

# MetadataConfiguration

The Shibboleth IdP generally requires SAML [metadata](#) to provision connectivity with relying parties and inform it about their capabilities and technical specifics. While you have the option to operate in a more "promiscuous" way (by enabling profiles for "unverified" RPs), this is relatively rare. In most cases, you will configure metadata sources in order to use the IdP's SAML features; this is done by adding `<MetadataProvider>` elements inside the *metadata-providers.xml* file.

## Contents

- [Schema names and locations](#)
- [MetadataProvider Types](#)
- [Attributes](#)
  - [Common Attributes](#)
  - [Reloading Attributes](#)
  - [Dynamic Attributes](#)
  - [HTTP Attributes](#)
    - [HTTP Connection Attributes](#)
    - [HTTP Security Attributes](#)
    - [HTTP Proxy Attributes](#)
    - [HTTP Caching Attributes](#)
- [Child Elements](#)
- [Miscellany](#)
  - [Multiple Configuration Files](#)
  - [Search Ordering](#)
- [V2 Compatibility](#)
  - [New Capabilities in V3](#)

A typical use case is to load (and periodically reload) entity metadata from a local file:

### Load entity metadata from a file

```
<!--
  Load (and reload) entity metadata from a local file.

  The metadata refresh process is determined by the configured values of the
  minRefreshDelay attribute (default: PT30S) and maxRefreshDelay attribute
  (default: PT4H). If you want to load (or reload) the metadata immediately,
  restart the IdP.
-->
<MetadataProvider id="LocalEntityMetadata" xsi:type="FilesystemMetadataProvider"
  metadataFile="%{idp.home}/metadata/local-metadata.xml"/>
```

Another use case is to load (and periodically reload) a metadata aggregate from a remote source via HTTP:

## Load a metadata aggregate from a remote server

```
<!--
  Load (and reload) a signed metadata aggregate from a remote HTTP server.

  This sample configuration assumes: (1) the top-level element of the XML
  document is signed; (2) the top-level element of the XML document is
  decorated with a validUntil attribute; (3) the validity interval is two
  weeks (P14D) in duration; and (4) the server supports HTTP conditional GET.

  The metadata refresh process is influenced by the configured values of
  the minRefreshDelay attribute (default: PT30S) and the maxRefreshDelay
  attribute (default: PT4H) and also by any cacheDuration and validUntil
  attributes in the metadata itself. If the server does not support HTTP
  conditional GET, the attributes should be adjusted accordingly.
-->
<MetadataProvider id="RemoteMetadataAggregate" xsi:type="FileBackedHTTPMetadataProvider"
  backingFile="%{idp.home}/metadata/federation-metadata-copy.xml"
  metadataURL="http://example.org/metadata/federation-metadata.xml">

  <!--
    Verify the signature on the root element of the metadata aggregate
    using a trusted metadata signing certificate.
  -->
  <MetadataFilter xsi:type="SignatureValidation" requireSignedRoot="true"
    certificateFile="%{idp.home}/conf/metadata/md-cert.pem"/>

  <!--
    Require a validUntil XML attribute on the root element and
    make sure its value is no more than 14 days into the future.
  -->
  <MetadataFilter xsi:type="RequiredValidUntil" maxValidityInterval="P14D"/>

  <!-- Consume all SP metadata in the aggregate -->
  <MetadataFilter xsi:type="EntityRoleWhiteList">
    <RetainedRole>md:SPSSODescriptor</RetainedRole>
  </MetadataFilter>

</MetadataProvider>
```

Increasingly, the dynamic providers (`LocalDynamicMetadataProvider` and `DynamicHTTPMetadataProvider`) are used in lieu of the reloading providers (`FilesystemMetadataProvider` and `FileBackedHTTPMetadataProvider`). See the [MetadataManagementBestPractices](#) topic for use cases and recommendations.

The [ChainingMetadataProvider](#) is often used to combine two or more metadata sources. The `metadata-providers.xml` file that ships with the software declares such a chain "wrapper" by default.

## Schema names and locations

Nearly all elements described in this page and its children are defined in the `urn:mace:shibboleth:2.0:metadata` namespace, the schema for which can be located at <http://shibboleth.net/schema/idp/shibboleth-metadata.xsd>. Throughout this document and its children, this is assumed to be the default XML namespace in effect. The namespace prefix "metadata:" is conventionally also bound to this namespace.

The namespace prefix "security:" is used to refer to the `urn:mace:shibboleth:2.0:security` namespace, the schema for which can be located at <http://shibboleth.net/schema/idp/shibboleth-security.xsd>, and is generally used only in advanced scenarios or for compatibility.

The namespace prefix "resource:" is used to refer to the `urn:mace:shibboleth:2.0:resource` namespace, the schema for which can be located at <http://shibboleth.net/schema/idp/shibboleth-resource.xsd>. It is only used by the `ResourceBackedMetadataProvider`

The namespace prefix "samlmd:" is used to refer to the `urn:oasis:names:tc:SAML:2.0:metadata` namespace, the schema for which can be located at <http://docs.oasis-open.org/security/saml/v2.0/saml-schema-metadata-2.0.xsd>

## MetadataProvider Types

The precise behavior of any `<MetadataProvider>` element is controlled by the `xsi:type` attribute (see below). The following types are supported and examples are provided for each type. If the `urn:mace:shibboleth:2.0:metadata` namespace is not the default, then a prefix (presumably "metadata:") is required when specifying these types.

<b>xsi:type</b>	<b>Function</b>
<a href="#">ChainingMetadataProvider</a>	A container for an ordered sequence of metadata providers of any type
<a href="#">DynamicHTTPMetadataProvider</a>	A dynamic provider that fetches metadata just-in-time from a suitably configured HTTP server
<a href="#">LocalDynamicMetadataProvider</a>	A dynamic provider that fetches metadata just-in-time from a local source such as a filesystem directory
<a href="#">FilesystemMetadataProvider</a>	A reloading provider that loads (and reloads) a metadata file from the filesystem in a background thread
<a href="#">FileBackedHTTPMetadataProvider</a>	A reloading provider that loads (and reloads) a metadata file from an HTTP server in a background thread
<a href="#">HTTPMetadataProvider</a>	<b>DEPRECATED:</b> Use <a href="#">FileBackedHTTPMetadataProvider</a> instead.
<a href="#">ResourceBackedMetadataProvider</a>	A reloading provider that loads (and reloads) a metadata file from a more complex source (such as SVN) in a background thread
<a href="#">InlineMetadataProvider</a>	A provider that allows metadata to be specified inline

## Attributes

Configuration attributes common to two or more metadata providers are listed in the subsections below. Other attributes are specific to the `xsi:type`, and these are documented on the pages specific to each type.

### Common Attributes

The following attributes are required on **all** metadata provider types:

Name	Type	Default	Description
id	String	required	Identifier for logging, identification for command line reload, etc.
xsi:type	String	required	Specifies the exact type of provider to use (from those listed above, or a custom extension type).

The following attributes are common to all metadata provider types except the `ChainingMetadataProvider` type:

Name	Type	Default	Description
requireValidMetadata	Boolean	true	Whether candidate metadata found by the resolver must be valid in order to be returned (where validity is implementation specific, but in SAML cases generally depends on a <code>validUntil</code> attribute.) If this flag is true, then invalid candidate metadata will not be returned.
failFastInitialization	Boolean	true	Whether to fail initialization of the underlying <code>MetadataResolverService</code> (and possibly the IdP as a whole) if the initialization of a metadata provider fails. When false, the IdP may start, and will continue to attempt to reload valid metadata if configured to do so, but operations that require valid metadata will fail until it does.
sortKey	Integer		Defines the order in which metadata providers are searched (see below), can only be specified on top level <code>&lt;MetadataProvider&gt;</code> elements.

The following are advanced settings supporting a new low-level feature allowing metadata lookup by keys other than the unique entityID and are rarely of use to a deployer.

criteriaPredicateRegistryRef <sup>3.3</sup>	Bean ID		Identifies the a custom <code>CriteriaPredicateRegistry</code> bean used in resolving predicates from non-predicate input criteria.
useDefaultPredicateRegistry <sup>3.3</sup>	Boolean	true	Flag which determines whether the default <code>CriteriaPredicateRegistry</code> will be used if a custom one is not supplied explicitly.
satisfyAnyPredicates <sup>3.3</sup>	Boolean	false	Flag which determines whether predicates used in filtering are connected by a logical 'OR' (true) or by logical 'AND' (false).

### Reloading Attributes

The following attributes are common to all reloading "batch-oriented" metadata providers (i.e., `FileBackedHTTPMetadataProvider`, `FilesystemMetadataProvider`, and `ResourceBackedMetadataProvider`):

Name	Type	Default	Description
parserPoolRef	Bean ID	shibboleth.ParserPool	Identifies a Spring bean for the XML parser used to parse metadata. Generally should not be changed.

taskTimerRef	Bean ID		Identifies a Spring bean containing a Java <code>TaskTimer</code> used to schedule reloads. When not set, an internal timer is created. Generally should not be changed.
minRefreshDelay	Duration	PT30S	Lower bound on the next refresh from the time calculated based on the metadata's expiration.
maxRefreshDelay	Duration	PT4H	Upper bound on the next refresh from the time calculated based on the metadata's expiration.
refreshDelayFactor	Real Number (strictly between 0.0 and 1.0)	0.75	A factor applied to the initially determined refresh time in order to determine the next refresh time (typically to ensure refresh takes place prior to the metadata's expiration). Attempts to refresh metadata will generally begin around the product of this number and the maximum refresh delay.
indexesRef 3.3	Bean ID		Identifies an optional <code>Set&lt;MetadataIndex&gt;</code> used to support resolution of metadata based on criteria other than an entityID.
resolveViaPredicatesOnly 3.3	Boolean	false	Flag indicating whether resolution may be performed solely by applying predicates to the entire metadata collection, when an entityID input criterion is not supplied.
expirationWarningThreshold 3.4	Duration	PT0S (disabled)	For each attempted metadata refresh (whether or not fresh metadata is obtained), if <code>requireValidMetadata</code> is true, and there is a <code>validUntil</code> XML attribute on the document root element, and the difference between <code>validUntil</code> and the current time is less than <code>expirationWarningThreshold</code> , the system logs a warning about the impending expiration.

## Dynamic Attributes

The following attributes are common to all dynamic metadata providers (i.e., `DynamicHTTPMetadataProvider` and `LocalDynamicMetadataProvider`):

Name	Type	Default	Description
parserPoolRef	Bean ID	shibboleth.ParserPool	Identifies a Spring bean for the XML parser used to parse metadata. Generally should not be changed.
taskTimerRef	Bean ID		Identifies a Spring bean containing a Java <code>TaskTimer</code> used to schedule reloads. When not set, an internal timer is created. Generally should not be changed.
refreshDelayFactor	Real Number (strictly between 0.0 and 1.0)	0.75	A factor applied to the initially determined refresh time in order to determine the next refresh time (typically to ensure refresh takes place prior to the metadata's expiration). Attempts to refresh metadata will generally begin around the product of this number and the maximum refresh delay.
minCacheDuration	Duration	PT10M (10 minutes)	The minimum duration for which metadata will be cached before it is refreshed.
maxCacheDuration	Duration	PT8H (8 hours)	The maximum duration for which metadata will be cached before it is refreshed.
maxIdleEntityData	Duration	PT8H (8 hours)	The maximum duration for which metadata will be allowed to be idle (no requests for it) before it is removed from the cache.
removeIdleEntityData	Boolean	true	Flag indicating whether idle metadata should be removed.
cleanupTaskInterval	Duration	PT30M (30 minutes)	The interval at which the internal cleanup task should run. This task performs background maintenance tasks, such as the removal of expired and idle metadata.
persistentCacheManagerRef 3.3	Bean ID		The optional manager for the persistent cache store for resolved metadata. On metadata provider initialization, data present in the persistent cache will be loaded to memory, effectively restoring the state of the provider as closely as possible to that which existed before the previous shutdown. Each individual cache entry will only be loaded if 1) the entry is still valid as determined by the internal provider logic, and 2) the entry passes the (optional) predicate supplied via <code>initializationFromCachePredicateRef</code> .
persistentCacheManagerDirectory 3.3	File specification		The directory used for an internally-constructed filesystem-based persistent cache. This is a convenience parameter to avoid specifying a full bean via <code>persistentCacheManagerRef</code> . This option will be ignored if <code>persistentCacheManagerRef</code> is specified.
persistentCacheKeyGeneratorRef 3.3	Bean ID	internal default instance	Identifies a Spring bean for a <code>Function</code> which generates the string key used with the cache manager. The default implementation produces the lower-case hex-encoded SHA-1 digest of the entityID of the <code>EntityDescriptor</code> .

initializeFromPersistentCacheInBackground <sup>3.3</sup>	Boolean	true	Flag indicating whether should initialize from the persistent cache in the background. Initializing from the cache in the background will improve IdP startup times.
backgroundInitializationFromCacheDelay <sup>3.3</sup>	Duration	PT2S (2 seconds)	The delay after which to schedule the background initialization from the persistent cache when <code>initializeFromPersistentCacheInBackground=true</code> .
initializationFromCachePredicateRef <sup>3.3</sup>	Bean ID	an "always true" predicate	Identifies a Spring bean for an optional <code>Predicate</code> which determines whether a given entity should be loaded from the persistent cache at resolver initialization time.

## HTTP Attributes

The following attributes are common to all HTTP metadata providers (i.e., `DynamicHTTPMetadataProvider` and `FileBackedHTTPMetadataProvider`).

An HTTP metadata provider includes a default implementation of the `org.apache.http.client.HttpClient` interface. The attributes in the following subsections control the behavior of the default HTTP client. To override the default client implementation, configure the following attribute:

Name	Type	Default	Description
httpClientRef	Bean ID		A reference to an externally defined Spring bean that specifies an <code>org.apache.http.client.HttpClient</code> object. This attribute conflicts with and overrides all of the HTTP attributes. See the <a href="#">HttpClientConfiguration</a> topic for more information.

Use of the `httpClientRef` attribute precludes the use of any and all of the HTTP attributes in the following subsections.

## HTTP Connection Attributes

The following attributes apply to the HTTP connections obtained and managed by an HTTP metadata provider:

Name	Type	Default	Description
connectionRequestTimeout <sup>3.3</sup>	Duration	Depends on the provider type	The maximum amount of time to wait for a connection to be returned from the HTTP client's connection pool manager. Set to <code>PT0S</code> to disable. This attribute is incompatible with <code>httpClientRef</code> .
connectionTimeout <sup>3.3</sup>	Duration	Depends on the provider type	The maximum amount of time to wait to establish a connection with the remote server. Set to <code>PT0S</code> to disable. This attribute is incompatible with <code>httpClientRef</code> .
requestTimeout	Duration	Depends on the provider type	<b>DEPRECATED:</b> Use <code>connectionTimeout</code> instead.
socketTimeout <sup>3.3</sup>	Duration	Depends on the provider type	The maximum amount of time to wait between two consecutive packets while reading from the socket connected to the remote server. Set to <code>PT0S</code> to disable. This attribute is incompatible with <code>httpClientRef</code> .

## HTTP Security Attributes

The following security-related attributes apply to any HTTP metadata provider:

Name	Type	Default	Description
disregardTLSCertificate	Boolean	false	If true, no TLS certificate checking will take place over an HTTPS connection. This attribute is incompatible with <code>httpClientRef</code> . (Be careful with this setting, it is typically only used during testing. See the <a href="#">HttpClientConfiguration</a> topic for more information.)
disregardSSLCertificate	Boolean	false	<b>DEPRECATED:</b> Use <code>disregardTLSCertificate</code> instead.
basicAuthUser	String		<b>DEPRECATED:</b> Use <code>httpClientSecurityParametersRef</code> instead.
basicAuthPassword	String		<b>DEPRECATED:</b> Use <code>httpClientSecurityParametersRef</code> instead.
tlsTrustEngineRef <sup>3.1</sup>	Bean ID		<b>DEPRECATED:</b> Use <code>httpClientSecurityParametersRef</code> instead.

httpClientSecurityParametersRef.3.3	Bean ID		A reference to an externally defined Spring bean that specifies an <code>org.opensaml.security.httpClient.HttpClientSecurityParameters</code> instance, which consolidates all HTTP security parameters including advanced TLS usage. This attribute conflicts with and overrides any explicit <code>TrustEngine</code> implementation configured as an inline <code>&lt;TLSTrustEngine&gt;</code> element. See the <a href="#">HttpClientConfiguration</a> topic for more information.
-------------------------------------	---------	--	---

## HTTP Proxy Attributes

The following attributes configure an HTTP proxy for use with an HTTP metadata provider:

Name	Type	Default	Description
proxyHost	String		The hostname of the HTTP proxy through which connections will be made. This attribute is incompatible with <code>httpClientRef</code> .
proxyPort	String		The port of the HTTP proxy through which connections will be made. This attribute is incompatible with <code>httpClientRef</code> .
proxyUser	String		The username used with the HTTP proxy through which connections will be made. This attribute is incompatible with <code>httpClientRef</code> .
proxyPassword	String		The password used with the HTTP proxy through which connections will be made. This attribute is incompatible with <code>httpClientRef</code> .

## HTTP Caching Attributes

The following attributes configure an HTTP cache on an HTTP metadata provider:

Name	Type	Default	Description
httpCaching	"none", "file", or "memory"	Depends on the provider type	<p>The type of low-level HTTP caching to perform. There are three choices:</p> <ul style="list-style-type: none"> <li>"none" indicates the HTTP response is not cached by the client library</li> <li>"file" indicates the HTTP response is written to disk (but will not survive a restart)</li> <li>"memory" indicates the HTTP response is stored in memory</li> </ul> <p>This attribute is incompatible with <code>httpClientRef</code> and its value may not be specified as a bean property.</p> <p>Some metadata providers, most notably the reloading "batch-oriented" providers, implement HTTP caching at a higher layer and tend to work best with <code>httpCaching="none"</code>.</p>
httpCacheDirectory	String		If <code>httpCaching="file"</code> , this attribute specifies where retrieved files are to be cached. This attribute is incompatible with <code>httpClientRef</code> .
httpMaxCacheEntries	Integer	"memory": 50 "file": 100	The maximum number of responses written to cache. This attribute is incompatible with <code>httpClientRef</code> .
httpMaxCacheEntrySize	Integer	"memory": 1048576 (1MB) "file": 10485760 (10MB)	The maximum response body size that may be cached, in bytes. This attribute is incompatible with <code>httpClientRef</code> .

## Child Elements

The following child elements may be used with all metadata provider types (except the `ChainingMetadataProvider` type):

Name	Cardinality	Description
<code>&lt;MetadataFilter&gt;</code>	0 or more	A metadata filter applied to candidate metadata as it flows through the metadata pipeline
<code>&lt;security:TrustEngine&gt;</code>	0 or more	<b>DEPRECATED:</b> See the note at the bottom of this page

Other allowable child elements are specific to the `xsi:type` of the provider used, and these are documented on the pages specific to each type.

## Miscellany

### Multiple Configuration Files

As described in the [ReloadableServices](#) documentation, the configuration is actually loaded from a bean whose name is specified by the property `idp.service.metadata.resources`, with the default value `shibboleth.MetadataResolverResources` which in turn is defined in `services.xml` to be a list with one entry: the file `metadata-providers.xml`

You can, if you choose, override this with additional or different files or more advanced sources. Each resource must supply a "top level" `<MetadataProvider>` element with attributes and child elements as described above. Search order amongst multiple top level elements is arbitrated by the `sortKey` attribute, where lower values are processed before higher ones.

## Search Ordering

If a specific relying party (as identified by a specific entityID) is duplicated in the metadata sources provided, then which precise entry is chosen is governed by the following rules:

- Metadata sources combined via a chain are searched in the order in which they occur in the chain, and the first entry matching the entityID is returned.
- If multiple "top level" Metadata Providers are provided then they are searched in an order derived from the (numeric) value of the `sortKey` attribute (lowest key first). If no `sortKey` is specified, then the search order is undefined.
- In whatever order of sources is in effect, the first entry matching the entityID is returned.
- If a single metadata source contains multiple entries with the same entityID, then which entry is returned is undefined (exception: invalid entries would be ignored in favor of valid ones in most cases).

## V2 Compatibility

A single `<MetadataProvider>` element may be embedded in a legacy *relying-party.xml* file as described in [the older documentation](#). Consult the V2 documentation for this, and do not mix and match this approach with newer configuration features.

During the V2 to V3 upgrade process, the original V2 *relying-party.xml* file is copied to *metadata-providers.xml*, to serve as the metadata configuration for the new version. It's strongly advisable after upgrading to update that file by stripping it of the older content and promote the `<MetadataProvider>` element in it to the root of the file. In the interim all other content in the file except for `<MetadataProvider>` elements (and any referenced `<security:TrustEngine>` elements) is ignored.

The following non-relevant trust engine types often found in a legacy *relying-party.xml* file are ignored if seen, and are not used for metadata verification (despite the confusing names):

- Chaining
- MetadataExplicitKey
- MetadataPKIXX509Credential
- MetadataExplicitKeySignature
- MetadataPKIXSignature
- StaticPKIXX509Credential

## New Capabilities in V3

The V3 metadata configuration syntax is backward-compatible with the V2 `<MetadataProvider>` syntax and adds some useful new shortcuts as well.



### Avoid deprecated features

In anticipation of V4, a number of IdP features have been deprecated in V3. To ensure a seamless upgrade to V4, avoid the use of deprecated features in your V3 deployment. In particular, avoid any [metadata-related features deprecated in V3](#).

You can now provide multiple metadata configuration files (not just multiple metadata sources in one file), as described above.

A `SignatureValidation` filter need not contain a `trustEngineRef` attribute referencing a separately-defined trust engine; instead a certificate file may be specified directly with the `certificateFile` attribute. Alternatively, a PEM-format public key may be supplied inline via the `<PublicKey>` element.



### Using a TrustEngine element

As a child element of the `<MetadataProvider>` element, the use of the `<security:TrustEngine>` element is **DEPRECATED**. If used at all, the element should be declared inside a `SignatureValidation` filter or in most cases simply replaced with the `certificateFile` attribute. See the [SignatureValidationFilter](#) topic for more information.