AttributeDefinitionConfiguration

Overview

Attribute Defintions produce a single IdPAttribute object, which may then be passed along to a relying party by attaching one or more AttributeEncoders to it. The ability to attach encoders is what distinguishes them from an IdPAttribute produced by data connectors. Attribute definitions may, but need not, be based on the output of a data connector, and often transform their input.

Note that multiple encoders can be attached to a single IdPAttribute object.

Schema Name and Location

All elements and plugins described in this page and its children are defined by the `urn:mace:shibboleth:2.0:resolver` namespace, the schema for which is located at http://shibboleth.net/schema/idp/shibboleth-attribute-resolver.xsd

Prior to V3.3 supplied plugins were defined by a schema type (`xsi:type`) in the `urn:mace:shibboleth:2.0:resolver:ad` namespace, the schema for which is located at http://shibboleth.net/schema/idp/shibboleth-attribute-resolver-ad.xsd. This is still supported, but every element or type in the `urn:mace:shibboleth:2.0:resolver:ad` namespace has an equivalently-named version in the `urn:mace:shibboleth:2.0:resolver` namespace.

Common Attributes

Name	Type	Default	Description
`id`	String		Identifier for the IdPAttribute as well as its definition. This is used for logging and to establish dependencies and relationships between connectors and definitions.
`activationConditionRef`	Bean Reference		Bean ID of a condition to decide whether to resolve this definition, see here. Mutually exclusive with `relyingParties`
`relyingParties` **3.4**	space-delimited list		List of entity IDs for which this Attribute Definition should be resolved. Mutually exclusive with `activationConditionRef`
`dependencyOnly`	boolean	false	If set to true, the attribute is not exposed outside the resolution process and is available solely within the resolution process

`sourceAttributeID`	String		*DEPRECATED in V3.4* **NOTE**: This attribute *only* applies when dependencies are supplied via the deprecated `<Dependency>` Element, and is ignored otherwise. This defines the name of an IdPAttribute used as input to the attribute definition, and can only be applied to some definition types. The source attribute may be the output of another attribute definition or the output of a DataConnector. If any data connectors are used as dependencies, the source attribute MUST be identified or an error will result.
`profileContextStrategyRef`	Bean Reference		Bean ID of a function injected to override the normal lookup process for the request's ProfileRequestContext

Common Child Elements

All Attribute Definitions can have zero or more of each the following three child elements.

Prior to V3.3 the child elements had to be specified in a strict order, with the Common Child Elements coming first. This has been relaxed in V3.3.

Name	Cardinality	Description
`<Dependency>`	0 or more	*DEPRECATED in V3.4* This element has a single attribute (ref="whatever") whose content is the ID of an attribute definition or data connector whose output is an input to this attribute definition
`<InputAttributeDefinition>` **3.4**	0 or more	This element identifies an attribute definition which is an input to this attribute definition.
`<InputDataConnector>` **3.4**	0 or more	This element identifies a data connector whose attributes are to be input to this attribute definition.
`<DisplayName>`	0 or more	A human readable name for this attribute. This name may, for example, be displayed to the user to consent to the attribute's release. If multiple display names are used, then they should bear an `xml:lang` attribute to distinguish them.
`<DisplayDescription>`	0 or more	A human readable description of for this attribute. This name may, for example, be displayed to the user to consent to the attribute's release. If multiple display descriptions are used, then they should bear an `xml:lang` attribute to distinguish them.
`<AttributeEncoder>`	0 or more	A definition of how an attribute will be encoded for inclusion in a message to a relying party. These are distinguished by an `xsi:type` attribute, and the different types are documented here.

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

AttributeDefinition Plugin Types

Attribute Definitions are distinguished by their schema type, which is inside the `xsi:type` XML attribute. The following types are supported:

xsi:type	Function
`Simple`	Copies an input attribute to an output attribute. Typically this is used to 'expose' attributes which are sourced from a DataConnector.
`PrincipalName`	Exposes the subject's canonicalized principal name as an attribute definition.
`Scoped`	Applies a (fixed) scope to the input attribute's values
`Prescoped`	Splits input attribute values into values and scopes
`RegexSplit`	Splits input attribute values according to a regular expression
`ScriptedAttribute`	Generates an attributes using a JSR-223 script
`Mapped`	Allows many to many mapping of input values to output values according to regular expression mapping rules

`Template`	Feeds the input values (potentially from multiple input attributes) into a Velocity template to construct output values
`SubjectDerived` 3.3	Extracts data from the authenticated Subject(s)
`ContextDerived` 3.3	Extract arbitrary data from the request context via a Function bean
`PrincipalAuthenticationMethod`	DEPRECATED, exposes the authentication flow used to authenticate the subject for front-channel requests
`TransientId`	DEPRECATED, see the V2 Documentation for details
`CryptoTransientId`	DEPRECATED, see the V2 Documentation for details
`SAML1NameIdentifier`	DEPRECATED, see the V2 Documentation for details
`SAML2NameID`	DEPRECATED, see the V2 Documentation for details