DSAddMetadata

Declaring the metadata sources

Every potential source of metadata is declared to the Discovery Service by a `<MetadataProvider>` element in the `$DS_HOME/wayfconfig.xml` file.

If the wayfconfig.xml file is changed, you need to restart the service

The `<MetadataProvider>` element must have the following attributes

- **identifier** - This is the name by which the metadata identified
- **displayName** - This is the name displayed for the metadata group.
- **url** - This is the location of the metadata. Three URL forms are supported `file:<LocalPath>`, `http://<RemotePath>` and `https://<RemotePath>`

In addition, if the `http://` or form `https://<RemotePath>` is used, the following attributes may be specified:

- **backingFile** - This is where the remote file will be stored. This is required for `http://` and `https://` urls.
- **timeout** - This is the timeout for the http connection, this is deprecated starting in V1.2 and replaced by requestTimeout.
- **disregardSslCertificate** (added in v1.2) - boolean flag indicating whether the servers SSL certificate should always be accepted regardless of whether its valid (defaults value: false)
- **requestTimeout** - (added in v1.2) Maximum length of time to wait for the remote server to finish its response given in XML duration notation (default value: PT5S).
- **proxyHost** (added in v1.2) - hostname of the HTTP proxy to use when fetching metadata
- **proxyPort** (added in v1.2) - port of the HTTP proxy to use when fetching metadata
- **proxyUser** (added in v1.2) - username used when connecting to the HTTP proxy to use when fetching metadata
- **proxyPassword** (added in v1.2) - password used when connecting to the HTTP proxy to use when fetching metadata
- **basicAuthUser** (added in v1.2) - HTTP BASIC authentication username used when connecting to the HTTP proxy to use when fetching metadata
- **basicAuthPassword** (added in v1.2) - HTTP BASIC authentication password used when connecting to the HTTP proxy to use when fetching metadata

In all cases, for releases starting V1.2.0, the following attributes to perform extra data validation, or the reload frequency,

- **certicateFile** - If specified, this is the *path* to a certificate file. This certificate is used to validate the signature on the root element of the incoming metadata. The filter will prevent loading of the metadata if it fails validation or if there is no certificate present.
- **maxValidityInterval** - If specified, this value is used to ensure that the metadata contains a validUntil attribute on the root of the metadata. This ensures that old metadata, which may contain entities which have been removed/revoked, is not used. If the value is "0" then it specifies the interval, in seconds, from now within which the validUntil date must fall. A value of zero indicates no upper limit.
- **refreshDelayFactor** (added in v1.2) - an number between 0.0 and 1.0, exclusive, used to determine the next metadata refresh cycle based on the current metadata's cache expiration time (default value: 0.75), see the IdP Documentation for more details.
- **minRefreshDelay** (added in v1.2) - the minimum interval between successive metadata refresh operations given in XML duration notation (default value: PT5M), see the IdP Documentation for more details.
- **maxRefreshDelay** (added in v1.2) - the maximum interval between successive metadata refresh cycles given in XML duration notation (default value: PT4H), see the IdP Documentation for more details.

Setting the min and max refresh delay to the same value is a nonsensical configuration. Don't do it.

Example Metadata Declarations

```
<MetadataProvider
     displayName="Local Federation"
     identifier="FileFed"
     url="file:///etc/DiscoveryService/metadata/sites.xml"/>

<MetadataProvider
     displayName="UK Federation"
     identifier="UkFed"
     certicateFile="/etc/metadata/ukfederation.pem"
     maxValidityInterval = "P7D"
     backingFile="/etc/metadata/ukfed_store.xml"
     url="http://metadata.ukfederation.org.uk/ukfederation-metadata.xml"/>
```


In all cases, the Discovery Service will reload metadata as soon as it has been changed. It is **not** necessary to restart the service

If the metadata contains <DiscoveryResponse> elements, then the binding attribute is checked. If an entity has an invalid binding then it is removed from the metadata and a note written to the log. If required the behavior can be limited to issuing a warning bu setting the element "warnOnBadBinding" in the <Default> configuration to be "true".

Using the metadata source.

Once a metadata source has been declared, it is associated with a specific location via the <DiscoveryServiceHandler> element.

Example Discovery Service declarations

```
<DiscoveryServiceHandler [...]>
   <Federation identifier="UkFed"/>
   [...]
</DiscoveryServiceHandler>
```

The DiscoveryServiceHandler is discussed in more details in DSUserInterface

Filtering Metadata

A <MetadataProvider> may have one or more custom filters added (written in Java). Each filter has to implement org.opensaml.saml2.metadata.provider.MetadataFilter and have a constructor which take a single parameter of type org.w3c.dom.Element (this being the element which defines the filter as described below).

A filter is associated with a Metadata Provider via a <Filter> element. This is unstructured. It may have any attributes and sub elements which can be used to provide parameters to the code. It **must** have the following attributes:

- **identifier** - An unique identifier for the filter
- **type** - The class name for the filter.

Example Metadata Filters declaration

```
<MetadataProvider [...]>
   <Filter identifier="Filter1"
      type="uk.ac.ed.sdss.FilterForStuff">
      <MoreSpecificStuff
          param="wibble"
      />
   </Filter>
   <Filter identifier="Filter2"
      type="edu.internet2.OtherFilter">
      <Stuff>
         <EvenMoreStuff/>
      </Stuff>
   </Filter>
</MetadataProvider>
```

White and BlackList

The DiscoveryService is shipped with a simple white-list and black-list filter. Given a list of entities, the metadata will be adjusted to remove all elements which are **not** in the list (white list operation) or to remove all entities which **are** on the list (blacklist operation).

The filter is configured thus:

Example Black List filter

```
<Filter identifier="Black"
        type="edu.internet2.middleware.shibboleth.wayf.plugins.provider.ListFilter"
        excludeEntries="true">
  <EntityId>https://first.blacklisted.entity.edu/IdP</EntityId>
  <EntityId>https://another.blacklisted.entity.edu/IdP</EntityId>
</Filter>
```

The `excludeEntries` controls whether elements on the list are excluded from the metadata (blacklist operation) or have to be included (white list operation).

The resulting metadata must include all SPs which interact with the DS. This is particularly important to remember when buidling white list (excludeEntries="false") filters