Test Cases

- Section numbers correspond to the eGov 2.0 draft profile.

Test Case eGov2-1 - Production of IOP-compliant Metadata

Scope

- Verify the ability to produce metadata conformant to the Metadata IOP.

IOP-conformant metadata has a different meaning from metadata intended to be evaluated in a PKIX environment, but syntactically should be identical based on the eGov profile language, so this should be sufficient to test production of metadata for both profiles.

Preconditions

- Implementation configured sufficiently to produce metadata identifying its signing, TLS, and encryption keys.
- Details of expected md:KeyDescriptor content available to tester

Test Sequence

1. Access published metadata

The metadata produced by the implementation is obtained.

CONFIRM: The content(s) of the md:KeyDescriptor element(s) matches the expected output.

Test Case eGov2-2 - Consumption of IOP-compliant Metadata

Testing is best accomplished with a fixed implementation to test against, as the purpose is not to test actual protocol correctness, but use of the metadata itself.

Scope

- Verify the ability to process metadata conformant to the Metadata IOP.
- Verify support for bare keys, self-signed, arbitrary-CA, and expired certificates.
- Check for bypass of CRL or OCSP extensions in certificates.
- Verify ability to match keys between different certificates
- Verify recognition of a removed/revoked key.
- Verify enforcement of validUntil.

Preconditions

- Keys and certificates generated with various characteristics:
    - Self-signed, no revocation-related extensions
    - Signed by a CA, with revocation-related extensions
    - Expired
    - Multiple certificates with the same public key

Test Sequence

1. Test Bare Keys

a. Metadata is prepared for the fixed implementation containing a pair of md:KeyDescriptors both containing a ds:KeyValue containing a ds:RSAKeyValue. The fixed implementation is configured to use one of the keys for signing and/or TLS purposes. The metadata is supplied to the candidate system, and one or more SSO operations is attempted. Bindings should be chosen to exercise both signature verification and TLS authentication if possible.

CONFIRM: The SSO operations are successful based on the use of the key(s) that appear in the metadata.

b. The fixed implementation is switched to use the second of the pair of keys in its metadata.

CONFIRM: The SSO operations are successful based on the use of the key(s) that appear in the metadata.

c. The fixed implementation is switched to use a key that is NOT in its metadata.

CONFIRM: The SSO operations are unsuccessful.

2. Test Self-Signed Certificates

a. Metadata is prepared for the fixed implementation containing a pair of md:KeyDescriptors both containing a ds:X509Certificate, with the certificates self-signed. One of the certificates should be expired. The fixed implementation is configured to use one of the certificates for signing and/or TLS purposes. The metadata is supplied to the candidate system, and one or more SSO operations is attempted. Bindings should be chosen to exercise both signature verification and TLS authentication if possible.

CONFIRM: The SSO operations are successful based on the use of the key(s) that appear in the metadata.

b. The fixed implementation is switched to use the second of the pair of certificates in its metadata.

CONFIRM: The SSO operations are successful based on the use of the key(s) that appear in the metadata.

c. The fixed implementation is switched to use a different certificate that is not in the metadata but contains the same key as one that is.

CONFIRM: The SSO operations are successful based on the use of the key(s) that appear in the metadata.

d. The fixed implementation is switched to use a certificate and key that is NOT in its metadata.

CONFIRM: The SSO operations are unsuccessful.

2. Test CA-Issued Certificates

a. Metadata is prepared for the fixed implementation containing a pair of md:KeyDescriptors both containing a ds:X509Certificate, with the certificates signed by one or more arbitrary CAs. The certificates should, if possible, include a CRL distribution point or OCSP location extension. These endpoints should not exist or be unavailable to the testing environment. The fixed implementation is configured to use one of the certificates for signing and/or TLS purposes. The metadata is supplied to the candidate system, and one or more SSO operations is attempted. Bindings should be chosen to exercise both signature verification and TLS authentication if possible.

CONFIRM: The SSO operations are successful based on the use of the key(s) that appear in the metadata. No delays in processing occur as a result of failed attempts to contact non-existent or unavailable CRL or OCSP endpoints.

b. The fixed implementation is switched to use the second of the pair of certificates in its metadata.

CONFIRM: The SSO operations are successful based on the use of the key(s) that appear in the metadata.

c. The fixed implementation is switched to use a different certificate that is not in the metadata but contains the same key as one that is.

CONFIRM: The SSO operations are successful based on the use of the key(s) that appear in the metadata.

d. The fixed implementation is switched to use a key that is NOT in its metadata.

CONFIRM: The SSO operations are unsuccessful.

3. Verify Revocation and validUntil

a. Metadata is prepared for the fixed implementation containing a md:KeyDescriptor containing a key in one of the supported formats. The fixed implementation is configured to the key for signing and/or TLS purposes. The metadata is supplied to the candidate system, and one or more SSO operations is attempted. Bindings should be chosen to exercise both signature verification and TLS authentication if possible.

CONFIRM: The SSO operations are successful based on the use of the key(s) that appear in the metadata.

b. The metadata is altered to remove the md:KeyDescriptor and the candidate implementation is configured to refresh the appropriate source of metadata.

CONFIRM: The SSO operations are unsuccessful.

c. The metadata is altered to add back the md:KeyDescriptor, but a validUntil attribute is added such that the metadata is expired, and the candidate implementation is configured to refresh the appropriate source of metadata.

CONFIRM: The SSO operations are unsuccessful.

Test Case eGov2-3 - Support for "Metadata Extension for Entity Attributes" Profile

Scope

- Test SP acceptance of SSO based on IdP metadata extension content

The proposed tag would be an attribute named "urn:oasis:names:tc:SAML:attribute:assurance-certification" and would appear as follows:

```
<EntityDescriptor entityID="https://idp.example.org/SAML" ... >
   <Extensions>
      <attr:EntityAttributes xmlns:attr="urn:oasis:names:tc:SAML:metadata:attribute">
        <saml:Attribute
            NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
            Name="urn:oasis:names:tc:SAML:attribute:assurance-certification">
          <saml:AttributeValue>
            http://foo.example.com/assurance/loa1
          </saml:AttributeValue>
        </saml:Attribute>
      </attr:EntityAttributes>
    </Extensions>
    <IDPSSODescriptor...>
      ...
    </IDPSSODescriptor>
</EntityDescriptor>
```

Preconditions

- SP configured with metadata for candidate IdP containing acceptable LOA "tag".
- SP configured with metadata for candidate IdP not containing acceptable LOA "tag".
- SP configured to require presence of "tag" in metadata for IdPs before it will accept SSO from them.

Test Sequence

1. Verify use of acceptable IdP

SP-initiated or IdP-initiated SSO is used to produce an assertion response from the candidate IdP.

CONFIRM: SSO is successful.

2. Verify non-use of unacceptable IdP

SP-initiated or IdP-initiated SSO is used to produce an assertion response from the candidate IdP.

CONFIRM: SSO is unsuccessful based on the policy requiring the tag.

Test Case eGov2-4 - Publication

Scope

- Publication (and maintenance) of metadata via Well-Known-Location resolution profile.

Preconditions

- An http/https entityID defined that is suitable for dereferencing
- Appropriate configuration of that entityID is completed
- Multiple details of configuration are available to tester (location of a profile endpoint, a key descriptor, etc.)
- Any pre-publishing step required is completed

Test Sequence

1. Access published metadata

The entityID is dereferenced to obtain the metadata document.

CONFIRM: The metadata is available, and correctly reflects the entityID accessed, and is returned with the correct MIME type (application /xml+samlmetadata). The configuration details expected are found in the metadata.

2. Alter metadata and republish

Alter the configuration (changing an endpoint, a key descriptor, etc.) and republish, then repeat the first test.

CONFIRM: As in (1), but also that the implementation did not require a restart or disruption of service.

Test Case eGov2-5 - Import from File

Scope

- Metadata consumption via local file
- Ability to detect and ignore invalid metadata
- Support for batches (md:EntitiesDescriptor)
- Ability to update from a changed source without disruption
- Maintenance of valid operation after a change that renders a source invalid.

Preconditions

- Valid metadata is available to the implementation via a local filesystem path
- The valid metadata contains at least two md:EntityDescriptor elements inside an md:EntitiesDescriptor element
- Invalid metadata is available to the implementation via a (different) local filesystem path
- Appropriate configuration for the use of those paths is applied
- No configuration of the information supplied via metadata is in place prior to import

Test Sequence

1. Import valid metadata

The implementation is directed in whatever manner is required to import or make use of the valid metadata. A set of SAML interactions is then attempted between the implementation and the metadata subject. A basic test of SP-initiated SSO is sufficient.

CONFIRM: Operation of a defined set of SAML interactions with the metadata subject is successful based on the content of the metadata (correct endpoints used, keys used in accordance with one of the supported metadata profiles, etc.).

2. Import invalid metadata

The implementation is directed in whatever manner is required to import or make use of the invalid metadata. A set of SAML interactions is then attempted between the implementation and the metadata subject. A basic test of SP-initiated SSO is sufficient.

CONFIRM: Import and/or interaction with the metadata subject is unsuccessful.

3. Update valid metadata

The valid metadata is modified in some manner that is detectable via the interactions used to confirm successful import (changing an endpoint, a key descriptor, etc.), but remains valid. If the implementation requires manual intervention to recognize the change, this is done. The SAML interactions are repeated.

CONFIRM: The interactions remain successful but cognizant of the change(s). No restart or other service interruption was required to accomodate the change.

4. Update valid metadata with invalid change.

The valid metadata is modified in some manner that renders it invalid. If the implementation requires manual intervention to recognize the change, this is done. The SAML interactions are repeated.

CONFIRM: The interactions remain successful in accordance with the metadata that existed prior to the change. No restart or other service interruption was required to accomodate the change.

Test Case eGov2-6 - Import from URL

Scope

- Metadata consumption via multiple http and https sources
- Ability to detect and ignore invalid or unavailable metadata
- Support for caching
- Ability to update via a changed source without disruption
- Maintenance of valid operation after a change that renders a source invalid.

Preconditions

- Valid metadata is available to the implementation via at least two URLs (one http, one https)
- Invalid metadata is available to the implementation via a different, possibly unavailable, URL
- Appropriate configuration for the use of those URLs is applied
- No configuration of the information supplied via metadata is in place prior to import

Test Sequence

1. Import valid metadata

The implementation is directed in whatever manner is required to import or make use of the valid metadata. A set of SAML interactions is then attempted between the implementation and the metadata subjects (at least two, one for each source of metadata). A basic test of SP-initiated SSO is sufficient.

CONFIRM: Operation of a defined set of SAML interactions with the metadata subjects is successful based on the content of the metadata (correct endpoints used, keys used in accordance with one of the supported metadata profiles, etc.).

2. Import invalid metadata

The implementation is directed in whatever manner is required to import or make use of the invalid metadata source. A set of SAML interactions is then attempted between the implementation and the metadata subject. A basic test of SP-initiated SSO is sufficient.

CONFIRM: Import and/or interaction with the metadata subject is unsuccessful.

3. Re-import unchanged metadata

The implementation is directed in whatever manner is required to re-import/refresh the valid source of metadata, with the source maintaining the same caching indicator(s). The SAML interactions are repeated.

CONFIRM: The import resulted in no exchange of the metadata document across the network, and the interactions remain successful in accordance with the metadata that existed prior to the re-import.

4. Re-import changed metadata

The valid metadata is modified in some manner that is detectable via the interactions used to confirm successful import (changing an endpoint, a key descriptor, etc.), but remains valid. The implementation is directed in whatever manner is required to re-import/refresh the valid source of metadata. The SAML interactions are repeated.

CONFIRM: The interactions remain successful but cognizant of the change(s). No restart or other service interruption was required to accomodate the change.

5. Update valid metadata with invalid change.

The valid metadata is modified in some manner that renders it invalid. The implementation is directed in whatever manner is required to re-import/refresh the now-invalid source of metadata. The SAML interactions are repeated.

CONFIRM: The interactions remain successful in accordance with the metadata that existed prior to the change. No restart or other service interruption was required to accomodate the change.

Test Case eGov2-7 - Verification by Known Key

Scope

- Test verification of root level signature via a known key.

Preconditions

- Any MTI signature algorithm may be used.
- Valid metadata signed by a known key is available at an http or https URL.
- Valid metadata with an invalid signature is available via a different URL.
- The key should not be present inside the signature of the metadata document.
- Appropriate configuration for the use of the URLs and verification with the key is applied.
- No configuration of the information supplied via metadata is in place prior to import

Test Sequence

1. Import and verify valid metadata

The implementation is directed in whatever manner is required to import or make use of the valid metadata. A set of SAML interactions is then attempted between the implementation and the metadata subject. A basic test of SP-initiated SSO is sufficient.

CONFIRM: Operation of a defined set of SAML interactions with the metadata subject is successful based on the content of the metadata (correct endpoints used, keys used in accordance with one of the supported metadata profiles, etc.).

2. Import and (fail to) verify invalid signature

The implementation is directed in whatever manner is required to import or make use of the metadata with the invalid signature. A set of SAML interactions is then attempted between the implementation and the metadata subject. A basic test of SP-initiated SSO is sufficient.

CONFIRM: Import and/or interaction with the metadata subject is unsuccessful.

Test Case eGov2-8 - Verification by Certificate Validation

Scope

- Test verificiation of root level signature via path validation of a signing certificate.

Preconditions

- Any MTI signature algorithm may be used.
- Two certificates issued by a sample certificate authority are created, one valid, one expired.
- The certificate must be present inside the signature of the metadata document.
- Valid metadata signed by the key in the valid certificate is available at an http or https URL.
- Valid metadata signed by the key in the invalid certificate is available via a different URL.
- Appropriate configuration for the use of the URLs and verification with the issuing CA is applied.

- No configuration of the information supplied via metadata is in place prior to import

Test Sequence

1. Import and verify valid metadata

The implementation is directed in whatever manner is required to import or make use of the valid metadata. A set of SAML interactions is then attempted between the implementation and the metadata subject. A basic test of SP-initiated SSO is sufficient.

CONFIRM: Operation of a defined set of SAML interactions with the metadata subject is successful based on the content of the metadata (correct endpoints used, keys used in accordance with one of the supported metadata profiles, etc.).

2. Import and (fail to) verify invalid signature

The implementation is directed in whatever manner is required to import or make use of the metadata signed with the invalid certificate. A set of SAML interactions is then attempted between the implementation and the metadata subject. A basic test of SP-initiated SSO is sufficient.

CONFIRM: Import and/or interaction with the metadata subject is unsuccessful.

Test Case eGov2-9 - Confirm support for NameIDPolicy of 'persistent' & 'transient'

Scope:

- Verify SP possibility to specify values of 'persistent' & 'transient' for the NameIDPolicy element in an AuthnRequest

Preconditions:

- Metadata exchanged and imported
- SP and IDP configured to use HTTP-Redirect binding for AuthnRequest

Test sequence:

1. Trigger SP-initiated single sign-on using the HTTP-Redirect binding, specifying at SP that the returned identifier be 'urn:oasis:names:tc:SAML:2.0:nameid-format:persistent'
CONFIRM: SP offers capability for specifying name identifier format of "urn:oasis:names:tc:SAML:2.0:nameid-format:persistent"
2. Observe HTTP redirect parameters and decode the SAMLRequest value using the DEFLATE algorithm reversal
CONFIRM: presence of element "NameIDPolicy" with value "urn:oasis:names:tc:SAML:2.0:nameid-format:persistent" on element <samlp:AuthnRequest>

3. Authenticate to IDP using test account
4. Observe response message
CONFIRM: NameID supplied according to NameIDPolicy specified in AuthnRequest

Repeat above test sequence for 'urn:oasis:names:tc:SAML:2.0:nameid-format:transient'

Test Case eGov2-10 - IDP Discovery

Scope:

- Verify that during web-based SSO that a SP is able to establish an IDP associated with the user.

Preconditions:

- Exchange and import metadata for SP (including extension element for the discovery's response) and IDP

Test Sequence:

1. Trigger the SP to redirect the user agent to the discovery service with an HTTP GET request

CONFIRM: SP capability for specifying policy of "urn:oasis:names:tc:SAML:profiles:SSO:idp-discovery-protocol:single"

2. Allow user to interact with discovery service to select an IDP by having default setting of <IsPassive> to "false"
3. The discovery service responds by redirecting the user agent back to the requesting service provider with an HTTP GET request at the location supplied in the <return> parameter in the original request and/ or discovery response element of the metadata

CONFIRM: SP recognizes the identifier for the selected Identity Provider

4. IdP is reconfigured with <IsPassive> set to "true". CDC is deleted.

5. User attempts to interact with discovery service and select IDP but <isPassive> set to "true" prevents interaction with the user.

CONFIRM: SP returns no IDP selection

Test Case eGov2-11 - Confirm support for verification of signatures

This case checks that an IDP can receive a signed AuthnRequest and act approriately according to the validity of the signature.

Scope:

- Test consumption of a signed AuthnRequest and Assertion with and without error according to validity.

Preconditions:

- Signing and encryption keys created.
- Keys exchanged and configured via meta-data.
- SP configured to sign authentication requests
- SP and IDP configured to use HTTP-Redirect binding for AuthnRequest
- SP and IDP configured to use HTTP-POST binding for Assertion
- RSA-SHA256 signature must be used to sign requests unless otherwise specified.
- AES256-CBC with RSA-OAEP must be used to encrypt the <saml2:EncryptedAssertion> in the <saml2p:Response> message unless otherwise specified.
- SP meta-data may specify separate <md:KeyDescriptor> for signing and encryption

Test sequence:

Test IdP for signature verification.

1. Configure IdP to require signed AuthnRequests

2. Trigger SP-initiated single sign-on using the HTTP-Redirect binding
3. Sign AuthnRequest with RSA-SHA1
4. Observe IDP handling of the request

CONFIRM: IDP accepts AuthnRequest message without error

5. Trigger SP-initiated single sign-on using the HTTP-Redirect binding
6. Sign AuthnRequest with RSA-SHA256
7. Observe IDP handling of the request

CONFIRM: IDP accepts AuthnRequest message without error

8. Insert new signature public key in the IdP (but do not change the private key on the SP)
9. Trigger SP-initiated single sign-on using the HTTP-Redirect binding
10. Sign AuthnRequest with RSA-SHA256
11. Observe IDP handling of the request

CONFIRM: IDP rejects AuthnRequest due to invalidity of the signature

12. Trigger SP-initiated single sign-on using the HTTP-Redirect binding
13. Don't sign AuthnRequest
14. Observe IDP handling of the request

CONFIRM: IDP rejects AuthnRequest due to invalidity of the signature

15. Re-Insert correct signature public key from IDP
16. Configure IdP to only accept RSA-SHA1 AuthnRequests
17. Trigger SP-initiated single sign-on using the HTTP-Redirect binding
18. Sign AuthnRequest with RSA-SHA1
19. Observe IDP handling of the request

CONFIRM: IDP rejects AuthnRequest due to invalidity of the signature

Test SP for signature verification

20. Configure IdP NOT to require signed AuthnRequests
21. Trigger SP-initiated single sign-on using the HTTP-Redirect binding
22. Sign <saml:Assertion> with RSA-SHA1
23. Observe SP handling of the request

CONFIRM: SP accepts <saml:Assertion> message without error

24. Configure SP not to accept RSA-SHA1
25. Trigger SP-initiated single sign-on using the HTTP-Redirect binding
26. Sign <saml:Assertion> with RSA-SHA1
27. Observe SP handling of the request

CONFIRM: SP rejects <saml:Assertion> due to invalidity of the signature

28. Trigger SP-initiated single sign-on using the HTTP-Redirect binding
29. Sign <saml:Assertion> with RSA-SHA256
30. Observe SP handling of the request

CONFIRM: SP accepts <saml:Assertion> message without error

31. Trigger SP-initiated single sign-on using the HTTP-Redirect binding
32. Don't sign <saml:Assertion>
33. Observe SP handling of the request

CONFIRM: SP rejects <saml:Assertion> due to invalidity of the signature

34. Change the IdP's signing key in the meta-data imported by the SP so that it no longer matches the IdP's private key
35. Trigger SP-initiated single sign-on using the HTTP-Redirect binding
36. Sign <saml:Assertion> with RSA-SHA256
37. Observe SP handling of the request

CONFIRM: SP rejects <saml:Assertion> due to invalidity of the signature

Test Case eGov2-12 - Confirm Service Provider support for AuthnRequest attributes AssertionConsumerServiceURL, ProtocolBinding, and AttributeConsumingServiceIndex

Scope:

- Verify possibility to specify, and presence of, the AuthnRequest attributes AssertionConsumerServiceURL, ProtocolBinding, and AttributeConsumingServiceIndex

Preconditions:

- Metadata exchanged and imported
- Service provider metadata contains indexed AssertionConsumerService entries
- Service provider metadata contains indexed AttributeConsumingService entries
- SP and IDP configured to use HTTP-Redirect binding for AuthnRequest

Test sequence:

1. Trigger SP-initiated single sign-on using the HTTP-Redirect binding, specifying that a particular AssertionConsumerServiceURL be called using a specified ProtocolBinding
CONFIRM: SP offers capability for specifying the AssertionConsumerService URL
CONFIRM: SP offers capability for specifying the ProtocolBinding to be used
2. Observe HTTP redirect parameters and decode the SAMLRequest value using the DEFLATE algorithm reversal
CONFIRM: presence of attribute "AssertionConsumerServiceURL" with the URL specified in step 1 as value, on element <samlp:AuthnRequest>
CONFIRM: presence of attribute "ProtocolBinding" with the URL specified in step 1 as value, on element <samlp:AuthnRequest>

3. Trigger SP-initiated single sign-on using the HTTP-Redirect binding, specifying that a particular AttributeConsumingServiceIndex be called
CONFIRM: SP offers capability for specifying the AttributeConsumingServiceIndex value
4. Observe HTTP redirect parameters and decode the SAMLRequest value using the DEFLATE algorithm reversal
CONFIRM: presence of attribute "AttributeConsumingServiceIndex" with the index specified in step 3 as value, on element <samlp:AuthnRequest>

Test Case eGov2-13 - Confirm Identity Provider verification of AssertionConsumerServiceURL value

Scope:

- Verify IDP handling of AuthnRequest attributes AssertionConsumerServiceURL and ProtocolBinding

Preconditions:

- Metadata exchanged and imported
- Service provider capable of sending AuthnRequest messages with AssertionConsumerServiceURL and ProtocolBinding attributes
- Service provider metadata contains AssertionConsumerService entries
- IDP configured to provide attribute response
- Valid and known account that can be authenticated using an available authentication method on IDP

Test sequence:

1. Trigger SP-initiated single sign-on specifying an AssertionConsumerServiceURL and ProtocolBinding that matches values in SP metadata
2. Observe IDP behaviour
CONFIRM: IDP accepts AuthnRequest without error

3. Trigger SP-initiated single sign-on specifying an AssertionConsumerServiceURL and ProtocolBinding that do not match values in SP metadata
4. Observe IDP behaviour
CONFIRM: IDP responds with appropriate error message

Test Case eGov2-14 - Responses to Authentication Failure

NOTE: This is an error test case. It does not need "full-matrix" testing but simply pairing of participants to insure it is tested once with each participant.

To complete this Test Case, the IdP under test must receive an authentication request for a User it cannot or will not authenticate. The cause of this authentication failure is not relevant but is expected to be an event such as:

- The user chooses to cancel the authentication process.

- The user identity does not exist or the number of failed login attempts has been exceeded.
- The user forgets his/her password and must wait for an email containing the password.

Preconditions

- Metadata exchanged and loaded
- Encryption disabled
- User Identities Not Federated

Test Sequence

1. AuthnRequest from SP to IdP, Redirect Binding, Federate

User/SP attempts Single Sign-On with Persistent Name Identifier with AllowCreate set to true. SP communication to the IdP for the SAML request is through HTTP-Redirect binding. IdP does not recognize User and thus cannot authenticate user.

IdP CONFIRM: User is not authenticated.

2. Response Failure

Being unable to authenticate User, IdP returns SAML Response with error indicating AuthnRequest failed.

SP CONFIRM: IdP returns SAML Response indicating authentication error.

Test Case eGov2-Extended-1 - Confirm Proxying Identity Provider supports “passthrough” for RequestedAuthnContext and NameIDPolicy

NOTE: This test case likely needs to be removed or deferred to future tests as we were not able to create a Holder of Key Single Sign-On test case and thus can not fully test eGov2-Full conformance mode which also requires Proxying test cases.

Background

The objective is to test the ability of an implementation to be configured to provide the mapping options required by the eGov2.0 profile. The eGov requirements for 2.7.1 Proxying Authentication Requests and 2.7.2 Proxying Responses can be combined into a single set of test cases.

The software under test would need to be configured for 3 different tests as a Proxying IdP:

1. to provide “passthrough” or no mapping of RequestedAuthnContext and NameIDPolicy on the authnRequest and the equivalent “passthrough” on the Assertion Response
2. to provide a “pre-defined” mapping of RequestedAuthnContext and on the authnRequest and the reverse equivalent “pre-defined” mapping on the Assertion Response. This will require a pre-defined mapping which will be from one federation’s LoA’s to another federation’s LoA’s
3. to test the ability of the Proxying IdP to be configured to remove the original RequesterID when it reforms the authentication request for the real IdP and the removal of the AuthenticatingAuthority when it creates the Assertion Response for the requesting SP.

Since these are tests of a Proxying IdP, a setup similar to the Test Case “F” in [3.2.2] would be required. In this setup, both the SP and the target IdP could be supporting implementations, with only the Proxying IdP being under test. The **CONFIRM** would be accomplished by observing the authentication requests received at the supporting IdP and the Assertion Responses received at the originating SP.

Scope:

- Verify that the Proxying IDP sends the exact same RequestedAuthnContext and NameIDPolicy attributes to the IdP as it received from the SP.

Preconditions:

- Metadata exchanged and imported
- Encryption disabled
- User Identities Not Federated
- SP and IdP provided as test support implementations
    - to send AuthnRequest messages with NameIDPolicy and RequestedAuthnContext
    - SP sends (or defaults) proxycount=1
- IdP configured to support LoA’s
    - Foo1.example.com/assurance/loa2
    - Foo1.example.com/assurance/loa3
    - Foo2.example.com/assurance/loa2
- Proxying IdP (PIdP), under test, configured to support “passthrough”

Test Sequence

Step 1: AuthnRequest from SP to PIdP, Redirect Binding, Federate

User/SP does Single Sign-On with RequestedAuthnContext set to exact “Foo1.example.com/assurance/loa2”. SP communication to the PIdP for the SAML Authentication Request is through HTTP Redirect binding. PIdP cannot authenticate the User but recognizes it can proxy the AuthnRequest to IdP.

- PIdP **CONFIRM**: User is not authenticated.
- PIdP **CONFIRM**: AuthnRequest contains RequestedAuthnContext set to exact “Foo1.example.com/assurance/loa2”.

Step 2: AuthnRequest from PIdP to IdP, Redirect Binding, Federate

PIdP proxies AuthnRequest to IdP through HTTP Redirect binding.

- IdP **CONFIRM**: Receives AuthnRequest from PIdP.
- IdP **CONFIRM**: AuthnRequest contains RequestedAuthnContext set to exact “Foo1.example.com/assurance/loa2”.

Step 3: Assertion Response from IdP to PIdP, POST binding

User provides assigned credentials to IdP for authentication. IdP provides assertion of User and returns a signed SAML Response message to PIdP through HTTP POST binding.

- PIdP **CONFIRM**: Receives SAML Response through HTTP POST binding.
- PIdP **CONFIRM**: Valid assertion is returned from IdP.
- PIdP **CONFIRM**: <AuthnStatement> contains < AuthnContext> set to “Foo1.example.com/assurance/loa2”.

Step 4: Assertion Response from PIdP to SP, POST binding

PIdP creates its own assertion for the User from the assertion it received from IdP and returns a signed SAML Response message to SP through HTTP POST binding.

- SP **CONFIRM**: Receives SAML Response through HTTP POST binding.
- SP **CONFIRM**: Valid assertion is returned from IdP_A.
- SP **CONFIRM**: <AuthnStatement> contains < AuthnContext> set to “Foo1.example.com/assurance/loa2”.

Test Case eGov2-Extended-2 - Confirm Proxying Identity Provider supports a “pre-defined” mapping of RequestedAuthnContext

NOTE: This test case likely needs to be removed or deferred to future tests as we were not able to create a Holder of Key Single Sign-On test case and thus can not fully test eGov2-Full conformance mode which also requires Proxying test cases.

Scope:

- Verify that the Proxying IDP maps the RequestedAuthnContext attribute sent to the IdP correctly based on the RequestedAuthnContext it received from the SP.

Preconditions:

- Metadata exchanged and imported
- Encryption disabled
- User Identities Not Federated
- SP and IdP provided as test support implementations
  - to send AuthnRequest messages with NameIDPolicy and RequestedAuthnContext
  - SP sends (or defaults) proxycount=1
- IdP configured to support LoA’s
  - Foo1.example.com/assurance/loa2
  - Foo1.example.com/assurance/loa3
  - Foo2.example.com/assurance/loa1
  - Foo2.example.com/assurance/loa2
- Proxying IdP (PIdP), under test, configured to support predefined mapping as follows:
  - Foo1.example.com/assurance/loa2 maps to Foo2.example.com/assurance/loa1
  - Foo1.example.com/assurance/loa3 maps to Foo2.example.com/assurance/loa2
  - All other RequestedAuthnContext’s are rejected as unsupported

Test Sequence

Step 1: AuthnRequest from SP to PIdP, Redirect Binding, Federate

User/SP does Single Sign-On with RequestedAuthnContext set to exact “Foo1.example.com/assurance/loa2”. SP communication to the PIdP for the SAML Authentication Request is through HTTP Redirect binding. PIdP cannot authenticate the User but recognizes it can proxy the AuthnRequest to IdP.

- PIdP **CONFIRM**: User is not authenticated.
- PIdP **CONFIRM**: AuthnRequest contains RequestedAuthnContext set to exact “Foo1.example.com/assurance/loa2”.

Step 2: AuthnRequest from PIdP to IdP, Redirect Binding, Federate

PIdP proxies AuthnRequest to IdP through HTTP Redirect binding.

- IdP **CONFIRM**: Receives AuthnRequest from PIdP.
- IdP **CONFIRM**: AuthnRequest contains RequestedAuthnContext set to exact “Foo2.example.com/assurance/loa1”.

Step 3: Assertion Response from IdP to PIdP, POST binding

User provides assigned credentials to IdP for authentication. IdP provides assertion of User and returns a signed SAML Response message to PIdP through HTTP POST binding.

- PIdP **CONFIRM**: Receives SAML Response through HTTP POST binding.
- PIdP **CONFIRM**: Valid assertion is returned from IdP.

- PIdP **CONFIRM**: <AuthnStatement> contains < AuthnContext> set to “Foo2.example.com/assurance/loa1”.

Step 4: Assertion Response from PIdP to SP, POST binding

PIdP creates its own assertion for the User from the assertion it received from IdP and returns a signed SAML Response message to SP through HTTP POST binding.

- SP **CONFIRM**: Receives SAML Response through HTTP POST binding.
- SP **CONFIRM**: Valid assertion is returned from PIdP.
- SP **CONFIRM**: <AuthnStatement> contains < AuthnContext> set to “Foo1.example.com/assurance/loa2”.

Test Case eGov2-Extended-3 - Confirm Proxying Identity Provider supports configuration to remove the original RequesterID and returned AuthenticatingAuthority

Scope:

- Verify that the Proxying IDP can be configured to remove the original RequesterID when it reforms the authentication request for the IdP and the removal of the AuthenticatingAuthority when it creates the Assertion Response for the requesting SP.

Preconditions:

- Metadata exchanged and imported
- Encryption disabled
- User Identities Not Federated
- SP and IdP provided as test support implementations
    - to send AuthnRequest messages with RequesterID
    - SP sends (or defaults) proxycount=1
- IdP configured to mask out RequesterID and AuthenticatingAuthority

Test Sequence

Step 1: AuthnRequest from SP to PIdP, Redirect Binding, Federate

User/SP does Single Sign-On with RequestedAuthnContext set to exact “Foo1.example.com/assurance/loa2”. SP communication to the PIdP for the SAML Authentication Request is through HTTP Redirect binding. PIdP cannot authenticate the User but recognizes it can proxy the AuthnRequest to IdP.

- PIdP **CONFIRM**: User is not authenticated.
- PIdP **CONFIRM**: AuthnRequest contains RequesterID set to ID of SP

Step 2: AuthnRequest from PIdP to IdP, Redirect Binding, Federate

PIdP proxies AuthnRequest to IdP through HTTP Redirect binding.

- IdP **CONFIRM**: Receives AuthnRequest from PIdP.
- IdP **CONFIRM**: AuthnRequest contains RequesterID set to ID of PIdP, not SP

Step 3: Assertion Response from IdP to PIdP, POST binding

User provides assigned credentials to IdP for authentication. IdP provides assertion of User and returns a signed SAML Response message to PIdP through HTTP POST binding.

- PIdP **CONFIRM**: Receives SAML Response through HTTP POST binding.
- PIdP **CONFIRM**: Valid assertion is returned from IdP.
- PIdP **CONFIRM**: <AuthnStatement> contains AuthenticatingAuthority set to ID of IdP.

Step 4: Assertion Response from PIdP to SP, POST binding

PIdP creates its own assertion for the User from the assertion it received from IdP and returns a signed SAML Response message to SP through HTTP POST binding.

- SP **CONFIRM**: Receives SAML Response through HTTP POST binding.
- SP **CONFIRM**: Valid assertion is returned from PIdP.
- SP **CONFIRM**: <AuthnStatement> contains AuthenticatingAuthority set to ID of PIdP, not IdP.

Signature and Encryption Requirements

eGov entities must support and test under the following security requirements. For security requirements allowing choices, eGov entities must support any option choosen by other test participants

Issuance (IdP) and acceptance (SP) of signed <saml2:Assertion> using http://www.w3.org/2001/04/xmldsig-more#rsa-sha256 (2048bit RSA key) or http://www.w3.org/2000/09/xmldsig#rsa-sha1 (2048bit RSA key).

Issuance (IdP) and acceptance (SP) of <saml2:EnctyptedAssertion> using either http://www.w3.org/2001/04/xmlenc#tripledes-cbc or http://www.w3.org/2001/04/xmlenc#aes128-cbc or http://www.w3.org/2001/04/xmlenc#aes256-cbc. The key transports choices are http://www.w3.org/2001/04/xmlenc#rsa-1_5 or http://www.w3.org/2001/04/xmlenc#rsa-oaep-mgf1p.