd like to use for your deployment. This must be the value used on registration of the SP in the eHealth I.AM Federation.

[23] https://wiki.shibboleth.net/confluence/display/SHIB2/NativeSPReplayCache

[24] https://wiki.shibboleth.net/confluence/display/SHIB2/NativeSPArtifactMap

[25] https://wiki.shibboleth.net/confluence/display/SHIB2/NativeSPApplication

```
<ApplicationDefaults id="default" entityID="http://spname.hostname.be/shibboleth"
        REMOTE_USER="Shib-Organization-Name Shib-Person-FirstName"
        signing="back" encryption="false"
       connectTimeout="10" timeout="20">
```

Note:

- **Id**: MUST be set to "default" (or omitted on 2.4+).
- **EntityID**: must be set to the ID used by your Shibboleth SP to identify itself in the eHealth I.AM Federation. It must therefore be unique across SPs.
- **REMOTE_USER**: Specifies a list of attribute IDs/aliases to look for in a session's cache of attributes. The first one found with a value is set as REMOTE_USER[26] (or in the case of IIS, HTTP_REMOTE_USER). The attribute IDs must match those configured in attribute-map.xml (see previous section).
- **Signing**: Controls outbound signing of XML messages. If "true", all are signed. If "front", only front-channel messages are signed. If "back", only back-channel messages are signed. You MUST set "back" so eHealth IDP can identify your SP when the back-channel message is received at the eHealth IDP.
- **Encryption:** Controls outbound encryption of XML messages. You can leave it to the default, which is "false".
- **connectTimeout**: Specifies the timeout for connecting to remote servers during back-channel SOAP communication. Defaults to 10.
- **Timeout**: Specifies the total time to allow for completing back-channel SOAP communication. Defaults to 20.

➢ Sessions

Configures the session handling behavior for the application, as well as all of the supported processing handlers and their locations.

```
<Sessions checkAddress="false">
```

All attributes on the <Sessions> element are optional.

Default values should be fine in most cases except the checkAddress attribute.

It is best to disable this check to prevent false errors[27].

For a detailed explanation on Session Configuration, please consult the Shibboleth WIKI[28].

✯ SSO

Enables support for one or more Single Sign-On / Authentication protocols.

---

[26] REMOTE_USER is a special server or CGI variable used to pass the primary identifier of a browser user.

[27] The IdP will place the IP address of the user agent it authenticated into the assertions it issues. When `checkAddress="true"` (default), the SP will check this address against the address of the client presenting an assertion before creating a session. While useful for security, NAT and proxy usage (as well as IPv6 support on only either the webserver hosting the IdP or the SP) often make this setting a source of errors.

[28] https://wiki.shibboleth.net/confluence/display/SHIB2/NativeSPSessions

Make sure the <SSO> element (since version 2.4) is present and configured for integration with the eHealth IDP.

```
<SSO entityID="http://idp.smals-mvm.be/shibboleth">SAML2 SAML1</SSO>
```

Both the eHealth IDP and the Shibboleth SP Software support all SAML1.1 and SAML 2.0 Web Browser SSO profiles. As it is the SP that initiates the authentication session, it is the SP that chooses which profile is used. The SP will screen the IDP's metadata for supported profiles and the first match will be executed.

Adding "SAML2 SAML1" as value of the SSO element lets the SP know that it may use one of the SAML2 or SAML1 profiles it can find in both the IDP's metadata and the local file 'protocols.xml' (see 'ProtocolProvider').

More information on the Shibboleth WIKI[29].

- Logout

Enables support for one or more Logout protocols.

Only Local Logout is supported by the eHealth IDP so do not specify SAML2 as Logout initiator. It will result in an error when initiating logout at runtime.

```
<Logout>Local</Logout>
```

More information on the Shibboleth WIKI[30].

- SessionInitiator

Initiates sessions by creating a request for authentication specific to a particular SSO protocol. Generally superseded in 2.4+ by the <SSO> service element so if you use that one, you can remove all `<SessionInitiator>` elements.

The Shibboleth SP will choose itself the best available protocol and profile (See <SSO> section).

However, if you want complete control over what protocol and binding is used exactly, you should consider using SessionInitiators.

If you do so, it's best to replace the <SSO> element entirely to avoid unforeseen overlaps and interactions.

See HOWTO 'Switch default SAML Profile using SessionInitiators' for more information.

More information on the <SessionInitiator> configuration on the Shibboleth WIKI[31].

[29] https://wiki.shibboleth.net/confluence/display/SHIB2/NativeSPServiceSSO

[30] https://wiki.shibboleth.net/confluence/display/SHIB2/NativeSPServiceLogout

[31] https://wiki.shibboleth.net/confluence/display/SHIB2/NativeSPSessionInitiator

- LogoutInitiator

Terminates a session by invoking some kind of logout process, which may be local to the SP or global to the SSO environment. Generally superseded in 2.4+ by the <Logout> service element so if you use that one, you can remove all <LogoutInitiator> elements.

More information on the <LogoutInitiator> configuration on the Shibboleth WIKI[32].

- AssertionConsumerService, ArtifactResolutionService, SingleLogoutService, ManageNameIDService

Incoming entry points for SAML messages depending on the used protocol and binding. Almost always superseded in 2.4+ by the <SSO> service element so if you use that one, you can remove all these elements.

However, if you use <SessionInitiator> elements to configure which protocol and binding is used for communication with the IDP, you will need to define one or more of these services.

See HOWTO 'Switch default SAML Profile using SessionInitiators' for more information.

- Handler

Generic endpoint for non-specific functionality implemented by the SP or an extension library.

The Handlers specified in the default shibboleth2.xml of the distribution are optional and may serve for local testing and debugging rather than for production use.

More information on <Handler> elements on the Shibboleth WIKI[33].

- Errors

Configures error-handling behavior and a few logout-related responses.

```
<Errors session="sessionError.html"
    metadata="metadataError.html"
    access="accessError.html"
    ssl="sslError.html"
    localLogout="localLogout.html"
    globalLogout="globalLogout.html"
    supportContact= support@hostname.be
    logoLocation="/shibboleth-sp/logo.jpg"
    styleSheet="/shibboleth-sp/main.css"/>
```

Note:

- You can point here to your own errorpage(s) or update these as you like
- LogoLocation and styleSheet are used here as defined in apache22.config (see previous section). You can update these to show the errorpages with your own logo and style.

[32] https://wiki.shibboleth.net/confluence/display/SHIB2/NativeSPLogoutInitiator

[33] https://wiki.shibboleth.net/confluence/display/SHIB2/NativeSPHandler

More information on the Shibboleth WIKI[34].

- MetadataProvider

Supplies metadata about identity providers.

One <MetadataProvider> element needs to be added for every metadata file you'd like to point to[35].

You will at least need a MetadataProvider for the metadata of the eHealth IDP.

The metadata file of eHealth IDP is available online.

To receive automatic updates, configure following MetadataProvider.

```
<MetadataProvider type="XML"
        uri="[idp-root]/profile/Metadata/SAML"
        backingFilePath="[workdir]/ehealth-metadata-idp-iam.xml"
        maxRefreshDelay="86400">
</MetadataProvider>
```

Note:

- **Idp-root**: Location of the eHealth IDP webapplication. Please use the location of the IDP in the eHealth environment you wish to integrate with (int, acc, prod)[36]. The content will be different for each environment.
- **Workdir**: (optional) A Location of your choice where the Shibboleth SP can store a cached version of the online metadata. Defaults to [SHIB_HOME]/var/run/shibboleth/
- **maxRefreshDelay**: This is a synonym for the reloadInterval setting, and determines the maximum allowed refresh interval (in seconds) when polling a remote resource for changes.

For more information on the content of the metadata and the reload process, see the documentation on Federation Metadata[37] and the Shibboleth WIKI[38].

- AttributeExtractor

Controls how SAML attributes are decoded and exposed to applications.

Ensure that the path of the Extractor points to the xml configuration file you updated in a previous step.

---

[34] https://wiki.shibboleth.net/confluence/display/SHIB2/NativeSPErrors

[35] With V2.4 and above, Chaining is implied by any configuration with multiple <MetadataProvider> elements, so if you have more than one file, it is no longer explicitly needed to wrap <MetadataProvider> elements inside a <MetadataProvider type="Chaining">.

[36] For production, this will be https://www.ehealth.fgov.be/idp.

[37] See document 'eHealth I.AM – Federation Metadata'.

[38] https://wiki.shibboleth.net/confluence/display/SHIB2/NativeSPMetadataProvider

```
<AttributeExtractor type="XML" validate="true" path="attribute-map.xml"/>
```

More information on <AttributeExtractor> on the Shibboleth WIKI[39].

- AttributeResolver

Controls access to other data sources for attribute information. Primary use is for support of SAML queries to an identity provider for attributes if none are received in the initial assertion. When absent, the SP will not query for attributes.

```
<AttributeResolver type="Query" subjectMatch="true"/>
```

Don't specify any identity attribute. eHealth IDP will know what attributes it must return in response to the AttributeQuery sent by the SP.

- AttributeFilter

Applies rules that filter out unacceptable attribute information.

Ensure that the path of the Filter points to the xml configuration file you updated in a previous step.

```
<AttributeFilter type="XML" validate="true" path="attribute-policy.xml"/>
```

Note:

- You can remove this entry from the shibboleth2.xml file if you do not wish that any filtering is done on the list of mapped attributes.

More information on <AttributeFilter> on the Shibboleth WIKI[40].

- CredentialResolver

Configures the private keys and certificates used by the SP to identify itself to identity providers. This is **NOT** related to the normal SSL/TLS server support provided by web servers.

The <CredentialResolver> element's content must point to the key and certificate to use for signing, decryption, and authenticating the SP to other systems. By default it matches the filenames used for the keypair generated by the installation process but you should reference an eHealth certificate.

```
<CredentialResolver type="File" key="server.pem" certificate="server.crt"/>
```

[39] https://wiki.shibboleth.net/confluence/display/SHIB2/NativeSPAttributeExtractor

[40] https://wiki.shibboleth.net/confluence/display/SHIB2/NativeSPAttributeFilter

Note:

- **Key**: used by the SP to sign and/or decrypt messages to and from the eHealth IDP (if applicable).
- **Certificate**: used by the SP to sign and/or decrypt message s to and from the eHealth IDP (if applicable).

More information on `<CredentialResolver>` on the Shibboleth WIKI[41].
More information on how to receive an eHealth certificate can be found in chapter 3.
More information on Key Rollover in HOWTO 'Ensure Key Rollover'.

- ApplicationOverride

Overrides default behavior by nesting exceptional configuration elements.

See HOWTO 'Add an application' for more information on using an <ApplicationOverride> to configure a new application with different setup.

- **SecurityPolicies**

Deprecated. For versions prior to 2.4.

- **SecurityPolicyProvider**

Replacement for the <SecurityPolicies> element, this externalizes the information previously handled by that element behind a pluggable interface, typically an external configuration file.

In most cases, the default policy provided in the distribution will be appropriate for all typical exchanges between the SP and eHealth IDP.

```
<SecurityPolicyProvider type="XML" validate="true" path="security-policy.xml"/>
```

More information on <SecurityPolicies> on the Shibboleth WIKI[42].

- **ProtocolProvider**

Plugin used to supply default binding/endpoint information for supported protocols to drive the simplified content of the <Sessions> element.

In most cases, the default protocol configuration provided in the distribution will be appropriate for all typical exchanges between the SP and eHealth IDP.

```
ProtocolProvider type="XML" validate="true" reloadChanges="false" path="protocols.xml"/>
```

The file protocols.xml contains by default all SAML1 and SAML2 profiles.

[41] https://wiki.shibboleth.net/confluence/display/SHIB2/NativeSPCredentialResolver

[42] https://wiki.shibboleth.net/confluence/display/SHIB2/NativeSPSecurityPolicies

If you configure the <SSO> element as described in the 'Sessions' element above, this will lead to the use of the profile **SAML 2.0 HTTP-POST,** which we advise you to use as prefered profile.

However, you might have valid reasons to choose another profile.

By removing <Binding> or even <Protocol> elements from the protocols.xml file, you can remove all the profiles you don't want to use.

You can also decide to use the old configuration style using <SessionInitiator> elements which gives you more control. In that case you should remove the <SSO> element to prevent overlaps. The protocols.xml file will be of no use in that case.

See HOWTO 'Switch default SAML Profile using SessionInitiators' for more information on how to configure your Sessions element to select your profile.

5.4 Launch

To start protecting your resources, the following actions must be taken in the given order:

- start apache server
- start shibboleth daemon

Actual launch commands will vary by installation.

More information on the Shibboleth WIKI[43].

5.5 Howto

This section contains optional configuration instructions which you might find useful.

5.5.1 Trace problems

Below are a few things you can do to get more information when you run into problems.

The Shibboleth WIKI[44] also contains an interesting page with tactics to follow.

5.5.1.1 Check Configuration

If the shibd daemon fails to start, you can add an argument to the executable which will verify the configuration.

```
[SHIB_HOME]/sbin/shibd -check
```

[43] https://wiki.shibboleth.net/confluence/display/SHIB2/NativeSPWindowsInstall

[44] https://wiki.shibboleth.net/confluence/display/SHIB2/NativeSPTroubleshootingTactics

5.5.1.2 Log

The level used for logging is defined in 2 logger configuration files, by default [SHIB_HOME]/etc/shibd.logger and [SHIB_HOME]/etc/native.logger.

With a text editor you can alter the loglevel for defaults and for specific categories.

To trace problems, set the default to DEBUG

```
# set overall behavior
log4j.rootCategory=DEBUG, shibd_log, warn_log
```

Tracing of SAML messages sent between the SP and IDP can also be very enlightening.

```
# tracing of SAML messages and security policies
log4j.category.OpenSAML.MessageDecoder=DEBUG
log4j.category.OpenSAML.MessageEncoder=DEBUG
log4j.category.OpenSAML.SecurityPolicyRule=DEBUG
```

When needed, you can lower the level of other categories as well.

You can also alter the location where the logfile of the shibboleth daemon is stored (defaults to [SHIB_HOME]/var/log/shibboleth/shibd.log).

More info on Logging on the Shibboleth WIKI[45].

5.5.2 Use AttributeDecoders

When a Shibboleth SP receives an Authentication Assertion from the IDP, it needs to extract information from it and make it somehow available in your environment.

By default, all identity attributes in the Assertion are ignored by the SP.

Each attribute that you want to extract from the Assertion must be configured in the file 'attribute-map.xml' (see Configuration section for more info) with an ID/Alias so the Attribute can be retrieved later on with this ID.

You MAY also define an `<AttributeDecoder>`.

By default, the `StringAttributeDecoder` will be used, indicated by `xsi:type="XMLAttributeDecoder"`.

It processes the SAML attribute values as a simple string. No additional processing is done.

This will be fine in most cases.

[45] https://wiki.shibboleth.net/confluence/display/SHIB2/NativeSPLogging

However, eHealth uses XML as content of the <AttributeValue> element for some attributes.

If you don't want to ignore those, you'll need to specify an XMLAttributeDecoder, indicated by xsi:type="XMLAttributeDecoder".

It processes SAML attribute values by directly serializing them as XML and storing them in that form. If exported through the CGI interface, the serialized XML is base64-encoded for that purpose.

More information on the Shibboleth WIKI[46].

Example of default decoder (StringAttributeDecoder)

```
<Attribute id="Shib-Organization-Name" name="urn:be:fgov:organization:name" />
```

Example of explicit StringAttributeDecoder

```
<Attribute id="Shib-Organization-Name " name="urn:be:fgov:organization:name ">
        <AttributeDecoder xsi:type="StringAttributeDecoder" />
</Attribute>
```

Example of XMLAttributeDecoder

```
<Attribute id="Shib-Organization-Name-Localised" name="urn:be:fgov:organization:name-
localised">
        <AttributeDecoder xsi:type="XMLAttributeDecoder" />
</Attribute>
```

5.5.3 Pass on identity information from the Shibboleth SP

As described in HOWTO 'Use AttributeDecoders', SAMLAttributes can be extracted from the SAMLAssertion sent by the IDP and decoded into local attributes.

There are two ways the Shibboleth SP can pass on those attributes: using Environment Variables or Request Headers.

Both are described in sections below.

More information on the Shibboleth WIKI[47].

[46] https://wiki.shibboleth.net/confluence/display/SHIB2/NativeSPAttributeDecoder

[47] https://wiki.shibboleth.net/confluence/display/SHIB2/NativeSPAttributeAccess

5.5.3.1 Use Apache Environment Variables

If you use the shib_mod module for Apache HTTPServer, by default, identity information will be made available to your environment thanks to a mechanism provided by the Apache HTTP Server for storing information in named variables, so called environment variables[48].

This information can be used to control various operations such as logging or access control. The variables are also used as a mechanism to communicate with external programs such as CGI scripts.

If your environment supports it, you might also access those variables in your webpages.

For example, to printout the environment variable REMOTE_USER in a php page, you could use the script below.

```
<?php echo( $_SERVER[ REMOTE_USER ] ) ?>
```

See the Shibboleth WIKI for some tool specific examples[49].

5.5.3.2 Use Request Headers

If you do not use Apache HTTPServer or your applicationserver running the protected webapplication has no support to receive the Apache Environment Variables, you can use Request Headers (should be supported on all systems).

This is off by default on Apache.

You can activate it by adding a line to your `Location` directive.

```
<Location /secure>
        …
        ShibUseHeaders On
</location>
```

Please note that using Environment Variables is more safe than Request Headers since client user-agents can not manipulate those whatsoever. You should **always** use this mechanism with web servers that support it.

5.5.4 Protect Java Servlets

The Shibboleth SP is presently only implemented in C++ as a module for Apache httpd, IIS, and NSAPI. However, it's quite easy to use the Shibboleth SP to provide authentication information for Java servlets in a wide variety of servlet containers.

To use Shibboleth to protect java webapplications on an applicationserver, there is an interesting link on the Shibboleth WIKI[50] that explains how to configure the Apache HTTPServer and the Applicationserver running

[48] http://httpd.apache.org/docs/2.2/env.html

[49] https://wiki.shibboleth.net/confluence/display/SHIB2/NativeSPAttributeAccess

[50] https://wiki.shibboleth.net/confluence/display/SHIB2/NativeSPJavaInstall

the webapplication in order to transfer the identity information (stored in Apache Environment Variables) to the applicationserver and use those to create an authenticated user on the ApplicationServer.

5.5.5 Use the RequestMapper

Significant caution should be exercised when using the <RequestMapper> with Apache. If you do so, you MUST ensure that the UseCanonicalName command is On.

Failure to do so will render your system vulnerable to trivial attacks. If for some reason you don't want to turn the option on, do NOT use the <RequestMapper> feature to determine how to protect content.

The <RequestMapper> element configures the component used by the SP to map incoming requests to the set of configuration options that should be applied. You can think of it as a portable equivalent of the Apache <Location> feature, which associates Apache directives with specific URLs. Like that feature, the request mapper operates based on URL, not on the physical path of files.

Shibboleth currently supports 2 types of RequestMapper:

- *Native (default if none specified)*: For most deployments, this is the type to use. It is a hybrid that allows you to combine Apache commands in .htaccess files with XML-based configuration. The native commands override any XML-based attributes. For servers without native commands (IIS), this type is equivalent to the XML request mapper type below.
- *XML*: Provides a portable XML syntax for configuring settings based on mappings assigned to URL host, path, and query string information. It does not permit settings to be defined in the web server's configuration.

The attributes on the <RequestMapper> element are similar to the apache content settings that can be configured.

The <RequestMap> subelement maps the location of the protected application(s) to specific Shibboleth configuration.

A special requirement of this <RequestMap> element is that it MUST contain an applicationId attribute with a value of "default", which in turn matches the required id attribute of the outer-most <ApplicationDefaults> element in the shibboleth2.xml file.

More information can be found on the Shibboleth WIKI[51].

Examples

```
<RequestMapper type="Native">
    <RequestMap applicationId="default"/>
</RequestMap>
```

Note:

[51] https://wiki.shibboleth.net/confluence/display/SHIB2/NativeSPRequestMapper

- Default native RequestMapper (only to use on Apache as this configuration will expect Apache directives to define at least the locations to protect).
- If no RequestMapper is specified, it will default to this one.

```
<RequestMap applicationId="default">
    <Host name="www.partner.org">
        <Path name="secure" authType="shibboleth" requireSession="true"/>
    </Host>
    <Host scheme=”https” name="admin.partner.org" port=”443” applicationId="admin"
authType="shibboleth" requireSession="true">
        <AccessControl>
            <Rule require="Shib-Role">User AdminUser</Rule>
        </AccessControl>
    </Host>
</RequestMap>
```

Note:

- Host name and port must match these as configured for the webserver.
- Path name must match the path in the url used by clients to reach the protected resource
- <AccessControl> elements can be attached to the <RequestMap> itself or to the <Host>,<Path> and <Query> elements on lower levels. See HOWTO ‘Attach an Access Control Policy’ for more information on this.

5.5.6 Use Apache Content Settings

Shibboleth SP supports an extensible set of content settings, properties that control how it interacts with requests and enforces various requirements.

On Apache, these settings can be controlled using either the ShibRequestSetting command or by attaching properties using the <RequestMapper> mechanism in the SP configuration. By default, the SP ships with the "Native" RequestMapper plugin enabled. This plugin is what permits settings to be attached using either the native Apache command, or the XML syntax.

Note that Apache takes precedence if configuration overlaps so be carefull.

To use the ShibRequestSetting command, you simply add a pair of parameters with the name of the content setting and the value you want.

Listed below are some settings that may be useful in common cases.

A complete overview can be found on the Shibboleth WIKI[52].

[52] https://wiki.shibboleth.net/confluence/display/SHIB2/NativeSPApacheConfig#NativeSPApacheConfig-ContentSettings

5.5.6.1 Select an ApplicationOverride

If you have multiple applications that need separate authentication sessions, you must define an ApplicationOverride.

See HOWTO 'Add an application' for information on how to do this.

To link the locations of your separate applications to these ApplicationOverrides, you can use the ShibRequestSetting applicationId.

```
<Location /myapp2>
        AuthType shibboleth
        ShibRequestSetting applicationId myApp2
        …
</Location>
```

Note:

- Shibboleth SP will look for `<ApplicationOverride id="myApp2" …>` in the shibboleth2.xml config file..

5.5.6.2 Select a SessionInitiator

If you don't want to rely on the <SSO> element or default `<SessionInitiator>` for one of your applications, you can select which SessionInitiator to use with the ShibRequestSetting requireSessionWith.

```
<Location /myapp2>
        AuthType shibboleth
        ShibRequestSetting requireSessionWith sso_saml1_post
        …
</Location>
```

Note:

- Shibboleth SP will look for `<SessionInitiator id="sso_saml1_post" …>` in the shibboleth2.xml config file..

5.5.7 Attach an Access Control Policy

Shibboleth SP contains a plugin interface for Access Control and includes a pair of implementations, one that's portable and one that enables "familiar" use of the Apache `Require` command in the usual places, including `.htaccess` files.

You can choose not to use this at all (by default, if you follow the basic installation steps in the configuration section, no access control is performed) and let a dedicated authorization module in your environment handle all access control or the application itself.

Often it's best if applications can do this checking themselves, because the experience can be more friendly and integrated with the rest of the application.

Depending on your environment, application and access rules to apply, you might find it apropriate to attach an Access Control Policy to your Shibboleth SP in which case you can use one of the options listed below.

5.5.7.1 Apache native configuration

If you use Apache Directives to map Locations to Shibboleth configuration, you can simply add native commands of Apache to perform some (basic) authorization rules.

Even if you use the <RequestMap>[53] element, you can activate Apache Authorisation Rules by attaching the <htaccess> element to it. This will enable Apache .htaccess support during the authorization phase. This is automatic and implicit for the "Native" request mapper, but can be enabled by hand if the "XML" request mapper is used. Note that this will fail for non-Apache servers.

When used with Apache, the SP includes an Access Control plugin implemented on top of the Apache Require authorization command. The placement rules for this command are dictated by Apache, and include its <Directory>, <File>, and <Location> blocks, as well as .htaccess files. By default, any place you can apply the Require command will trigger the Access Control implementation supplied with the SP's Apache module.

Make sure that any user will have already established a session, or require a session for the same path, or any access attempt will result in a 403.

More information can be found on the Shibboleth WIKI[54].

Examples

```
<Location /secure>
        AuthType shibboleth
        ShibRequestSetting requireSession On
        Require valid-user
</Location>
```

Note:

- Basic example that only specifies that authentication is done by Shibboleth and that an authenticated session is required, but nothing else. No information of any kind about the user is required in order to satisfy this rule and it should never be used in the absence of additional application logic to perform authorization.

```
<Location /secure>
        AuthType shibboleth
        ShibRequestSetting requireSession On
        Require Shib-Authz-Decision Permit
</Location>

<Location /secure/admin>
        Require Shib-Authz-Decision Permit
        Require Shib-Person-ID ! 70011208925
        Require Shib-Role AdminUser
```

[53] https://wiki.shibboleth.net/confluence/display/SHIB2/NativeSPRequestMap

[54] https://wiki.shibboleth.net/confluence/display/SHIB2/NativeSPhtaccess

```
        ShibRequireAll On
</Location>
```

Note:

- There are more `Require` rule-types supported but this examples uses the Shibboleth extension to verify environment variables matching the Attribute IDs defined in attribute-map.xml. See Configuration Section.
- The above Access Control Policy requires users for the whole `/secure` webapplication to have the attribute `Shib-Authz-Decision` with value '`Permit`' and for the `/admin` part also to have the role of `AdminUser`. Person with ID '`70011208925`' is blacklisted.

5.5.7.2 AccessControl Plugin

The `<AccessControl>` element is the root of an XML-based access control policy that prevents access to a resource unless the user's session satisfies the policy. It's a simple, boolean-capable language provided as an example of how to implement an access control plugin.

The `<AccessControl>` element can be attached inline to a `<RequestMap>`[55] or in an external file in which case you'll need to configure an `<AccessControlProvider>`[56] instead to reference the external resource.

More information can be found on the Shibboleth WIKI[57].

Example

```
<AccessControl>
    <AND>
        <Rule require="Shib-Authz-Decision">Permit</Rule>
        <NOT>
            <Rule require="Shib-Person-ID">70011208925</Rule>
        </NOT>
        <OR>
            <Rule require="Shib-Role">User</Rule>
            <Rule require="Shib-Role">AdminUser</Rule>
        </OR>
    </AND>
</AccessControl>
```

Note:

- The Access Control Policy above requires the user to have the attribute "`Shib-Authz-Decision`" with value "`Permit`", not to have the ID '`70011208925`' en have one or both of the roles `User`, `AdminUser`.

[55] https://wiki.shibboleth.net/confluence/display/SHIB2/NativeSPRequestMap

[56] https://wiki.shibboleth.net/confluence/display/SHIB2/NativeSPAccessControlProvider

[57] https://wiki.shibboleth.net/confluence/display/SHIB2/NativeSPXMLAccessControl

- The values of the `require` attribute MUST match the Attribute IDs of the attribute-map.xml configuration file. See Configuration Section.
- The content of the <Rule> element will be evaluated against the values of the received attributes.
- If you are using the AccessControl element in an external file outside of shibboleth2.xml, you may have to add one of the following root elements to make it work (depends on your Shibboleth distribution).
    - `<AccessControl type="edu.internet2.middleware.shibboleth.sp.provider.XMLAccessControl">`
    - `<AccessControl xmlns="urn:mace:shibboleth:2.0:native:sp:config">`

5.5.7.3 CGIScript

Another option for access control could be a CGIScript installed on your HTTPServer.

Such a script has access to the environment variables set by the SP in the HTTPServer.

When executed between the user's return to the SP and the intended "target" resource, this script could check the user's session for required information, and either return an error or forward the user on to the original URL..

An example can be found on the Shibboleth WIKI[58].

5.5.8 Enforce eHealth Authz Decision

Before an authenticated user is granted access to an application, you must always perform some kind of Access Control.

You can do this in the application itself, in a dedicated authorization module on your server or, as seen in the HOWTO 'Attach an Access Control Policy', the SP can perform basic access control. Or you can even choose to combine them (basic authorization at the level of the webserver and more specific authorization in a dedicated authorization module).

If your application is integrated with eHealth's central identiy & authorization management, part of the authorization decision can allready have been made by the eHealth IDP.

If so, partners MUST take this decision into account when executing access control.

5.5.8.1 eHealth Authorization Attributes

The eHealth authorization decision is sent by the IDP using some extra identity attributes in the Assertion to the SP.

They are listed below.

- **Attribute 'urn:be:fgov:ehealth:1.0:authz-decision'**

Sent: Optional, i.e. always unless executing any access control rules was disabled at eHealth for your application(s).

Possible values:

[58] https://wiki.shibboleth.net/confluence/display/SHIB2/NativeSPErrors#NativeSPErrors-AdditionalThoughts

- *'Permit'*: based on the rules executed by eHealth, current user is authorized to access the application.
    - Action: You MAY execute additional access controls before access is granted or denied.
- *'Deny'*: based on the rules executed by eHealth, current user is not authorized to access the application.
    - Action: You MUST refuse access to the user.
- *'Indeterminate'*: eHealth could not take a decision.
    - Action: You SHOULD refuse access to the user unless you have valid reasons to expect an 'Indeterminate' decision by eHealth in which case you MUST execute yourself access rules to take an apropriate decision.

- **Attribute 'urn:be:fgov:ehealth:1.0:role'**

Sent: Optional, i.e when attribute 'urn:be:fgov:ehealth:1.0:authz-decision' equals 'Permit'.

Possible values: one or more roles (string), known in the eHealth Identity & Authorization Management (they are listed in the 'DU' of your application).

- **Attribute 'urn:be:fgov:ehealth:1.0:ehealth-ref'**

Sent: Always.

Possible values: one unique identifier (string) for the current authentication session of the user in the eHealth IDP.

To be able to use these attributes, you must make sure they are mapped to local Attribute IDs in the attribute-map.xml (see Configuration section) otherwise they will be filtered out by the Shibboleth SP.

Example (SAML 2.0 attribute-map.xml)

```
<Attribute id="Shib-Authz-Decision" name="urn:be:fgov:ehealth:1.0:authz-decision" />
<Attribute id="Shib-eHealth-Ref" name="urn:be:fgov:ehealth:1.0:ehealth-ref" />
<Attribute id="Shib-Role" name="urn:be:fgov:ehealth:1.0:role" />
```

Example Apache Require rule

```
<Location /secure>
        AuthType shibboleth
        ShibRequestSetting requireSession On
        Require Shib-Authz-Decision Permit
</Location>
```

Example <AccessControl> rule

```
<AccessControl>
        <Rule require="Shib-Authz-Decision">Permit</Rule>
</AccessControl>
```

Example JEE Role Based Access

```
<security-constraint>
        <web-resource-collection>
                <web-resource-name>Administrators</web-resource-name>
                <description>Only for administrators</description>
                <url-pattern>/admin/*</url-pattern>
                <http-method>GET</http-method>
                <http-method>POST</http-method>
        </web-resource-collection>
        <auth-constraint>
                <role-name>AdminUser</role-name>
        </auth-constraint>
</security-constraint>
<login-config>
        <auth-method>FORM</auth-method>
        <form-login-config>
                <form-login-page>/login</form-login-page>
                <form-error-page>/error</form-error-page>
        </form-login-config>
</login-config>
<security-role>
        <role-name>AdminUser</role-name>
</security-role>
```

Note

- This is a typical example of Role based Access on an applicationserver (running your application)
- This example assumes you have a central authorization (JAAS or other) module that creates a Subject with Principals and Roles and a script (like a java servlet) to handle authentication information, made available by the SP.

5.5.8.2 Access Denied Errorpage

When a user is denied access, you MUST show him a user friendly message.

For this purpose (**only on condition Shib-Authz-Decision 'Deny' or 'Indeterminate'**), you MAY redirect the user to the NoAccess page of the eHealth IDP[59] where a standard friendly 'not authorized' message will be shown, along with the 'Shib-eHealth-Ref' he can use to contact the serviceDesk if he does not agree.

You SHOULD do a local logout first and then redirect to the eHealth IDP to invalidate the local session which did not grant him access to the application.

Example Apache ErrorDocument

```
<Location /secure>
    AuthType shibboleth
    ShibRequestSetting requireSession On
    Require Shib-Authz-Decision Permit
    ErrorDocument 403 https://www.partner.org/Shibboleth.sso/Logout?return=https://www-
env.ehealth.fgov.be/idp/noaccess
</Location>
```

[59] '[ehealth environment]/idp/noaccess' where [ehealth environment] is the hostname of the ehealth environment from which your SP received the message.

Note

- You may only use this 403 redirect for the conditions as explained above.

You MAY also choose to show your own 'access denied' page if you feel that it is more apropriate for your application. If you do so, you MUST display the 'Shib-eHealth-Ref' attribute on that page so the user is able to request help with it if he does not agree.

In ANY other case of access denial (which means you executed extra Access Control Rules), you MUST use an 'access denied' Errorpage of your own environment. You SHOULD also not use the 'Shib-eHealth-Ref' attribute as the decision was made by your additional access control rules, not by eHealth. If a user disagrees he should request help from the servicedesk of the SP.

Example <AccessControl> rule

```
<AccessControl>
    <AND>
        <Rule require="Shib-Authz-Decision">Permit</Rule>
        <NOT>
            <Rule require="Shib-Person-ID">70011208925</Rule>
        </NOT>
        <OR>
            <Rule require="Shib-Role">User</Rule>
            <Rule require="Shib-Role">AdminUser</Rule>
        </OR>
    </AND>
</AccessControl>
```

Note

- A 403 will be thrown when access is refused by the Shibboleth Access Control Plugin. This will lead to the standard 403 Errorpage of your webserver if none specified or to the ErrorDocument 403 URL if one is specified for a matching location (as in above example). You can use the `access` attribute on the `<Errors>` element (or `accessErrors` on `<RequestMap>`) to specify a template html or use `redirectErrors` to go to a dynamic errorpage that can handle the error apropriately. More information on the Shibboleth WIKI[60].

Example JEE AccessControl by Application

```
<filter>
        <filter-name>AuthFilter</filter-name>
        <filter-class>org.partner.app.AuthFilter</filter-class>
</filter>
<filter-mapping>
        <filter-name>AuthFilter</filter-name>
        <url-pattern>/*</url-pattern>
</filter-mapping>

<error-page>
```

[60] https://wiki.shibboleth.net/confluence/display/SHIB2/NativeSPErrors

```
        <error-code>403</error-code>
        <location>/noaccess</location>
</error-page>
```

Note

- An AuthFilter could retrieve all the identity attributes (using request.getAttribute('Shib-…')[61]), execute access rules and throw a 403 HTTP Error when needed. A script (like a servlet) on /noaccess could then verify if the 403 was caused by a Deny from eHealth or not and redirect to eHealth or forward to a local view.

5.5.9 Add an application

If you only have one application to protect, you can use the default configuration of your webserver and the Shibboleth SP.

If you have more applications that cannot share authentication sessionstate, you'll need to configure an ApplicationOverride that overrides one or more of the default settings from the `<ApplicationDefaults>` element.

More information can be found on the Shibboleth WIKI[62].

Example

```
<ApplicationOverride id="myApp2" entityID="http://www.partner.be/sp/myApp2">

        <Sessions handlerURL="/myapp2/Shibboleth.sso" cookieProps="; path=/myapp2; secure;"
checkAddress="false">
        </Sessions>

        <AttributeExtractor type="XML" file="attribute-map-app2.xml"/>

</ApplicationOverride>
```

Note

- You must specify an `id` other than "default".
- If you don't use a different 'virtual host', you must specify a `<Sessions>` element with a `handlerURL` that lives **inside** the path you set aside for this application to use.
- If your application requires other attributes, you MAY add an `<AttributeExtractor>`. (additional configuration for other elements of `<ApplicationDefaults>` can be added as well if needed)

```
<Location /myapp2/Shibboleth.sso>
        SetHandler shib
</Location>
```

[61] See HOWTO 'Protect Java Servlets'.

[62] https://wiki.shibboleth.net/confluence/display/SHIB2/NativeSPApplicationOverride

```
<Location /myapp2>
       AuthType shibboleth
       ShibRequestSetting applicationId myApp2
       Require valid-user
</Location>
```

Note

- This configuration is for Apache only.
- To make sure that the handlerURL is properly sent to the shib daemon, set the shib handler.
- Set a `ShibRequestSetting applicationId` to reference your `<ApplicationOverride>`
- `Require` rules and other Apache directives can be set if needed

5.5.10 Switch default SAML Profile using SessionInitiators

As mentioned in "Update <shib-sp>\etc\shibboleth\shibboleth2.xml", the sessions section of the shibboleth2.xml file defines which protocol and binding will be used to communicate with the eHealth IDP.

Both eHealth IDP and Shibboleth SP support all SAML 1.1 and SAML 2.0 Web Browser SSO profiles.

As it is the SP that initiates the session, it is the SP that decides which profile is used.

Since version 2.4 you can use the <SSO>[63] element to let the SP decide which profile is most apropriate.

If you need more control, you can use <SessionInitiator> elements to decide which Protocol and Binding the SP should (prefer to) use.

You can define multiple SessionInitiators, each with a unique id attribute[64].

One SessionInitiator can also be set as default.

If none specified explicitely for your application, this default will be used.

See the sections below on how to switch the default Protocol and Binding by configuring <SessionInitiator> elements.

For a full explanation of the supported profiles by the IDP, see document 'eHealth I.AM – IDP' or the SAML Specifications[65].

5.5.10.1 SAML 2.0 HTTP-POST

With the configuration below SAML 2.0 HTTP-POST is configured as default profile and SAML 1.1 Browser/POST as backup. eHealth proposes SAML 2.0 HTTP-POST as default. It limits the number of communications between SP and IDP to one redirect and one POST back with an encrypted assertion[66].

[63] See the Configuration section on how to use this element and how you can eliminate Protocols and Bindings you don't want to use.

[64] See HOWTO 'Select a SessionInitiator' to link a certain SessionInitiator to your application.

[65] http://docs.oasis-open.org/security/saml/v2.0/

```
<Sessions lifetime="28800" timeout="3600" checkAddress="false"
          handlerURL="/Shibboleth.sso" handlerSSL="false"
         exportLocation="http://localhost/Shibboleth.sso/GetAssertion" exportACL="127.0.0.1"
idpHistory="false" idpHistoryDays="7">

<SessionInitiator type="Chaining" Location="/Login" id="sso"
    relayState="cookie" entityID="http://idp.smals-mvm.be/shibboleth" isDefault="true">
    <SessionInitiator type="SAML2" acsIndex="1" acsByIndex="false"
template="bindingTemplate.html"/>
    <SessionInitiator type="Shib1" acsIndex="5" />
</SessionInitiator>

<md:AssertionConsumerService Location="/SAML2/POST" index="1"
                Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST" />
<md:AssertionConsumerService Location="/SAML/POST" index="5"
                Binding="urn:oasis:names:tc:SAML:1.0:profiles:browser-post"/>
```

Note

- **Sessions handlerURL**: must be unique for every protected application. It will be constructed as follows:

  [scheme]://[servername]:[port][handlerURL][AssertionConsumerServiceLocation]

  where AssertionConsumerServiceLocation is the value of the attribute `Location` of the desired `AssertionConsumerService` (see elements <`md:AssertionConsumerService`>).

- **SessionInitiator type:** SAML2, Shib1, Chaining.
  - **SAML2:** to trigger a SAML2 profile for the authentication process with the IDP.
  - **Shib1:** to trigger a SAML1 profile for the authentication process with the IDP.
  - **Chaining:** to group some SessionInitiators together. First in the list will have precedence as long as both Shibboleth SP and eHealth IDP support what is defined in the configuration files. In above example, the Shibboleth SP will try to use SAML2 over SAML1 but will only do so if it finds enough configuration in the SP and eHealth IDP metadata to do so.
- **SessionInitiator Location:** a relative url you can choose. It will be constructed by shibboleth at runtime, in conjunction with the handlerURL for your application. Using that url with trigger lazy session initialisation.
- **SessionInitiator entityID**: must contain the ID of the ehealth IDP (value as in example for all environments). The Shibboleth SP will use it to search the metadata of the IDP for the correct URL, certificate, nameIDFormat, … to use in its communication to the eHealth IDP.
- **SessionInitiator relayState**: should be set to cookie so the Shibboleth SP keeps track of the requested target application URL in a cookie until the end of the authentication process. After that, it will give back the handle to the webserver with this target application URL.
- **SessionInitiator isDefault**: should be set for the above SessionInitiator if you have more than 1 sessionInitiator configured in your configuration file so the Shibboleth SP uses this one by default for integration with the eHealth IDP.
- **SessionInitiator acsIndex:** should point to the index of the `md:AssertionConsumerService` of your choice, that is the one you want to use for the authentication process.

---

[66] Using the <SSO> element as described in the Configuration section will also result in default profile SAML 2.0 HTTP-POST if you use the protocols.xml file from the Shibboleth distribution.

- **SessionInitiator acsByIndex**: should be set to false so the SP uses the `md:AssertionConsumerService Location` and not the `index` in its Authentication Request to the eHealth IDP. eHealth needs the location to map the request to an application.
- **AssertionConsumerService Location**: relative url you can choose. It will be constructed by shibboleth at runtime, in conjunction with the `handlerURL` for your application. It will be sent by the SP as parameter in the request to the IDP which will use it to identify your application. After authentication, the IDP will use it to reply to the SP to complete the authentication process.
- **AssertionConsumerService index:** each `AssertionConsumerService` must have a unique index. It is used to let the `SessionInitiator` specify which `AssertionConsumerService` to use.
- **AssertionConsumerService Binding:** used by the Shibboleth SP to know which binding to use for the selected profile. Don't change these values.

5.5.10.2 SAML 2.0 HTTP-Artifact

When this profile is chosen, there will be a backchannel callback from the SP to the IDP to resolve the sent Artifact.

Configuration is similar as for SAML 2.0 HTTP-POST except that you must update the `acsIndex` of your SAML2 `SessionInitiator` to point to the index of your AssertionConsumerService with binding '`urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Artifact`'.

```
    <SessionInitiator type="SAML2" acsIndex="3" acsByIndex="false"
template="bindingTemplate.html"/>

<md:AssertionConsumerService Location="/SAML2/Artifact" index="3"
                Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Artifact" />
```

5.5.10.3 SAML 1.1 Browser/POST

When this profile is chosen, there will probably be a backchannel callback from the SP to the IDP with an AttributeQuery to receive the identity information. Since SAML 1.1 does not support encryption, posting the identity information over a browser form would in some cases not be secure enough.

Configuration is similar as for SAML 2.0 HTTP-POST except that you must reverse the order of your chained SessionInitiators.

```
<SessionInitiator type="Chaining" Location="/Login" id="sso"
    relayState="cookie" entityID="http://idp.smals-mvm.be/shibboleth" isDefault="true">
    <SessionInitiator type="Shib1" acsIndex="5" />
    <SessionInitiator type="SAML2" acsIndex="1" acsByIndex="false"
template="bindingTemplate.html"/>
</SessionInitiator>

<md:AssertionConsumerService Location="/SAML2/POST" index="1"
                Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST" />
<md:AssertionConsumerService Location="/SAML/POST" index="5"
                Binding="urn:oasis:names:tc:SAML:1.0:profiles:browser-post"/>
```

5.5.10.4 SAML 1.1 Browser/Artifact

When this profile is chosen, there will be a backchannel callback from the SP to the IDP to resolve the sent Artifact.

Configuration is similar as for SAML 1.1 Browser/POST except that you must update the acsIndex of your Shib1 SessionInitiator to point to the index of your AssertionConsumerService with binding '`urn:oasis:names:tc:SAML:1.0:profiles:artifact-01`'.

```
    <SessionInitiator type="Shib1" acsIndex="6" />

<md:AssertionConsumerService Location="/SAML/Artifact" index="6"
                Binding="urn:oasis:names:tc:SAML:1.0:profiles:artifact-01" />
```

5.5.11 Configure the SAML 2.0 AuthnRequest

If you are using a SAML 2.0 Profile that sends a SAML 2.0 AuthnRequest, it will by default send a basic `<AuthnRequest>` element.

However, this element has some optional attributes and elements which the SP can use for specific configuration settings between SP and IDP during the authentication process.

Some of them can be set using the SAML2 SessionInitiator which provides attributes to configure the different AuthnRequest parts.

For advanced request content, it also allows an embedded `<AuthnRequest>` element so the SP can completely decide what content it wants to send to the IDP in its authentication request.

More info on the Shibboleth WIKI[67].

5.5.12 Ensure Key Rollover

If you are using a SAML Profile that includes signing requests and decrypting responses, your SP has keys and certificates configured to perform those tasks. Typically those keys and certificates expire after some time.

Unless you don't worry about a period of unavailability for the applications this SP is protecting, you'll need to ensure Key "Rollover", which means that your SP and the IDP gradually start using the new key and certificate to sign requests and encrypt responses without interruptions.

Typically with rolling over keys or certificates you have two separate concerns: migrating your signing/authentication keys and your encryption keys. On the authentication side, the solution is to publish the new certificate to your partners and wait for them to register that certificate before switching your SP from the old credential to the new one. In that scenario, you need not ever configure both within the SP itself (i.e., it's different from this HOWTO's goal).

On the other hand, with encryption you have the opposite problem. Once you publish a new key to your partners, they may start using it so you must install it in your system before they do so. So in this case, you MUST configure both the old and new credentials into the SP so that either key can be available for decryption.

[67] https://wiki.shibboleth.net/confluence/display/SHIB2/NativeSPSessionInitiator#NativeSPSessionInitiator-SAML2SessionInitiatorProtocolHandler

5.5.12.1 Signing credentials

If the IDP never encrypts responses to your SP, the process of Key Rollover is quite simple.

You probably have something like this in your Shibboleth2.xml file

```
<CredentialResolver type="File" key="sp-key.pem" certificate="sp-cert.pem"/>
```

Before you start using the new signing credential, you should communicate the new certificate (never the private key!!) to eHealth. In the example above, this is sp-cert.pem.

You must use the registration form[68] to do so.

Once eHealth confirms that it has uploaded the new certificate as trusted for your SP, you can safely replace the Signing Credential of your SP.

5.5.12.2 Signing/Encryption credentials

If the IDP encrypts responses to your SP, it is often, a single credential which is used for both signing and encryption and the migration is from one credential for both to another for both.

Therefore, if you want to perform rollover of a single keypair for both signing and encryption, you can follow these steps:

1. Add a second private key to your SP configuration explicitly as a decryption key (use="encryption").
2. Publish the corresponding public key to your partners.
3. Switch the encryption usage constraint in the SP configuration from the new key to the old key.
4. Let your partners remove the old public key from the metadata so they will surely no longer use it for encryption nor trust it for signing.
5. Remove the old private key from your SP configuration.

As an example, if you start here:

```
<CredentialResolver type="File" key="sp-key.pem" certificate="sp-cert.pem"/>
```

Then the configuration after step 1 might be:

```
<CredentialResolver type="Chaining">
     <CredentialResolver type="File" key="new-key.pem" certificate="new-cert.pem"
use="encryption"/>
     <CredentialResolver type="File" key="sp-key.pem" certificate="sp-cert.pem"/>
</CredentialResolver>
```

After step 3:

[68] Document 'eHealth I.AM – Registration'. You should have filled this form upon registration of your application. Please update the X.509 certificate fields where applicable.

```
<CredentialResolver type="Chaining">
     <CredentialResolver type="File" key="new-key.pem" certificate="new-cert.pem"/>
     <CredentialResolver type="File" key="sp-key.pem" certificate="sp-cert.pem"
use="encryption"/>
</CredentialResolver>
```

And after step 5:

```
<CredentialResolver type="File" key="new-key.pem" certificate="new-cert.pem"/>
```

More information on the Shibboleth WIKI[69].

[69] https://wiki.shibboleth.net/confluence/display/SHIB2/NativeSPMultipleCredentials

6 Risks and security

6.1 Security

6.1.1 Business security

In case the development adds an additional use case based on an existing integration, eHealth must be informed at least one month in advance with a detailed estimate of the expected load. This will ensure an effective capacity management.

In case of technical issues on the web service, the partner may obtain support from the contact center that is responsible for this service.

In case eHealth finds a bug or vulnerability in its software, the partner is advised to update his application with the newest version of the software within 10 business days.

In case the partner finds a bug or vulnerability in the software or web service that eHealth delivered, he is obliged to contact and inform eHealth immediately and he is not allowed to publish this bug or vulnerability in any case.

6.1.2 Web SSO

6.1.2.1 AuthnRequest

Web SSO Authentication Requests from a requesting Service Provider (SP) to eHealth are sent over a one-way SSL connection.
Depending on the used SSO Profile (see 'eHealth I.AM - IDP'), the request from an SP can include an enveloped signature as described in the SAML Specifications (1.x or 2.0).
This signature will be used to authenticate the SP as requesting entity.
The key used to sign these messages must be stored on the partner system in a secure manner in order to prevent unauthorized use.

6.1.2.2 Response

In response to a Web SSO AuthnRequest, eHealth will send a signed Response.
The recipient should process the Response following the processing rules as described in the specifications of the used SSO Profile (see 'eHealth I.AM - IDP').
The recipient (SP) must also verify if the Response is signed by a valid eHealth IDP certificate and that it is untampered with.
The eHealth IDP certificates are published online using SAML 2.0 Metadata (See eHealth I.AM - Federation Metadata for more information).
If the used SSO Profile is SAML 2.0 HTTP-POST, the response from eHealth will also be encrypted with a certificate belonging to the recipient (SP).

7 Test and release procedure

7.1 Procedure

This chapter explains the procedures for testing and releasing an application in acceptation or production.

7.1.1 Initiation

If you intend to use the eHealth service, please contact info@ehealth.fgov.be. The Project department will provide you with the necessary information and mandatory documents.

7.1.2 Development and test procedure

You have to develop a client in order to connect to our web service. Most of the required integration info to integrate is published in the technical library on the eHealth portal.

In some cases eHealth provides you with a mock-up service or test cases in order for you to test your client before releasing it in the acceptance environment.

7.1.3 Release procedure

When development tests are successful, you can request to access the eHealth acceptance environment.

From this moment, you start integration and acceptance tests. eHealth suggests testing during minimum one month.

After successful acceptance tests, the partner sends his test results and performance results with a sample of “eHealth request” and “eHealth answer” to the eHealth point of contact by email.

Then eHealth and the partner agree on a release date. eHealth prepares the connection to the production environment and provides the partner with the necessary information. During the release day, the partner provides eHealth with feedback on the test and performance tests.

For further information and instructions, please contact: integration@ehealth.fgov.be.

7.1.4 Operational follow-up

Once in production, the partner using the eHealth service for one of its applications will always test first in the acceptance environment before releasing any adaptations of its application in production. In addition, he will inform eHealth on the progress and test period.