IDP 3 Metadata configuration

Recommended configuration for adding Metadata to a Shibboleth IDP v3 for the participation in eduID.at.

> All eduID.at IDPs can use this configuration, even those not yet participating in Interfederation/eduGAIN.

See Metadata for more info. This page has the following sections:

eduID.at metadata

In the Shibboleth IDP's `/opt/shibboleth-idp/conf/metadata-providers.xml` add the following new `MetadataProvider`, *before* the file's closing `</MetadataProvider>` on the last line. The resulting complete file should look like this example (which also contains a commented out `MetadataProvider` for managing your own, local (non-eduID.at) Service Providers, see below).

> Note that the following configuration already contains the *public key* component of the eduID.at Metadata Signing certificate in a `SignatureValidation` filter. This ensures that only SAML 2.0 metadata that is signed with a key that matches *this* public key is accepted as valid and authentic, without any references to external X.509 certificate files in the file system.

MetadataProvider element for conf/metadata-providers.xml

```
<MetadataProvider id="eduID.at"
    xmlns="urn:mace:shibboleth:2.0:metadata"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="urn:mace:shibboleth:2.0:metadata http://shibboleth.net/schema/idp/shibboleth-metadata.
xsd"
    xsi:type="FileBackedHTTPMetadataProvider"
    metadataURL="https://eduid.at/md/aconet-interfed.xml"
    backingFile="%{idp.home}/metadata/federation-metadata.xml">
    <MetadataFilter xsi:type="RequiredValidUntil" maxValidityInterval="P28D" />
    <MetadataFilter xsi:type="SignatureValidation" requireSignedRoot="true">
        <PublicKey>
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAwSDLS25Y5spmrB8fykSq
zXbTaEssR/cD12foFIDoLwN5PUEwXWWPsl8zXyoFw2nWlrORK47Z8wjU1QlO4BcZ
Rt8ix52GJXs9Q5xgBs4zE/Xp6hgUBa0if2PxOWoA2UTqUgBj8L6joVkz5rBeiY7J
2CkfvRw+QSzkMm+YEsmAcwpyghavKfDvYSOxubuYBacqkwGa0J8AkDuiG3kfpydr
CE5R8KTt8P65Xie5+g8YCU0mql1vCzD0O48y5dK5SHD4PhkpG2BAayGiNUR7bDSk
VElb3uybwjb0BQI+q0hu4NqpeZjTY0pTnu5oZhQW49e4M+gKJEoSUceI3CSZ6nSf
HwIDAQAB</PublicKey>
    </MetadataFilter>
    <MetadataFilter xsi:type="EntityRoleWhiteList">
      <RetainedRole>md:SPSSODescriptor</RetainedRole>
    </MetadataFilter>
</MetadataProvider>
```

Maybe also remove any of the existing metadata examples from that file for better clarity. (But don't remove the wrapping "MetadataProvider" at the top of the file nor its closing tag at the very end of the file!)

Other metadata

Here be dragons

> **Skip this section** unless you *know* you have a need to federate with SAML Service Providers that are not and cannot become eduID.at members and are not available via Interfederation/eduGAIN either! (Check with REFEDS MET or the eduGAIN entities database. Or contact the ACOnet community and/or support team.)

There are cases that require use of SAML 2.0 Metadata that has not been verified, improved and signed by the ACOnet Identity Federation operator: If a desired service is not available in eduID.at nor via eduGAIN you'll have to take on duties and work that otherwise ACOnet would have performed for you. But managing such local or bilateral relations can be dangerous and completely insecure unless you understand the trust model and security issues involved. So always proceed with caution, pay attention to best practices and ask first when in doubt!

Don't trust metadata loaded over the network unless you have good reason to!

The above method of loading eduID.at metadata from a remote URL is secure only because the eduID.at metadata is regularly signed cryptographically *and* each signed copy contains an expiration date in the near future (days to few weeks) *and* signatures on that metadata are automatically verified with a trusted key before accepting the metadata in the configuration presented above. Loading any other metadata over the network automatically **is not secure**: Among other things metadata contains protocol endpoints where an IDP would send personal data to (which should be inspected and verified by a human before use, the way this has been done for the eduID.at metadata by the ACOnet Team) as well as cryptographic keys that will be used to secure protocol messages exchanged later. But if you automatically imported cryptographic keys over the network that are later used to verify protocol messages exchanged over the network there's no trust anchor and therefore no security in that: That'd be like automatically regularly importing X.509/TLS Certificate Authority certificates over the network and blindly trusting the content of what you've imported to verify the authenticity and integrity of other protocol messsages. I.e., using SAML with remote, unsigned metadata (or not validating signed remote metadata with a *previously established, trusted key*) is simply Security Theatre!

Yet more reasons not to trust remote metadata

Also note that loading SAML 2.0 metadata from a remote URL – metadata that has not been checked and curated by your trusted local federation operator or by one of our peer federations – may include all kinds of entities, endpoints or requests for personal data you don"t expect at that URL (or stuff that simply *wasn't there when you looked at that URL, once*)! While there are ways to limit that risk (e.g. by filtering such remote metadata for only the *expected* entities) often the best way to deal with the underlying issues is to *not* automatically load such metadata from remove URLs at all. Therefore we document a method below that requires downloading and manually verifying remote metadata once and then putting that checked metadata into a local file (or directory) that's not updated automatically anymore (essentially creating a snapshot of the remote metadata). This way you're trading the security issues of improper metadata exchange for having to manually update those "snapshots" of metadata manually every once in a while when deemed necessary.
As an added bonus locally managing a copy of that remote metadata allows you to tune/fix the metadata so that it reflects the Service Provider's actual requirements (or your deployment preferences with regard to that service), e.g. listing the RequestedAttribute elements you intend to release to that service (and with the right NameFormat), listing the correct/preferred NameIDFormat, add MDUI elements to get a proper service name and logo shown on the IDP login page, and so on, including cases where the metadata provided at the remote URL is incomplete, unsuitable or simply wrong (i.e., always).

How to manage metadata for non-federated Service Providers

Since multi-party federation is a concept foreign to most commercial Service Providers -- or maybe because doing it properly is simply *too hard for the biggest and richest corporations on the planet*, even though we all can do it using Free/Libre/Open Source software – you will probably also have to manage SAML 2.0 Metadata for Service Providers that are not available in eduID.at or other trusted federation metadata.

Commonly the Service Provider's SAML integration is simply broken in that they create IDP-specific SPs that cannot be federated in any meaningful way: One logical SP for each IDP, instead of simply using *one* SAML SP – for their *one* service – with as many IDPs as needed. Sometimes the Service Provider may be ignorant of federations and will ask institutions to exchange metadata bilaterally for no good reason, in which case the SP may still be persuaded to join the federation instead. Often bilateral metadata exchange is done in a "self-service console/portal" way, though, that pushes all the integration work to each institution's IDP admin, combined with the SP not supporting any of the features SAML provides for security and/or scalability. And of course such Service Providers' documentation for their SAML interface either doesn't exist or might as well not exist because it's incomplete, inaccurate, unclear or just plain wrong.

Anyway, one way to manage these non-federated Service Provider's SAML 2.0 Metadata (which you sometimes even have to create yourself on their behalf) is to put all such SPs into a *single file* in your IDP's /opt/shibboleth-idp/conf/metadata/ directory and reference that within your `metadata-providers.xml` configuration, like in the attached example configuration file:

MetadataProvider element for conf/metadata-providers.xml

```
<MetadataProvider id="LocalMetadata" xsi:type="FilesystemMetadataProvider" metadataFile="%{idp.home}/metadata
/local-sps.xml"/>
```

The *content* of that `local-sps.xml` metadata file could look like this initially (replace "`example.org`" in the first line with your own domain):

local-sps.xml

```
<EntitiesDescriptor Name="http://example.org/local-entities"
  xmlns="urn:oasis:names:tc:SAML:2.0:metadata"
  xmlns:md="urn:oasis:names:tc:SAML:2.0:metadata"
  xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
  xmlns:disco="urn:oasis:names:tc:SAML:profiles:SSO:idp-discovery-protocol"
  xmlns:idpdisc="urn:oasis:names:tc:SAML:profiles:SSO:idp-discovery-protocol"
  xmlns:init="urn:oasis:names:tc:SAML:profiles:SSO:request-init"
  xmlns:mdui="urn:oasis:names:tc:SAML:metadata:ui"
  xmlns:mdrpi="urn:oasis:names:tc:SAML:metadata:rpi"
  xmlns:mdattr="urn:oasis:names:tc:SAML:metadata:attribute"
  xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
  xmlns:xenc="http://www.w3.org/2001/04/xmlenc#"
  xmlns:alg="urn:oasis:names:tc:SAML:metadata:algsupport">

<!-- Add any SP's metadata here, one EntityDescriptor after the other -->

</EntitiesDescriptor>
```

Then for each new SP you have to manage local metadata for just copy/paste their EntityDescriptor into that file, after the opening and before the closing `EntitiesDescriptor` tag.

> **Always check metadata after editing**
>
> Be sure to check the metadata file after editing: `xmlwf` checks well-formedness, `xmllint` can check for XSD-schema validity, XmlSecTool can do it all. See also MetadataCorrectness in the Shibboleth wiki.

> If you have to manage metadata for more than a dozen or two SPs you should consider using the LocalDynamicMetadataProvider instead of the FilesystemMetadataProvider documented above. That works by dropping individual metadata files (one for each SP) into a common directory. Details in the documentation.

Activate changed metadata configuration

Then reload the IDP's Metadata service to make the new `MetadataProvider` configuration active.

> Reloading the Metadata service as shown immediately below is only necessary in order to activate changes to your `/opt/shibboleth-idp/conf/metadata-providers.xml` configuration, *not* for changes to the metadata itself. So ideally you'd do this only once and be done with it. If you find youself adding new MetadataProviders all the time (e.g. one for each new SAML SP) You're Doing It Wrong™ and should consider the alternatives documented here.

```
/opt/shibboleth-idp/bin/reload-service.sh -id shibboleth.MetadataResolverService
```

That should result in the message "Configuration reloaded." being shown in your shell, a "Service 'shibboleth.MetadataResolverService': Reload complete" log entry in `/opt/shibboleth-idp/logs/idp-process.log`, as well as a web server log entry in `/var/log/tomcat9/access.log`. You should also see a new metadata backup file (with the file name configured above in the parameter `backingFile`) being generated in `/opt/shibboleth-idp/metadata/`.

The configured `id="eduID.at"` MetadataProvider will regularly and automatically reload and signature-validate eduID.at metadata, ensuring you always have current and authentic metadata available. Though if you wanted you *could* also manually trigger reloads of the configured eduID.at metadata from the command line, e.g. in the exceptional (and hypothetical) case you've been made aware of a service that was removed from eduID.at metadata for security reasons but your IDP is still using cached metadata containing that service.
In normal use of the IDP *doing this is not necessary*, though:

```
/opt/shibboleth-idp/bin/reload-metadata.sh -id eduID.at
```

The `id` parameter's value needs to match the configured `MetadataProvider`'s id XML attribute value and only that metadata will be reloaded.

Finally, in case you're also using the `id="LocalMetadata"` MetadataProvider to manage your local (and bilateral) Service Provider metadata you can manually reload metadata for those entities the same way: Because that metadata is "static" (it does not change unless you change it and it's not loaded over the network from where others would change it) the MetadataProvider for "LocalMetadata" does not contain any provisions for regular automatic reloading. As a consequence you will have to reload that specific metadata yourself each time you have changed the referenced `/opt/shibboleth-idp/metadata/local-sps.xml` metadata document:

```
/opt/shibboleth-idp/bin/reload-metadata.sh -id LocalMetadata
```

One could configure the IDP to reload that metadata automatically, too, but since loading incorrect metadata might break the IDP doing so automatically with self-managed metadata is *not recommended*. The ACOnet Team has extensive experience with curating and validating SAML Metadata for publication as part of the eduID.at metadata and so reloading that automatically is safe. The same may not be the case for the metadata you manage yourself, therefore we suggest adopting the method of manual reloading after having changed (and verified, see note "Always check metadata after editing" above) the metadata.