IdPConfigResource

Define a New Input Resource

Input resources are components that define how to retrieve information necessary within other components. Some concrete examples are using the URL input resource to fetch the metadata loaded by a metadata provider or fetch CRLs used by X.509 credentials.

Input Resource Types

Input resources are always declared as the type of some element within the configuration of the component using the resource. In this example we will use the element `<EXAMPLE>` as an example element that might be defined as a resource. Configuration elements which may be resources will be noted as such in the documentation for those components.

Classpath Resource

An input resource that reads information from a file located on the Java classpath.

The resource is declared using type attribute `xsi:type="resource:ClasspathResource"` with the following additional attribute:

- **file** - the classpath path to the file to be used as the resource. This should always start with a '/' which represents the root of the classpath.

Example Classpath Resource

```
<EXAMPLE xsi:type="resource:ClasspathResource"
         file="/org/example/data/key.pem" />
```

Filesystem Resource

An input resource that reads information from a file located on the filesystem.

The resource is declared using type attribute `xsi:type="resource:FilesystemResource"` with the following additional attribute:

- **file** - the filesystem path to the file to be used as the resource

Example Filesystem Resource

```
<EXAMPLE xsi:type="resource:FilesystemResource"
         file="/path/to/some/key.pem" />
```

HTTP Resource

An input resource that reads information from an HTTP(S) URL.

The resource is declared using type attribute `xsi:type="resource:HttpResource"` with the following additional attribute:

- **url** - the URL to the file to be used as the resource

Example HTTP Resource

```
<EXAMPLE xsi:type="resource:HttpResource"
         url="http://www.example.org/someFile.xml" />
```

File-backed HTTP Resource

An input resource that reads information from an HTTP(S) URL and also stores a backup on the filesystem. The backup copy is used if the remote copy cannot be reached (assuming an initial copy was successfully fetched).

The resource is declared using type attribute `xsi:type="resource:FileBackedHttpResource"` with the following additional attribute:

- **url** - the URL to the file to be used as the resource

- **file** - the filesystem path to which the backup file will be written

Example File-backed HTTP Resource

```
<EXAMPLE xsi:type="resource:FileBackedHttpResource"
         url="http://www.example.org/someFile.xml"
         file="/path/to/backup/someFile.xml" />
```

Subversion Resource

An input resource that reads information from a Subversion repository.

> For licensing reasons, this resoure type is not usable out of the box as of V2.4.1. To enable SVN support, you will need to download SVNKit yourself, add its jars to the install tree's **lib** folder, and reinstall. Make sure to use a version compatible with your working copy format, or remove the working copy and start fresh. Note that only 1.3 has been tested.

The resource is declared using type attribute `xsi:type="resource:SVNResource"` with the following required attributes:

- **repositoryURL** - Subversion URI to the remote directory (svn, svn+ssh, http(s), file supported)
- **workingCopyDirectory** - location where the local working copy of the resource will be kept
- **resourceFile** - the name of the file, relative to the `workingCopyDirectory`, that should be loaded in to the IdP

The following optional attributes may also be set:

- **connectionTimeout** - length of time in milliseconds to wait for the SVN connection to be established - *default: 3000 (3 seconds)*
- **readTimeout** - length of time in milliseconds to wait for the SVN checkout to complete - *default: 5000 (5 seconds)*
- **revision** - The revision number, a positive integer, that should be retrieved. Revision dates and keywords (e.g. HEAD, PREV) are not supported. Omission of this attribute will cause the HEAD revision to be retrieved.
- **username** - user name used to authenticate to the remote repository
- **password** - password used to authentication to the remote repository
- **proxyHost** - HTTP proxy used through which HTTP/HTTPS connections will be sent
- **proxyPort** - HTTP proxy port through which HTTP/HTTPS connections will be sent
- **proxyUsername** - username for the HTTP proxy server
- **proxyPassword** - password for the HTTP proxy server

Example Subversion Resource

```
<EXAMPLE xsi:type="resource:SVNResource"
         repositoryURL="http://svn.example.org/myProject/some/path"
         workingCopyDirectory="/path/to/working/copy"
         resourceFile="project.xml"
         revision="513" />
```

Resource Filter

After being fetched but prior to being made available to the IdP a resource may be run through a filter. These filters may perform any processing on the resource that they wish.

Resource filters are defined by the `<ResourceFilter>` element. If defined, this element must be the first child element within a input resource's configuration. A resource may only have one filter.

Chaining Resource Filter

A filter that allows a list of child filters to be executed in the order they are defined.

This filter is defined using `<ResourceFilter xsi:type="Chaining" xmlns="urn:mace:shibboleth:2.0:resource">` and may contain any number of child `<ResourceFilter` elements.

Example Chaining Resource Filter

```
<ResourceFilter xsi:type="Chaining" xmlns="urn:mace:shibboleth:2.0:resource">
    <ResourceFilter xsi:type="PropertyReplacement" xmlns="urn:mace:shibboleth:2.0:resource"
                    propertyFile="/path/to/file1.properties"/>
    <ResourceFilter xsi:type="PropertyReplacement" xmlns="urn:mace:shibboleth:2.0:resource"
                    propertyFile="/path/to/file2.properties"/>
</ResourceFilter>
```

Property Replacement Filter

A filter that replaces macros in a text input resources with values from a property file. Macros must be on the form `${NAME}`.

This filter is defined using `<ResourceFilter xsi:type="PropertyReplacement" xmlns="urn:mace:shibboleth:2.0:resource">` and must have the following configuration attribute:

- **propertyFile** - Path to the property file whose values will be used to replace the macros in the loaded input resource

Example Property Replaced Resource Filter

```
<ResourceFilter xsi:type="PropertyReplacement" xmlns="urn:mace:shibboleth:2.0:resource"
                propertyFile="/path/to/file1.properties"/>
```