SPRelyingConfig

The basic unit for defining the other half of exchanges with the SP is the relying party. A relying party can be comprised of an entire federation, a single enterprise, or even one application, and is treated the same way by the service provider. Any specifications made regarding a relying party will be applied for all transactions to which it applies on a closest-match basis. This allows for more specific definitions for individual relying parties within a broader relying party, e.g. one IdP in a federation.

The `RelyingParty` element is used to customize the set of credentials used for one or more relying parties that this SP must recognize. This may include any federations the SP is a member of, any IdP's that have established bilateral agreements with the SP, or any other trust structure that SP must be aware of.

`<RelyingParty name="`*`string`*`" TLS="`*`string`*`" Signing="`*`string`*`">`	One or more `RelyingParty` elements may be contained by a `CredentialUse` element to enumerate relying parties for which a distinct set of credentials should be used. The `TLS` and `Signing` attribute values reference the identifiers of credential resolvers defined in `CredentialsProvider` elements.

- `name`: Identifies the origin site or group of sites to which the credentials specified in the element apply. This is used to match the providerId sent within attribute assertions from origin sites against a set of "groups" based on metadata.
- `TLS`: Specifies a set of signing credentials to be used in TLS mutual authentication with IdPs for attribute query requests.
- `Signing`: Specifies a set of signing credentials to be used for signing authentication and attribute query assertions.

Federations

Federations are collections of IdPs and SPs that agree to exchange information according to a set of rules with a list of acceptible root certificate authorities, often utilizing a common set of attributes. Membership in a federation does not limit bi-lateral agreements with other sites inside or outside the federation, although the political and mechanical framework of the federation can make these interactions easier.

Shibboleth supports the use of federations in order to simplify trust interactions and policy in support of these exchanges. It is not necessary to be part of a Federation to utilize Shibboleth, nor is a federation necessarily confined to supporting only Shibboleth. A federation from the point of view of a service provider is just a construct that behaves in many ways like any other relying party.

Most federations will distribute a `sites.xml` file to facilitate trust interactions which must be pointed to by `shibboleth.xml`. This file identifies certificates and root CAs that are considered authoritative for that federation, and identify service locations, contact and error handling information, URN's, and certificate naming for individual sites in that federation. In addition, for Shibboleth 1.2 or earlier, there may be a `ca-bundle.crt` naming signing authorities for SSL server certificates that must be referenced by a `SSLCACertificateFile` element in Apache's configuration.

Membership and participation in multiple federations can be accomplished in most cases by simply pointing to other metadata files with additional `FederationProvider` elements.

`MetadataProvider`

This configuration element supplies metadata to the SP to identify relying parties in transactions. It may appear as many times as necessary.

`<MetadataProvider type="edu.internet2.middleware.shibboleth.metadata.provider.XMLMetadataLoadWrapper" uri="`*`pathname`*`">`	This element is contained by the main `SPConfig` element and must appear once for each metadata file that the SP uses to identify and authenticate relying parties. The `URI` attribute points to a `metadata.xml` XML file, generally signed and distributed by federations, although a local copy may be maintained for bilateral relationships. This file should be regularly updated using `siterefresh`.

`siterefresh`

Shibboleth provides a simple tool called `siterefresh` in the `/opt/shibboleth/bin` folder of the distribution to maintain metadata files referenced by `shibboleth.xml`. It will return 0 only on success and a negative number on failure and log errors to `stderr`. If the data in the new metadata file is unusable or schema invalid, or the signature is invalid, the existing copy is kept and not overwritten. The SP stats all metadata files each time the data is used, allowing it to detect and utilize updates in real-time during system operation.

`siterefresh` takes the following command-line parameters:

`--url` *`URL`*	Specifies the `URL` of the remote metadata file with which to update the local file. `https://` is not supported at this time.
`--out` *`pathname`*	Specifies the local file to which to write the new metadata.
`--noverify`	Explicitly disables the requirement for the file to be signed and allows the certificate parameter to be ommitted. If the file is signed, the signature will be verified using whatever key is supplied inside it, and an invalid signature will still result in an error, but if the file is unsigned or has a valid signature, only a warning will be logged, and the result will be success.
`--cert` *`pathname`*	Specifies the location of a certificate stored in `PEM` format used to validate the signature of the metadata file. Since much of Shibboleth's security flows from metadata files, this should always be used when possible, and the certificate used should be verified independently in some out of band fashion.
`--schema` *`pathname`*	Optionally defines a base path for schemas to use when validating the file. Defaults to a location based on the installation path on Unix, or `\opt\shibboleth\etc\shibboleth` on Windows.
`--rootns` *`XML namespace`*	Optionally defines the XML namespace of the root element expected in the new file. Normally unused, provided to support alternative metadata formats that may be backported to older releases.

`--rootname` *XML element name*	Optionally defines the name of the root element expected in the new file. Normally unused, provided to support alternative metadata formats that may be backported to older releases.

If a zero is returned, the command will copy the retrieved file to the output location. Otherwise one of the following error values will be returned:

-100	an invalid combination of parameters was specified
-10	the OpenSAML library failed to initialize
-1	the file's XML digital signature was invalid
-2	a SAML exception was trapped
-3	an XML library exception was trapped
-4	a general XML security library exception was trapped
-5	an XML security library crypto exception was trapped
-6	an unknown exception was trapped

A complete command issued to `siterefresh` might take the form:

```
/opt/shibboleth/bin/siterefresh --out IQ-sites.xml --cert inqueue.pem \
--url http://wayf.internet2.edu/InQueue/IQ-sites.xml
```

It is recommended that a similar command be added to a `crontab` to keep the metadata files refreshed. Frequent updates will improve the security of an installation by providing immediate notification in the case a federation member becomes compromised.