Authentication request: bindings, signing and options

When a user logs in, the SP will send a SAML 2.0 authentication request (AuthnRequest) to SURFconext. Below you will find detail information about the supported bindings, the signing and the available options.

- Supported bindings
- Signing
- Authentication request options
  - AssertionConsumerServiceIndex
  - AssertionConsumerServiceURL
  - ForceAuthn
  - IsPassive
  - NameIDPolicy
    - AllowCreate
    - Format
  - RequestedAuthnContext
  - Scoping
    - RequesterID
    - IDPList
      - Navigate

Supported bindings

SURFconext supports two types of bindings:

- urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Redirect (preferred in accordance to SAML2int profile)
- urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST (can only be used in case your software doesn't support HTTP-Redirect)

Signing

By default, the signature on AuthnRequests is ignored. Some use cases and features require AuthnRequest however to be signed. Contact SURFconext support to enable this.

- Currently SHA256 signing is supported:
  - http://www.w3.org/2001/04/xmldsig-more#rsa-sha256.
- The SP must provide the public RSA key required for verification of the signature out of band, preferably using SAML metadata.
- The key must be provided an X.509 certificate in PEM format.
- Certificates should be self signed and valid (= not expired).
- SURFconext retrieves only the public key from the certificate. No other validation is done.
- The public modulus of the RSA key must be 2048 or 4096 bits.
- SURFconext can register two trusted certificates per SP.

Authentication request options

You can use the following options in AuthnRequest:

AssertionConsumerServiceIndex

This method is not allowed in the saml2int profile (see above). Use AssertionConsumerServiceURL instead.

AssertionConsumerServiceURL

This attribute is used to specify an alternate AssertionConesumerService Location for the resulting SAML response. This can be any location from the SP metadata registered in SURFconext. When signature verification of AuthnRequest is enabled, the SP is allowed to request any AssertionConesumerService Location.

If no AssertionConsumerService is specified, SURFconext will use the default from the SP metadata registered in SURFconext.

ForceAuthn

With the ForceAuthn parameter you can disable single sign-on. If a user is already logged in and he tries to log in for another service at SURFconext, he will have to authenticate again at his home IP.

To disable single sign-on both AuthnRequests must be signed: (1) the AuthnRequest from SP to SURFconext and (2) the resulting AuthnRequest from SURFconext to the home IP. By default signing is disabled: contact SURFconext support to enable it. SURFconext can also contact the home IPs involved: they must enable signing of AuthnRequests on their side.

IsPassive

Partially supported. When receiving an authnrequest with an IsPassive flag, SURFconext will, in accordance with the SAML specification:

- Return NoPasssive if a WAYF would have been presented (multiple IdPs whitelisted for SP and no IdP preselected in AuthnRequest or SSOLocation), else,
- Return NoPassive if the IdP returned a NoPassive response,
- Return an assertion if the IdP returned an assertion.

Support is partial in the sense that currently no response is returned to an IsPassive flag when authentication is stopped at SURFconext because the user needs to give consent, a SURFconext authorization rule forbids login, or second factor authentication for the user is required by SURFconext. For these cases, SURFconext will not return a response, also not an IsPassive response; the flow stops.

NameIDPolicy

AllowCreate

The value of AllowCreate is ignored.

Format

You can specify a format attribute in the NameIDPolicy parameter:

- If specified and if the format is in the list of allowed NameIDFormats it will be used. Otherwise the default NameID format (urn:oasis:names:tc:SAML:2.0:nameid-format:transient only) will be used.
- If not specified, the default format will be used.

RequestedAuthnContext

If it contains a single value with a known SURFsecureID LoA, Engineblock will ensure the user authenticates with this or a higher SURFsecureID LoA and return the attained LoA identifier. See Requesting SURFsecureID in authentication requests.

On request we can enable it to be transparent, and any value set will be transparently copied to the Identity Provider who may or may not act on it.
If not, or any other value is requested, it is ignored.

Scoping

This element is used to restrict the IdPs available though a proxying IdP (like SURFconext). The only elements supported are RequesterID and IDPList:

RequesterID

This element can be used for a SP performing authentication on behalf of another SP (SP proxy). RequesterID contains the EnityID of a SP. This EntityID must be registered at SURFconext and meet all requirements for a SP, except the SAML endpoint information (e.g. AssertionConsumerService Location). The IdPs available are restricted to the ones that have access to both SPs. For a SP to be in that selection rely it needs to ensure that a malicious user can not circumvent the restriction by modifying the AuthnRequest, which can be done in two ways:

1. Signing the AuthnRequest and requesting SURFconext to enable signature validation for the proxying SP.
2. Verifying that the EntityID in the AuthenticatingAuthority element in the AuthnContext in the SAML Assertion is from an IdP that is allowed access. For a list with EntityIDs of the IdPs that are allowed access to your SP: https://engine.surfconext.nl/authentication/proxy/idps-metadata?sp-entity-id=*<the EntityID of your SP>*

IDPList

This element defines the IdPs shown in the SURFconext WAYF. It contains one or more IDPEntry elements with the entityID of the IdP in the ProviderID attribute. The entiyIDs of the available IdPs can be found in the IdPs metadata.

If IDPList contains only one IdP, the WAYF will be skipped and the user will be directed to that IdP.

Note that there is no guarantee that the user will actually be authenticated by one of the IdPs in the IDPList. The IdP used to authenticate the user can be found in the AuthenticatingAuthority element in the AuthnContext in the SAML Assertion.

An SAML AuthenRequest using Scoping with an IDPList

```
<samlp:AuthnRequest xmlns:samlp="urn:oasis:names:tc:SAML:
2.0:protocol"
    xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
    ID="_123456789097da188e1e4016be09fb13c422553cea"
Version="2.0"
    IssueInstant="2015-02-25T14:13:02Z"
    Destination="https://engine.surfconext.nl/authentication
/idp/single-sign-on"
    AssertionConsumerServiceURL="https://profile.surfconext.
nl/simplesaml/module.php/saml/sp/saml2-acs.php/default-sp"
    ProtocolBinding="urn:oasis:names:tc:SAML:2.0:bindings:
HTTP-POST">
    <saml:Issuer>https://profile.surfconext.nl/simplesaml
/module.php/saml/sp/metadata.php/default-sp</saml:Issuer>
    <samlp:NameIDPolicy Format="urn:oasis:names:tc:SAML:2.0:
nameid-format:transient" AllowCreate="true"/>
    <samlp:Scoping>
        <samlp:IDPList>
            <samlp:IDPEntry ProviderID="https://idp.surfnet.
nl"/>
            <samlp:IDPEntry ProviderID="http://authentigate.
surfmarket.nl/adfs/services/trust"/>
        </samlp:IDPList>
    </samlp:Scoping>
</samlp:AuthnRequest>
```

Dutch scoping

When a SP cannot support scoping, also the "transparent metadata" at https://engine.surfconext.nl/authentication/proxy/idps-metadata can be used to select a IdP instead of the SURFconext WAYF. This method relies on each IdP having a different SSO Location. A typical use case is a SP having its own WAYF or discovery service (Dutch scoping).

Navigate