NativeSPMetadataProvider

The `<MetadataProvider>` element configures a source of Metadata for the SP to use. Generally used only within the `shibd` service.

Unlike other configuration files which describe how the SP will behave, the metadata loaded by the SP describes the IdPs it wants to interact with. Each application determines the set of partner sites to trust by loading their metadata (or providing some kind of dynamic mechanism to do so).

For help with understanding and/or creating IdP metadata, see the Metadata and MetadataForIdP topics.

Common Attributes

- `type` (string)
  - Name of plugin type.

XML MetadataProvider

Identified by `type="XML"`, supplies metadata from local or remote XML files in the standard SAML 2.0 format.

The XML "portion" is a reloadable resource, which means that the XML content can be supplied inline, in a local file, or a remote file, and can be monitored for changes and reloaded on the fly. The root of the XML instance **MUST** be a `<md:EntityDescriptor>` or `<md:EntitiesDescriptor>` element.

```
<MetadataProvider type="XML" url="https://federation.org/metadata.xml" backingFilePath="fedmetadata.xml">
    <MetadataFilter type="Signature" certificate="fedsigner.pem"/>
</MetadataProvider>
```

Version 2.1 and Above: In the case of remotely acquired metadata, an instance that contains a `cacheDuration` attribute will affect the reload interval of the resource by potentially causing more frequent reloading.

Attributes

Inherits attributes supported by reloadable resources.

Version 2.4 and Above:

- `minRefreshDelay` (time in seconds) (defaults to 600)
  - Determines the minimum refresh interval when polling a remote resource for changes.
- `maxRefreshDelay` (time in seconds)
  - This is a synonym for the `reloadInterval` setting, and determines the maximum allowed refresh interval when polling a remote resource for changes.
- `refreshDelayFactor` (decimal between 0 and 1, non-inclusive) (defaults to 0.75)
  - Factor applied to the metadata's own validity or caching period to determine the reload interval to use. Once applied, the result is bounded by the `minRefreshDelay` and `maxRefreshDelay` settings to determine the time of the next attempt. If reload attempts fail, the interval will increase in linear fashion to limit attempts.
- `discoveryFeed` (boolean) (defaults to true)
  - When true, a JSON feed of IdP information will be produced and cached in memory for use by the new DiscoveryFeed handler. Can be disabled to save processing and memory.
- `legacyOrgNames` (boolean, deprecated) (defaults to false)
  - When true, the JSON feed of IdP information (if produced) will use the same nonstandard sources for organization names as the Shibboleth centralized DS. *This attribute is provided for transition purposes only, is deprecated and will be removed from future versions*.

Version 2.5 and Above

- `dropDOM` (boolean) (defaults to true)

        - When true, the underlying XML DOM structure is dropped after processing a new metadata instance. Set to false to maintain the DOM in memory if there are supplemental options being used that operate more efficiently with the DOM maintained.
    - `tagsInFeed` (boolean) (defaults to false)
        - When true, adds EntityAttribute extension attribute values (simple-valued only) to the discovery feed.

Child Elements

- `<MetadataFilter>` (zero or more)
    - A filter to run against any metadata supplied by the plugin.

Version 2.1 and Above:

- `<TransportOption>` (zero or more)
    - The transport implementation is supplied by the same underlying code used for SOAP client communication, and the same configuration properties that affect that process are used, such as HTTP forward proxies, timeouts, client authentication via certificates or HTTP, etc.

Version 2.5 and Above

- `<DiscoveryFilter>` (zero or more)
    - Specifies a whitelist or blacklist to apply to the entities in the metadata for purposes of the JSON discovery feed optionally produced by the plugin. Any entities not in the feed are still included in all other uses of the metadata.
        - `type` ("Whitelist" or "Blacklist")
            - Required attribute, specifies what type of filtering to apply.
        - `matcher` (string)
            - Required attribute specifying a type of EntityMatcher to use in evaluating the entities in the feed. Other content will be included as specified by the type of plugin.

Example

Example `<MetadataProvider>` with `<DiscoveryFilter>` that applies a blacklist to entities in the metadata for purposes of the JSON discovery feed optionally produced by the plugin. This blacklist is based on the presence of the http://refeds.org/category/hide-from-discovery entity attribute.

```
<MetadataProvider type="XML" uri="http://federation.org/federation-metadata.xml"
      backingFilePath="federation-metadata.xml" reloadInterval="7200">
    <MetadataFilter type="RequireValidUntil" maxValidityInterval="2419200"/>
    <MetadataFilter type="Signature" certificate="fedsigner.pem"/>
    <DiscoveryFilter type="Blacklist" matcher="EntityAttributes" trimTags="true"
      attributeName="http://macedir.org/entity-category"
      attributeNameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
      attributeValue="http://refeds.org/category/hide-from-discovery" />
</MetadataProvider>
```

Dynamic MetadataProvider

Indicated by `type="Dynamic"`, allows for resolution of metadata based on the "well-known" location mechanism defined in the SAML 2.0 metadata specification .

If an `entityID` is a URL, this plugin will attempt to lookup its metadata by resolving the URL into an XML instance rooted by a `md:EntityDescriptor`. The result will be cached for the length of time indicated by the metadata's `cacheDuration` or `validUntil` attributes (or until process restart or configuration reload).

Version 2.0:

Absolutely **no** trust processing is performed based on the location, use of TLS/SSL, or any other transport layer technology. The metadata is loaded as-is, and will be implicitly trusted. As a result, it is **STRONGLY RECOMMENDED** that this plugin be used only in conjunction with a Signature metadata filter to authenticate the source of the metadata.

Absent such a filtering step, the SP will essentially be insecure and open to straightforward attack.

Version 2.1 and Above:

As of version 2.1, this plugin includes support for authentication of the transport layer used to acquire the metadata. This allows for experimentation with the exchange of unsigned metadata using TLS-protected entityIDs, but the use of a Signature metadata filter is still permitted, in combination with or instead of the transport check.

Also as of version 2.1, the transport implementation is supplied by the same underlying code used for SOAP client communication, and the same configuration properties that affect that process are used, such as timeouts, client authentication via certificates or HTTP, etc.

Version 2.6 and Above:

As of version 2.6, this plugin can handle resolving metadata via files. If the resulting value to resolve starts with "file://", then a local file will be resolved to satisfy the request. Typically the `<Subst>` element can be used together with the new `hashed` attribute to map entityIDs to hashed filenames.

Attributes

- `validate` (boolean) (defaults to "false")
  - If "true", metadata will be schema validated when parsed.

Version 2.1 and Above:

- `maxCacheDuration` (time in seconds) (defaults to 28800, 8 hours)
  - Upper bound on time before metadata for entity will be re-acquired.
- `verifyHost` (boolean) (defaults to true)
  - If true, attempts to resolve metadata using a TLS-enabled URL will verify the hostname in the server's certificate against the expected hostname.
- `ignoreTransport` (boolean) (defaults to false)
  - If true, authentication of the transport layer will be ignored when resolving metadata. This must be set to true to allow non-https entityID values. If false, a `<TrustEngine>` child element must be specified.

Version 2.4 and Above:

- `minCacheDuration` (time in seconds) (defaults to 600)
  - Lower bound on time before metadata for entity will be re-acquired.
- `refreshDelayFactor` (decimal between 0 and 1, non-inclusive) (defaults to 0.75)
  - Factor applied to the metadata's own validity or caching period to determine the reload interval to use. Once applied, the result is bounded by the `minCacheDuration` and `maxCacheDuration` settings to determine the time of the next attempt. If reload attempts fail, the existing metadata (if any) will be reused until it actually expires.

Version 2.6 and Above:

- `cleanupInterval` (time in seconds) (defaults to 1800)
  - Time between execution of background thread to scan for expired cached metadata and remove it from memory. Set to 0 to disable any cleanup.
- `cleanupTimeout` (time in seconds) (defaults to 1800)
  - Extra time to leave "stale" entries in the cache before the background cleanup process will remove them.

Child Elements

- `<MetadataFilter>` (zero or more)
  - A filter to run against any metadata supplied by the plugin.

Version 2.1 and Above:

- `<TrustEngine>` (optional)
  - A trust engine to apply to server certificates when resolving metadata using a TLS-enabled URL. This trust engine obviously must not require the use of metadata to operate. If not supplied, the `ignoreTransport` attribute must be set to true to avoid a configuration error. This is done to prevent a misconfigured trust engine from resulting in insecure metadata resolution.

Version 2.4 and Above

- `<Subst>` (optional)
  - Simple transform whose element content consists of a string containing the substring "$entityID", into which the entityID value is substituted. If the element contains a `hashed` attribute (supported as of V2.6+), the value must be a digest algorithm (e.g. SHA1) to apply to the entityID. If the element contains an `encoded` attribute set to "false", the value will be replaced directly, otherwise it will be URL-encoded first.
- `<Regex>` (optional)
  - Complex transform containing a `match` attribute containing a regular expression against which the entityID value is applied, and whose element content contains a replacement expression to run based on the results of the match. Only numeric/positional group references (e.g. $1) are supported.

Chaining MetadataProvider

Indicated by `type="Chaining"`, allows multiple sources of metadata to be supplied in sequence.

With V2.4 and above, this is implied by any configuration with multiple `<MetadataProvider>` elements, so is no longer explicitly needed unless one of its optional settings is required.

While there is some limited capability for controlling the handling of duplicate entities, it is explicitly **NOT** supported for a single `entityID` to appear more than once with the same valid role, and the software will **NOT** behave predictably in such a case. In other words, if the same entity supports a given role, its metadata **MUST** be identical in all chained sources.

```
<MetadataProvider type="Chaining">
    <MetadataProvider type="XML" path="partners.xml"/>
    <MetadataProvider type="XML" url="https://federation.org/metadata.xml" backingFilePath="fedmetadata.xml"/>
</MetadataProvider>
```

Attributes

- `precedence` ("first" or "last")
    - Controls the search process. If "last", then a search will examine every source of metadata and the last match found will be used. Otherwise a search will terminate at the first match found.

Child Elements

- `<MetadataProvider>` (one or more)
    - The metadata sources to chain together.

Folder MetadataProvider (Version 2.5 and Above)

Indicated by `type="Folder"`, loads a single directory of local file-based metadata sources as though each individual file were defined via the `type="XML"` provider inside a chain. Each ordinary file found in the specified directory is loaded, but nested directories are ignored. The directory is not monitored for changes, so additional files added after initial configuration will not be seen, but changes to the existing files found can be detected in the usual fashion.

```
<MetadataProvider type="Folder" path="metadata"/>
```

Attributes

- `path` (absolute or relative pathname)
    - Directory to use for metadata files.
- `precedence` ("first" or "last")
    - Controls the search process. If "last", then a search will examine every source of metadata and the last match found will be used. Otherwise a search will terminate at the first match found. However, unlike the Chaining provider, one cannot control the actual order of files loaded. The log can be used to identify the order used, so this option is primarily to aid in debugging problems with duplication.
- `reloadChanges` (boolean) (defaults to "true")
    - If "true", monitors individual files for changes.
- `validate` (boolean) (defaults to "false")
    - If "true", metadata will be schema validated when parsed.
- `discoveryFeed` (boolean) (defaults to true)
    - When true, a JSON feed of IdP information will be produced and cached in memory for use by the new DiscoveryFeed handler. Can be disabled to save processing and memory.
- `legacyOrgNames` (boolean, deprecated) (defaults to false)
    - When true, the JSON feed of IdP information (if produced) will use the same nonstandard sources for organization names as the Shibboleth centralized DS. *This attribute is provided for transition purposes only, is deprecated and will be removed from future versions*.
- `dropDOM` (boolean) (defaults to true)
    - When true, the underlying XML DOM structure is dropped after processing a new metadata instance. Set to false to maintain the DOM in memory if there are supplemental options being used that operate more efficiently with the DOM maintained.

Child Elements

Any child elements supported by the `type="XML"` plugin documented above can be included, and will be applied uniformly to all embedded sources.