SignatureValidationFilter

The `SignatureValidation` filter verifies that a metadata instance is signed correctly with a trusted key, and is the linchpin of the security of most Shibboleth deployments.

> **The "Sign and Expire" distribution model**
>
> In practice, a `SignatureValidation` filter and a `RequiredValidUntil` filter are often used together to securely obtain remote metadata via HTTP. See the `FileBackedHTTPMetadataProvider` and `DynamicHTTPMetadataProvider` topics for explicit configuration examples. Other distribution models are discussed in the TrustManagement topic.

There are four approaches to supplying the trust policy to the `SignatureValidation` filter:

- A pointer to a certificate file
- A reference to an externally defined TrustEngine bean
- An inline `<PublicKey>` element
- An inline `<security:TrustEngine>` element

> **Filter order is important!**
>
> In the overall sequence of filters, a filter of type `SignatureValidation` **must** appear before any filter that alters the metadata instance. Examples of the latter include EntityAttributesFilter, EntityRoleWhiteListFilter, NameIDFormatFilter, and PredicateMetadataFilter.

Schema

The `<MetadataFilter>` element and the type `SignatureValidation` are defined by the `urn:mace:shibboleth:2.0:metadata` schema, which can be located at http://shibboleth.net/schema/idp/shibboleth-metadata.xsd.

The `<security:TrustEngine>` element is defined in the `urn:mace:shibboleth:2.0:security` namespace, the schema for which is located at http://shibboleth.net/schema/idp/shibboleth-security.xsd

Attributes

Name	Type	Default	Description
`requireSignedRoot` 3.2	Boolean	true	If true, this fails to load metadata with no signature on the root XML element.
`requireSignedMetadata`	Boolean	true	(DEPRECATED) Old version of `requireSignedRoot`
`certificateFile`	File		Path to a certificate file whose key is used to verify the signature. Conflicts with `trustEngineRef` and both of the child elements.
`defaultCriteriaRef`	Bean Reference	shibboleth.MetadataSignatureValidationStaticCriteria	The ID of an externally defined CriteriaSet used as input the to the trust engine, not generally used.
`signaturePrevalidatorRef`	Bean Reference	SAMLSignatureProfileValidator	The ID of an externally defined SignaturePrevalidator. Used to perform pre-validation of an XML Signature, for example to validate that the signature conforms to a particular profile of XML Signature.
`dynamicTrustedNamesStrategyRef`	Bean Reference	BasicDynamicTrustedNamesStrategy	The ID of an externally defined Function<XMLObject, Set<String>>. This will be used to extract dynamic trusted names from signed metadata elements.
`trustEngineRef`	Bean Reference		The ID of a `<security:TrustEngine>` defined somewhere else in the configuration. Conflicts with `certificateFile` and both of the child elements.

Child Elements

One of the following two child elements may be configured. Their use conflicts with the `certificateFile` and `trustEngineRef` XML attributes.

Name	Description

`<PublicKey>`	A PEM-format public key. You can obtain a public key from a certificate using a command such as: `$ openssl x509 -pubkey -in cert.pem -noout`
`<security:TrustEngine>`	A trust engine plugin that defines how the signature is to be checked

Examples

Externally specified certificate file

```
<MetadataFilter xsi:type="SignatureValidation" requireSignedRoot="true" certificateFile="${idp.home}/credentials
/signer.pem"/>
```

Inline key

```
<MetadataFilter xsi:type="SignatureValidation" requireSignedRoot="true">
  <PublicKey>
    MIIBI.....
  </PublicKey>
</MetadataFilter>
```

Metadata Provider with inline trust engine

```
<MetadataFilter xsi:type="SignatureValidation" requireSignedRoot="true">
    <security:TrustEngine id="SignerTrustEngine" xsi:type="security:StaticExplicitKeySignature">
        <security:Credential id="SignerCredential" xsi:type="security:X509ResourceBacked">
            <security:Certificate>${idp.home}/credentials/signer.pem</security:Certificate>
        </security:Credential>
    </security:TrustEngine>
</MetadataFilter>
```

Metadata Provider with inline trust engine with multiple validation credentials

```
<MetadataFilter xsi:type="SignatureValidation" requireSignedRoot="true">
    <security:TrustEngine id="SignerTrustEngine" xsi:type="security:StaticExplicitKeySignature">
        <security:Credential id="SignerCredential_1" xsi:type="security:X509ResourceBacked">
            <security:Certificate>${idp.home}/credentials/signer1.pem</security:Certificate>
        </security:Credential>
        <security:Credential id="SignerCredential_2" xsi:type="security:X509ResourceBacked">
            <security:Certificate>${idp.home}/credentials/signer2.pem</security:Certificate>
        </security:Credential>
    </security:TrustEngine>
</MetadataFilter>
```

PKIX signature validation with static trust anchors

```
<MetadataFilter xsi:type="SignatureValidation" requireSignedRoot="true">
    <security:TrustEngine id="VTSignerTrustEngine" xsi:type="security:StaticPKIXSignature">
        <security:TrustedName>shib</security:TrustedName>
        <security:ValidationInfo id="VTPKIXValidationInfo" xsi:type="security:PKIXResourceBacked">
            <security:Certificate>${idp.home}/credentials/vtmwca.pem</security:Certificate>
        </security:ValidationInfo>
    </security:TrustEngine>
</MetadataFilter>
```