XACML v3.0 Dynamic Attribute Authority Version 1.0

Committee Specification 01

25 January 2022

This stage:
https://docs.oasis-open.org/xacml/xacml-3.0-dyn-attr/v1.0/cs01/xacml-3.0-dyn-attr-v1.0-cs01.docx (Authoritative)
https://docs.oasis-open.org/xacml/xacml-3.0-dyn-attr/v1.0/cs01/xacml-3.0-dyn-attr-v1.0-cs01.html
https://docs.oasis-open.org/xacml/xacml-3.0-dyn-attr/v1.0/cs01/xacml-3.0-dyn-attr-v1.0-cs01.pdf

Previous stage:
https://docs.oasis-open.org/xacml/xacml-3.0-dyn-attr/v1.0/csd01/xacml-3.0-dyn-attr-v1.0-csd01.docx (Authoritative)
https://docs.oasis-open.org/xacml/xacml-3.0-dyn-attr/v1.0/csd01/xacml-3.0-dyn-attr-v1.0-csd01.html
https://docs.oasis-open.org/xacml/xacml-3.0-dyn-attr/v1.0/csd01/xacml-3.0-dyn-attr-v1.0-csd01.pdf

Latest stage:
https://docs.oasis-open.org/xacml/xacml-3.0-dyn-attr/v1.0/xacml-3.0-dyn-attr-v1.0.docx (Authoritative)
https://docs.oasis-open.org/xacml/xacml-3.0-dyn-attr/v1.0/xacml-3.0-dyn-attr-v1.0.html
https://docs.oasis-open.org/xacml/xacml-3.0-dyn-attr/v1.0/xacml-3.0-dyn-attr-v1.0.pdf

Technical Committee:
OASIS eXtensible Access Control Markup Language (XACML) TC

Chairs:
Hal Lockhart (harold.w.lochhart@gmail.com), Individual
Bill Parducci (bill@parducci.net), Individual

Editor:
Steven Legg (steven.legg@viewds.com), ViewDS Identity Solutions

Related work:
This document is related to:

- *eXtensible Access Control Markup Language (XACML) Version 3.0 Plus Errata 01.* Edited by Erik Rissanen. 12 July 2017. OASIS Standard incorporating Approved Errata. http://docs.oasis-open.org/xacml/3.0/errata01/os/xacml-3.0-core-spec-errata01-os-complete.html. Latest stage: http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-en.html.

Abstract:
This specification defines a new XACML system component, the Dynamic Attribute Authority, which augments the request context of an XACML authorization request with additional attributes and attribute values that are generated on demand according to a set of rules. The rules are expressed as XACML policies, use obligations to specify the additional attributes and values, and are processed in the normal manner of a Policy Decision Point. This means that a Dynamic Attribute Authority can be readily constructed from existing XACML system components.

A primary use case for the Dynamic Attribute Authority is role enablement, where the dynamic attribute in question is the subject role.

Status:
This document was last revised or approved by the OASIS eXtensible Access Control Markup Language (XACML) TC on the above date. The level of approval is also listed above. Check the "Latest stage" location noted above for possible later revisions of this document. Any other numbered Versions and other technical work produced by the Technical Committee (TC) are listed at https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml#technical.

TC members should send comments on this specification to the TC's email list. Others should send comments to the TC's public comment list, after subscribing to it by following the instructions at the "Send A Comment" button on the TC's web page at https://www.oasis-open.org/committees/xacml/.

This specification is provided under the RF on Limited Terms Mode of the OASIS IPR Policy, the mode chosen when the Technical Committee was established. For information on whether any patents have been disclosed that may be essential to implementing this specification, and any offers of patent licensing terms, please refer to the Intellectual Property Rights section of the TC's web page (https://www.oasis-open.org/committees/xacml/ipr.php).

Note that any machine-readable content (Computer Language Definitions) declared Normative for this Work Product is provided in separate plain text files. In the event of a discrepancy between any such plain text file and display content in the Work Product's prose narrative document(s), the content in the separate plain text file prevails.

Key words:
The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] and [RFC8174] when, and only when, they appear in all capitals, as shown here.

Citation format:
When referencing this document, the following citation format should be used:

[XACML-Dyn-Attr-v3.0]

XACML v3.0 Dynamic Attribute Authority Version 1.0. Edited by Steven Legg. 25 January 2022. OASIS Committee Specification 01. https://docs.oasis-open.org/xacml/xacml-3.0-dyn-attr/v1.0/cs01/xacml-3.0-dyn-attr-v1.0-cs01.html. Latest stage: https://docs.oasis-open.org/xacml/xacml-3.0-dyn-attr/v1.0/xacml-3.0-dyn-attr-v1.0.html.

Notices:

Table of Contents

1 Introduction

[All text is normative unless otherwise labeled]

[Informative]

The RBAC Profile **[XACML-3.0-RBAC]** refers to a processing entity called a Role Enablement Authority that assigns role attributes and values to users or enables role attributes and values during a user's session. Although the final profile prescribes no specific form for a Role Enablement Authority, intermediate drafts did describe a mechanism using a separate PDP and XACML policies to enable role attributes and values during a user's session, which was subsequently removed.

This erstwhile mechanism was conceptually inefficient. It required the client of the role enablement authority (which is likely to be a context handler) to send a request for each role, called a role enablement request, that effectively asked the question "is this role enabled for this access subject?". The client would necessarily need a complete list of roles to ask about; if there were 1,000 roles then there would be 1,000 role enablement requests.

In addition, the role enablement request used the access-subject and environment categories of the ***initial request*** but identified the action as role enablement and set the resource to the role identifier. This meant that a role enablement decision could not be made based on the action and resource of the ***initial request***. For example, a particular role could not be enabled only for *querying* of resources in a specific location.

What we really want to ask is "what roles are enabled for the access subject given this entire request context?" and preferably to do so with a single request to a Role Enablement Authority. This profile defines a method to ask this question in one request within a more general framework that allows us to ask what values of arbitrary attributes of arbitrary categories are enabled given the entire request context.

This more general authority is called a Dynamic Attribute authority (***DAA***). It is a rules engine that takes attributes in a request context and runs through its rules to generate a collection of attribute values for one or more attribute types (it can do more than one attribute type simultaneously). Insofar as the ***DAA*** is generating values of the subject role attribute it is acting as a Role Enablement Authority.

In order to reuse existing capabilities, and thereby simplify implementation, the rules held by the ***DAA*** are in the form of XACML policies and policy sets, i.e., ***DA policies*** and ***DA policy sets***, that are processed in the usual manner for a PDP. The extended data flow model that includes the DAA is described in Section 2.

The attribute values to be enabled are specified by two kinds of obligations described in Section 3. The first kind of obligation specifies values of a nominated attribute that are enabled (inclusions) and the other kind of obligation specifies exclusions that override the values specified by the first kind of obligation. In their simplest form these obligations contain lists of literal values. The ***DA policies*** mostly contain permit rules that, if applicable, return (through these obligations) either values to include or values to exclude. The context handler unions the values of the inclusion obligations and subtracts the values of the exclusion obligations for each nominated attribute to get a final list of enabled values for that attribute. The enabled values are added to the ***initial request*** to form the ***final request***, which is then processed as usual.

The rules that return exclusions take the place of deny rules. An actual deny rule would override the permit rules for every attribute to enable when we only want to deny specific values of one specific attribute.

1.1 Glossary

1.1.1 Definitions of terms

Initial request

The authorization request received by the context handler from a Policy Enforcement Point (PEP).

Final request

The authorization request after it has been augmented by the context handler with any attributes and values from processing the obligations returned by the ***DAA***.

Dynamic attribute authority (DAA)

The system entity that evaluates an ***initial request*** against the collection of ***DA policies*** and ***DA policy sets*** to arrive at a set of obligations which when processed by the context handler result in additional attributes to add into the ***final request***. The ***DAA*** as described by this profile is functionally equivalent to a PDP.

Dynamic attribute category (DA category)

The category of an attribute potentially added to the ***final request***.

Dynamic attribute type (DA type)

The data-type of a value of an attribute potentially added to the ***final request***.

Dynamic attribute identifier (DA identifier)

The identifier of an attribute potentially added to the ***final request***.

Dynamic attribute issuer (DA issuer)

The optional issuer of an attribute potentially added to the ***final request***.

Dynamic attribute obligation (DA obligation)

An obligation defined in this profile.

Dynamic attribute policy (DA policy)

An XACML policy that is used by the ***DAA***. May be a component of an ***DA policy set***.

Dynamic attribute policy set (DA policy set)

An XACML policy set that is used by the ***DAA***. May be a component of another ***DA policy set***.

Dynamic attribute request (DA request)

A request generated from the initial request by the context handler and submitted to the ***DAA*** for evaluation.

Dynamic attribute rule (DA rule)

An XACML rule that is used by the ***DAA***. A component of an ***DA policy***.

1.1.2 Acronyms and abbreviations

1.1.3 Document conventions

- Naming conventions
- Font colors and styles
- Typographical conventions

The replacement text for the XML entity reference "`&xacml1;`" used in examples is `"urn:oasis:names:tc:xacml:1.0:"`.

The replacement text for the XML entity reference "`&xacml2;`" used in examples is `"urn:oasis:names:tc:xacml:2.0:"`.

The replacement text for the XML entity reference "`&xacml3;`" used in examples is `"urn:oasis:names:tc:xacml:3.0:"`.

2 Extended Data-flow Model

The context handler first submits the ***initial request*** to the ***DAA***. The ***initial request*** is evaluated by the ***DAA*** in the same manner that a PDP evaluates an authorization request except that the applicable policy is the collection of ***DA policies*** and ***DA policy sets***. Other policies and policy sets SHALL be ignored by the ***DAA***.

If the decision returned by the ***DAA*** is "`Permit`", then the ***DA obligations*** returned by the ***DAA*** are used by the context handler to transform the ***initial request*** into the ***final request***. The required processing is described in Section 3.

If the decision returned by the ***DAA*** is "`Deny`" or "`NotApplicable`", then the decision has no effect and the ***final request*** is identical to the ***initial request***.

If the decision returned by the ***DAA*** is "`Indeterminate`", then the original request SHALL immediately evaluate to "`Indeterminate`" without creating and submitting the ***final request***.

The ***DA obligations*** are discarded after processing by the context handler.

The ***DA policies*** and ***DA policy sets*** MAY contain `AdviceExpressions`, and `ObligationExpressions` for obligations other than the ones defined in this profile. If the ***DAA*** returns obligations that the context handler does not know how to satisfy under this extended data-flow model then the original request SHALL immediately evaluate to "`Indeterminate`" without creating and submitting the ***final request***. If the DAA returns advice that the context handler does not know how to process under this extended data-flow model then that advice SHALL be discarded. Note that the processing requirements imposed on a PEP by a previously defined obligation may not be appropriate to the context handler, nor appropriate to return to the PEP unaltered.

Any policy and policy set identifiers in the `PolicyIdentifierList` accompanying the decision returned by the ***DAA*** SHALL be retained to be added to the `PolicyIdentifierList` of the final result returned to the PEP.

The context handler submits the ***final request*** to the PDP. The PDP SHALL NOT use any of the ***DA policies*** or ***DA policy sets*** when evaluating the ***final request***.

In the case of a request for multiple decisions **[xacml-3.0-multiple-v1.0]**

, each individual request is treated as an ***initial request***. The context handler SHALL submit each such ***initial request*** to the ***DAA*** and use the outcome to form the corresponding ***final request*** for each individual request. The collection of ***final requests*** is then processed according to **[xacml-3.0-multiple-v1.0]**

.

3 Obligations

The `Obligation` elements for ***DA obligations*** are processed collectively.

The inclusion obligations are processed first to generate attribute values that are each added to a distinct value set associated with a specific XACML attribute according to that value's ***DA category***, ***DA identifier***, ***DA type*** and ***DA issuer***. The ***DA category***, ***DA identifier***, ***DA type*** and ***DA issuer*** are determined from the content of the Obligation element in a manner specific to the identified obligation. Duplicate values are discarded, where equality is determined according to the `type-equal` function **XACML-v3.0-Errata01-complete** for the ***DA type***. Equality is implementation-defined for data-types that do not have a `type-equal` function.

The exclusion obligations are then processed to remove zero or more values from these sets.

Finally, the initial request is transformed into the final request by adding the values from the values sets as XACML attribute values, replacing any existing values of the same attribute.

The ***DA issuer*** may be unspecified, in which case it takes on the special value of null. The null value is distinct from any other string value, including the empty string.

3.1 Common Attribute Assignments

Some of the ***DA obligations*** specify the ***DA category***, ***DA identifier***, ***DA type*** or ***DA issuer*** using separate `AttributeAssignment` elements.

A ***DA category*** attribute assignment is an `AttributeAssignment` element with `urn:oasis:names:tc:xacml:3.0:daa:attribute:category` as the value of its `AttributeId` XML attribute. The value of its `DataType` XML attribute MUST be `http://www.w3.org/2001/XMLSchema#anyURI`. It MUST NOT have a `Category` or `Issuer` XML attribute. The ***DA category*** is the character data content of the element, which SHOULD be the URI of an XACML attribute category.

A ***DA identifier*** attribute assignment is an `AttributeAssignment` element with `urn:oasis:names:tc:xacml:3.0:daa:attribute:attribute-id` as the value of its `AttributeId` XML attribute. The value of its `DataType` XML attribute MUST be `http://www.w3.org/2001/XMLSchema#anyURI`. It MUST NOT have a `Category` or `Issuer` XML attribute. The ***DA identifier*** is the character data content of the element, which SHOULD be the URI of an XACML attribute.

A ***DA type*** attribute assignment is an `AttributeAssignment` element with `urn:oasis:names:tc:xacml:3.0:daa:attribute:data-type` as the value of its `AttributeId` XML attribute. The value of its `DataType` XML attribute MUST be `http://www.w3.org/2001/XMLSchema#anyURI`. It MUST NOT have a `Category` or `Issuer` XML attribute. The ***DA type*** is the character data content of the element, which SHOULD be the URI of an XACML data-type.

A ***DA issuer*** attribute assignment is an `AttributeAssignment` element with `urn:oasis:names:tc:xacml:3.0:daa:attribute:issuer` as the value of its `AttributeId` XML attribute. The value of its `DataType` XML attribute MUST be `http://www.w3.org/2001/XMLSchema#string`. It MUST NOT have a `Category` or `Issuer` XML attribute. The ***DA issuer*** is the character data content of the element, which identifies an issuer.

3.2 Inclusion Obligations

3.2.1 The include Obligation

The `include` obligation specifies zero or more attribute values to be added to one or more value sets and has the identifier `urn:oasis:names:tc:xacml:3.0:daa:obligation:include`. It does not use any of the common attribute assignments. Each `AttributeAssignment` element within the `Obligation` element describes one attribute value. The ***DA category***, ***DA identifier*** and ***DA type*** are,

respectively, the values of the `Category`, `AttributeId` and `DataType` XML attributes of the element. The ***DA issuer*** is the value of the `Issuer` XML attribute, if present, otherwise it is null. The `DataType` XML attribute and character data content of the element define a single XACML attribute value. The ***DA category***, ***DA identifier***, ***DA type*** and ***DA issuer*** determine which value set the value is placed in, unless it is a duplicate value. Since each `AttributeAssignment` element can have different values for its XML attributes, a single `include` obligation can cause values to be added to more than one value set.

3.2.2 The include-values Obligation

The `include-values` obligation specifies zero or more attribute values to be added to one or more value sets with the same ***DA category***, ***DA identifier*** and ***DA issuer***, and has the identifier `urn:oasis:names:tc:xacml:3.0:daa:obligation:include-values`. The ***DA category***, ***DA identifier*** and ***DA issuer*** are specified by separate `AttributeAssignment` elements within the `Obligation` element.

There MUST be exactly one ***DA category*** attribute assignment.

There MUST be exactly one ***DA identifier*** attribute assignment.

There MAY be one ***DA issuer*** attribute assignment. If there is no ***DA issuer*** attribute assignment then the ***DA issuer*** is null.

There MAY be one or more `AttributeAssignment` elements with `urn:oasis:names:tc:xacml:3.0:daa:attribute:value` as the value of their `AttributeId` XML attribute. They MUST NOT have a `Category` or `Issuer` XML attribute. The `DataType` XML attribute and character data content of the element define a single XACML attribute value. The ***DA type*** is the value of the `DataType` XML attribute. The value is added to the value set with the corresponding ***DA category***, ***DA identifier***, ***DA type*** and ***DA issuer***, unless it is a duplicate. These elements MAY have different values for their `DataType` XML attribute.

No other `AttributeAssignment` elements are permitted.

The order of the `AttributeAssignment` elements is not restricted.

If the `Obligation` element does not satisfy the requirements in this section then the obligation SHALL be treated as unknown.

The `include-values` obligation can be thought of as a long-form version of the `include` obligation, but has the additional ability to compute the ***DA category***, ***DA identifier*** or ***DA issuer*** using an XACML expression instead of them being restricted to literal URIs.

3.3 Exclusion Obligations

3.3.1 The exclude Obligation

The `exclude` obligation specifies zero or more attribute values to be removed from one or more value sets and has the identifier `urn:oasis:names:tc:xacml:3.0:daa:obligation:exclude`. It does not use any of the common attribute assignments. Each `AttributeAssignment` element within the `Obligation` element describes one attribute value. The ***DA category***, ***DA identifier*** and ***DA type*** are, respectively, the values of the `Category`, `AttributeId` and `DataType` XML attributes of the element. The ***DA issuer*** is the value of the `Issuer` XML attribute, if present, otherwise it is null. The `DataType` XML attribute and character data content of the element define a single XACML attribute value. The ***DA category***, ***DA identifier*** and ***DA type*** and ***DA issuer*** determine from which value set the value is removed, where value equality is determined according to the `type-equal` function **[XACML-v3.0-Errata01-complete]** for the ***DA type***. Equality is implementation-defined for data-types that do not have a `type-equal` function. The `AttributeAssignment` element has no effect if the value is not in the value set. Since each `AttributeAssignment` element can have different values for its XML attributes, a single `exclude` obligation can cause values to be removed from more than one value set.

3.3.2 The exclude-values Obligation

The `exclude-values` obligation specifies zero or more attribute values to be removed from one or more value sets with the same ***DA category***, ***DA identifier*** and ***DA issuer***, and has the identifier `urn:oasis:names:tc:xacml:3.0:daa:obligation:exclude-values`. The ***DA category***, ***DA identifier*** and ***DA issuer*** are specified by separate `AttributeAssignment` elements within the `Obligation` element.

There MUST be exactly one ***DA category*** attribute assignment.

There MUST be exactly one ***DA identifier*** attribute assignment.

There MAY be one ***DA issuer*** attribute assignment. If there is no ***DA issuer*** attribute assignment then the ***DA issuer*** is null.

There MAY be one or more `AttributeAssignment` elements with `urn:oasis:names:tc:xacml:3.0:daa:attribute:value` as the value of their `AttributeId` XML attribute. They MUST NOT have a `Category` or `Issuer` XML attribute. The `DataType` XML attribute and character data content of the element define a single XACML attribute value. The ***DA type*** is the value of the `DataType` XML attribute. The value is removed from the value set with the corresponding ***DA category***, ***DA identifier***, ***DA type*** and ***DA issuer***, if it is present. These elements MAY have different values for their `DataType` XML attribute.

No other `AttributeAssignment` elements are permitted.

The order of the `AttributeAssignment` elements is not restricted.

If the `Obligation` element does not satisfy the requirements in this section then the obligation SHALL be treated as unknown.

The `exclude-values` obligation can be thought of as a long-form version of the `exclude` obligation, but has the additional ability to compute the ***DA category***, ***DA identifier*** or ***DA issuer*** using an XACML expression instead of them being restricted to literal URIs.

3.3.3 The exclude-matching-values Obligation

The `exclude-matching-values` obligation specifies zero or more attribute values to be removed from one value set and has the identifier `urn:oasis:names:tc:xacml:3.0:daa:obligation:exclude-matching-values`. The ***DA category***, ***DA identifier***, ***DA type*** and ***DA issuer*** are specified by separate `AttributeAssignment` elements within the `Obligation` element.

There MUST be exactly one ***DA category*** attribute assignment.

There MUST be exactly one ***DA identifier*** attribute assignment.

There MAY be one ***DA issuer*** attribute assignment. If there is no ***DA issuer*** attribute assignment then the ***DA issuer*** is null.

There MUST be exactly one ***DA type*** attribute assignment.

There MUST be exactly one `AttributeAssignment` element with `urn:oasis:names:tc:xacml:3.0:daa:attribute:value` as the value of its `AttributeId` XML attribute. Call this the value attribute assignment. It MUST NOT have a `Category` or `Issuer` XML attribute. The `DataType` XML attribute and character data content of the element define a single XACML attribute value.

There MUST be exactly one `AttributeAssignment` element with `urn:oasis:names:tc:xacml:3.0:daa:attribute:function-id` as the value of its `AttributeId` XML attribute. The value of its `DataType` XML attribute MUST be `http://www.w3.org/2001/XMLSchema#anyURI`. It MUST NOT have a `Category` or `Issuer` XML attribute. The character data content of the element MUST be the URI of an XACML function that takes two arguments and returns a Boolean result. The first argument MUST have the same data-type as the value of the `DataType` XML attribute of the value attribute assignment. The data-type of the second argument MUST be the same as the ***DA type***.

No other `AttributeAssignment` elements are permitted.

The order of the `AttributeAssignment` elements is not restricted.

If the `Obligation` element does not satisfy the requirements in this section then the obligation SHALL be treated as unknown; otherwise, each value in the value set with the corresponding ***DA category***, ***DA identifier***, ***DA type*** and ***DA issuer*** that matches the specified value according to the nominated function is removed. That is, remove each value where the function returns true with the value defined by the value attribute assignment as the first argument and the value from the value set as the second argument.

3.3.4 The exclude-all-values Obligation

The `exclude-all-values` obligation specifies the removal of all attribute values from one value set and has the identifier `urn:oasis:names:tc:xacml:3.0:daa:obligation:exclude-all-values`. The ***DA category***, ***DA identifier***, ***DA type*** and ***DA issuer*** are specified by separate `AttributeAssignment` elements within the `Obligation` element.

There MUST be exactly one ***DA category*** attribute assignment.

There MUST be exactly one ***DA identifier*** attribute assignment.

There MUST be exactly one ***DA type*** attribute assignment.

There MAY be one ***DA issuer*** attribute assignment. If there is no ***DA issuer*** attribute assignment then the ***DA issuer*** is null.

No other `AttributeAssignment` elements are permitted.

The order of the `AttributeAssignment` elements is not restricted.

If the `Obligation` element does not satisfy the requirements in this section then the obligation SHALL be treated as unknown; otherwise, all the values in the value set with the corresponding ***DA category***, ***DA identifier***, ***DA type*** and ***DA issuer*** are removed.

3.4 Final Processing

Each value set, including those with no remaining values, is considered in turn to transform the ***initial request*** into the ***final request***. This section assumes the XML representation of an XACML request. An equivalent transformation using the JSON representation **xacml-json-v1.1** is permitted. Modifying the request context in lieu of transforming the ***initial request*** is permitted, provided it produces the same result as specified here.

Any attribute values in the initial request that are associated with the same category, attribute identifier, data type and issuer are removed. If the ***DA issuer*** is null then only values of the attribute without a specified issuer are removed.

If the ***initial request*** does not contain a category for the ***DA category*** then a category with the value of the ***DA category*** as its category identifier is created.

The remaining values of the value set are added to the appropriate category as `<AttributeValue>` elements enclosed in one or more `<Attribute>` elements. The value of the ***DA type*** is used as the value of the `DataType` XML attribute of the `<AttributeValue>` elements. Note that the XACML core specification **XACML-v3.0-Errata01-complete** allows attribute values to be distributed over multiple `<Attribute>` elements with the same `AttributeId` and `Issuer` XML attributes. The value of the ***DA identifier*** is used as the value of the `AttributeId` XML attribute. If the ***DA issuer*** is not null then the `Issuer` XML attribute is present with the value of the ***DA issuer***. The <`AttributeValue`> elements MAY be added to an existing `<Attribute>` element with the requisite `AttributeId` and `Issuer`. This situation can arise if another value set with the same ***DA category***, ***DA identifier*** and ***DA issuer***, but different ***DA type***, has already been processed.

If a value set has no remaining values, then the corresponding attribute will not appear in the ***final request***.

4 Writing DA Policies

[Informative]

In a ***DA policy set***, ***DA policy*** or ***DA rule***, the value of the `FulfillOn` attribute of an `<ObligationExpression>` element should always be `Permit` because only obligations accompanying a "`Permit`" decision from the ***DAA*** are added to the ***final request*** submitted to the PDP. Though the decision returned to the context handler by the ***DAA*** may be "`Permit`" or "`NotApplicable`", the final result returned to the PEP could be any of "`Permit`", "`Deny`", "`NotApplicable`" or "`Indeterminate`".

In order to ensure that all applicable attribute values are enabled it is necessary to ensure that all potentially applicable ***DA policy sets*** and ***DA policies*** are considered and that the decision returned by the ***DAA*** is not "`Deny`". To this end, ***DA policies*** should only contain rules where the `Effect` attribute is set to `Permit`. The combining algorithm for a ***DA policy*** should be chosen to ensure that all applicable rules are considered.

Where the rules potentially contribute dynamic attribute values that are independent of each other the deny-overrides and ordered-deny-overrides rule combining algorithms are suitable choices. These combining algorithms will cause the DAA to evaluate every rule because there are no deny rules to cause premature termination of the algorithm. The permit-unless-deny rule combining algorithm is also usable in the absence of deny rules since a "`Permit`" result from the ***DAA*** with no ***DA obligations*** has the same effect on the context handler as a "`NotApplicable`" result.

Where the rules potentially contribute dynamic attribute values that are dependent on each other, e.g., are mutually exclusive, the first-applicable rule combining algorithm may be appropriate.

Given that ***DA policies*** are designed to never return a "`Deny`" decision, the deny-overrides, ordered-deny-overrides and permit-unless-deny policy combining algorithms are suitable choices for the combining algorithm of a ***DA policy set***. If the attributes generated by the component ***DA policies*** and ***DA policy sets*** are meant to be mutually exclusive then the first-applicable policy combining algorithm may be appropriate.

5 Examples

[Informative]

5.1 Role Enablement Example

This example shows ***DA policies*** and ***DA policy sets*** being used for enablement of roles providing access to projects in a project management application.

Assume the following roles are defined and used in permission and role policy sets **[XACML-3.0-RBAC]** (not shown):

- `urn:example:xacml:roles:project-owner`

  This role grants a subject full administrative permissions over a project.

- `urn:example:xacml:roles:project-member`

  This role grants a subject update permissions over the components of a project.

- `urn:example:xacml:roles:project-observer`

  This role grants a subject permissions to view a project.

Assume that `project-owner` inherits the permissions of `project-member`, which inherits the permissions of `project-observer`.

A project is represented as a resource with the following attributes:

- `urn:oasis:names:tc:xacml:1.0:resource:resource-id`

  A unique project identifier.

- `urn:example:xacml:project-owners`

  Contains the subject identifiers of one or more owners of the project.

- `urn:example:xacml:project-members`

  Contains the subject identifiers of zero or more team members who work on the project.

- `urn:example:xacml:project-department`

  The department within the organization that sponsors the project.

- `urn:example:xacml:project-classification`

  A string value indicating the sensitivity of the project's content. Typical values include "`unrestricted`" and "`confidential`".

A user is represented as a subject with the following attributes:

- `urn:oasis:names:tc:xacml:1.0:subject:subject-id`

  A unique user identifier.

- `urn:example:xacml:user-strong-authentication`

  A Boolean value indicating whether the user has authenticated with multiple factors or a secure hardware token.

- `urn:example:xacml:user-department`

  The department to which the user is assigned.

- `urn:example:xacml:user-on-leave`

  A Boolean value indicating whether the user is currently on annual leave.

The role enablement rules are contained in a single ***DA policy set***:

```
<PolicySet xmlns="&xacml3;core:schema:wd-17"
  PolicySetId="http://example.com/DA/enable-roles" Version="1.0"
  PolicyCombiningAlgId="&xacml3;policy-combining-algorithm:deny-overrides">

  <Target/>

  <Policy PolicyId="http://example.com/DA/enable-project-roles" Version="1.0"
    RuleCombiningAlgId="&xacml3;rule-combining-algorithm:deny-overrides">
    <Description>
      Project roles enablement rules.
    </Description>
    <Target/>

    <Rule RuleId="enable-project-owner-role" Effect="Permit">
      <Description>
        The project-owner role is enabled if the subject is listed as one of
        the project's owners.
      </Description>
      <Condition>
        <Apply FunctionId="&xacml1;function:string-at-least-one-member-of">
          <AttributeDesignator
            Category="&xacml1;subject-category:access-subject"
            AttributeId="&xacml1;subject:subject-id"
            DataType="http://www.w3.org/2001/XMLSchema#string"
            MustBePresent="false"/>
          <AttributeDesignator
            Category="&xacml3;attribute-category:resource"
            AttributeId="urn:example:xacml:project-owners"
            DataType="http://www.w3.org/2001/XMLSchema#string"
            MustBePresent="false"/>
        </Apply>
      </Condition>
      <ObligationExpressions>
        <ObligationExpression ObligationId="&xacml3;daa:obligation:include"
          FulfillOn="Permit">
          <AttributeAssignmentExpression
            Category="&xacml1;subject-category:access-subject"
            AttributeId="&xacml2;subject:role">
            <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#anyURI"
              >urn:example:xacml:roles:project-owner</AttributeValue>
          </AttributeAssignmentExpression>
        </ObligationExpression>
      </ObligationExpressions>
    </Rule>

    <Rule RuleId="enable-project-member-role" Effect="Permit">
      <Description>
        The project-member role is enabled if the subject is listed as one of
        the project's members.
      </Description>
      <Condition>
        <Apply FunctionId="&xacml1;function:string-at-least-one-member-of">
          <AttributeDesignator
            Category="&xacml1;subject-category:access-subject"
            AttributeId="&xacml1;subject:subject-id"
            DataType="http://www.w3.org/2001/XMLSchema#string"
            MustBePresent="false"/>
          <AttributeDesignator
            Category="&xacml3;attribute-category:resource"
            AttributeId="urn:example:xacml:project-members"
            DataType="http://www.w3.org/2001/XMLSchema#string"
            MustBePresent="false"/>
        </Apply>
      </Condition>
      <ObligationExpressions>
        <ObligationExpression ObligationId="&xacml3;daa:obligation:include"
```

```
        FulfillOn="Permit">
        <AttributeAssignmentExpression
          Category="&xacml1;subject-category:access-subject"
          AttributeId="&xacml2;subject:role">
          <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#anyURI"
            >urn:example:xacml:roles:project-member</AttributeValue>
        </AttributeAssignmentExpression>
      </ObligationExpression>
    </ObligationExpressions>
  </Rule>

  <Rule RuleId="enable-project-observer-role" Effect="Permit">
    <Description>
      The project-observer role is enabled if the subject and the project
      belong to the same department.
    </Description>
    <Condition>
      <Apply FunctionId="&xacml1;function:string-at-least-one-member-of">
        <AttributeDesignator
          Category="&xacml1;subject-category:access-subject"
          AttributeId="urn:example:xacml:user-department"
          DataType="http://www.w3.org/2001/XMLSchema#string"
          MustBePresent="false"/>
        <AttributeDesignator
          Category="&xacml3;attribute-category:resource"
          AttributeId="urn:example:xacml:project-department"
          DataType="http://www.w3.org/2001/XMLSchema#string"
          MustBePresent="false"/>
      </Apply>
    </Condition>
    <ObligationExpressions>
      <ObligationExpression ObligationId="&xacml3;daa:obligation:include"
        FulfillOn="Permit">
        <AttributeAssignmentExpression
          Category="&xacml1;subject-category:access-subject"
          AttributeId="&xacml2;subject:role">
          <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#anyURI"
            >urn:example:xacml:roles:project-observer</AttributeValue>
        </AttributeAssignmentExpression>
      </ObligationExpression>
    </ObligationExpressions>
  </Rule>

  <Rule RuleId="hide-confidential-projects" Effect="Permit">
    <Description>
      Observer roles are disabled if the project is classified as
      confidential.
    </Description>
    <Condition>
      <Apply FunctionId="&xacml1;function:string-is-in">
        <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string"
          >confidential</AttributeValue>
        <AttributeDesignator
          Category="&xacml3;attribute-category:resource"
          AttributeId="urn:example:xacml:project-classification"
          DataType="http://www.w3.org/2001/XMLSchema#string"
          MustBePresent="false"/>
      </Apply>
    </Condition>
    <ObligationExpressions>
      <ObligationExpression
        ObligationId="&xacml3;daa:obligation:exclude-matching-values"
        FulfillOn="Permit">
        <AttributeAssignmentExpression
          AttributeId="&xacml3;daa:attribute:category">
          <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#anyURI"
```

```
            >&xacml1;subject-category:access-subject</AttributeValue>
        </AttributeAssignmentExpression>
        <AttributeAssignmentExpression
          AttributeId="&xacml3;daa:attribute:attribute-id">
          <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#anyURI"
            >&xacml2;subject:role</AttributeValue>
        </AttributeAssignmentExpression>
        <AttributeAssignmentExpression
          AttributeId="&xacml3;daa:attribute:data-type">
          <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#anyURI"
            >http://www.w3.org/2001/XMLSchema#anyURI</AttributeValue>
        </AttributeAssignmentExpression>
        <AttributeAssignmentExpression
          AttributeId="&xacml3;daa:attribute:value">
          <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string"
            >urn:example:xacml:roles:.*-observer</AttributeValue>
        </AttributeAssignmentExpression>
        <AttributeAssignmentExpression
          AttributeId="&xacml3;daa:attribute:function-id">
          <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#anyURI"
            >&xacml2;function:anyURI-regexp-match</AttributeValue>
        </AttributeAssignmentExpression>
      </ObligationExpression>
    </ObligationExpressions>
  </Rule>

  <Rule RuleId="strong-authentication-required" Effect="Permit">
    <Description>
      The project-owner and project-member roles require strong
      authentication. These roles are disabled if strong authentication has
      not been used.
    </Description>
    <Condition>
      <Apply FunctionId="&xacml1;function:not">
        <Apply FunctionId="&xacml1;function:boolean-is-in">
          <AttributeValue
            DataType="http://www.w3.org/2001/XMLSchema#boolean"
            >true</AttributeValue>
          <AttributeDesignator
            Category="&xacml1;subject-category:access-subject"
            AttributeId="urn:example:xacml:user-strong-authentication"
            DataType="http://www.w3.org/2001/XMLSchema#boolean"
            MustBePresent="false"/>
        </Apply>
      </Apply>
    </Condition>
    <ObligationExpressions>
      <ObligationExpression ObligationId="&xacml3;daa:obligation:exclude"
        FulfillOn="Permit">
        <AttributeAssignmentExpression
          Category="&xacml1;subject-category:access-subject"
          AttributeId="&xacml2;subject:role">
          <Apply FunctionId="&xacml1;function:anyURI-bag">
            <AttributeValue
              DataType="http://www.w3.org/2001/XMLSchema#anyURI"
              >urn:example:xacml:roles:project-owner</AttributeValue>
            <AttributeValue
              DataType="http://www.w3.org/2001/XMLSchema#anyURI"
              >urn:example:xacml:roles:project-member</AttributeValue>
          </Apply>
        </AttributeAssignmentExpression>
      </ObligationExpression>
    </ObligationExpressions>
  </Rule>

</Policy>
```

```
  <Policy PolicyId="http://example.com/DA/general-exceptions" Version="1.0"
    RuleCombiningAlgId="&xacml3;rule-combining-algorithm:deny-overrides">
    <Description>Roles enablement exceptions.</Description>
    <Target/>

    <Rule RuleId="leave-exclusion" Effect="Permit">
      <Description>
        All roles are disabled if the subject is on leave.
      </Description>
      <Condition>
        <Apply FunctionId="&xacml1;function:boolean-is-in">
          <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#boolean"
            >true</AttributeValue>
          <AttributeDesignator
            Category="&xacml1;subject-category:access-subject"
            AttributeId="urn:example:xacml:user-on-leave"
            DataType="http://www.w3.org/2001/XMLSchema#boolean"
            MustBePresent="false"/>
        </Apply>
      </Condition>
      <ObligationExpressions>
        <ObligationExpression
          ObligationId="&xacml3;daa:obligation:exclude-all-values"
          FulfillOn="Permit">
          <AttributeAssignmentExpression
            AttributeId="&xacml3;daa:attribute:category">
            <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#anyURI"
              >&xacml1;subject-category:access-subject</AttributeValue>
          </AttributeAssignmentExpression>
          <AttributeAssignmentExpression
            AttributeId="&xacml3;daa:attribute:attribute-id">
            <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#anyURI"
              >&xacml2;subject:role</AttributeValue>
          </AttributeAssignmentExpression>
          <AttributeAssignmentExpression
            AttributeId="&xacml3;daa:attribute:data-type">
            <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#anyURI"
              >http://www.w3.org/2001/XMLSchema#anyURI</AttributeValue>
          </AttributeAssignmentExpression>
        </ObligationExpression>
      </ObligationExpressions>
    </Rule>

  </Policy>

</PolicySet>
```

5.1.1 Example Request 1

Suppose that the ***initial request*** contains the following attributes:

```
<Request xmlns="&xacml3;core:schema:wd-17"
         ReturnPolicyIdList="false" CombinedDecision="false">
  <Attributes Category="&xacml1;subject-category:access-subject">
    <Attribute AttributeId="&xacml1;subject:subject-id"
               IncludeInResult="false">
      <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string"
        >Bob</AttributeValue>
    </Attribute>
    <Attribute AttributeId="urn:example:xacml:user-strong-authentication"
               IncludeInResult="false">
      <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#boolean"
        >false</AttributeValue>
```

```
    </Attribute>
    <Attribute AttributeId="urn:example:xacml:user-department"
               IncludeInResult="false">
      <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string"
        >Online Sales</AttributeValue>
    </Attribute>
    <Attribute AttributeId="urn:example:xacml:user-on-leave"
               IncludeInResult="false">
      <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#boolean"
        >false</AttributeValue>
    </Attribute>
  </Attributes>
  <Attributes Category="&xacml3;attribute-category:resource">
    <Attribute AttributeId="&xacml1;resource:resource-id"
               IncludeInResult="false">
      <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string"
        >Enhanced Shopping Cart</AttributeValue>
    </Attribute>
    <Attribute AttributeId="urn:example:xacml:project-owners"
               IncludeInResult="false">
      <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string"
        >Alice</AttributeValue>
    </Attribute>
    <Attribute AttributeId="urn:example:xacml:project-members"
               IncludeInResult="false">
      <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string"
        >Alice</AttributeValue>
      <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string"
        >Bob</AttributeValue>
      <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string"
        >Carol</AttributeValue>
    </Attribute>
    <Attribute AttributeId="urn:example:xacml:project-department"
               IncludeInResult="false">
      <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string"
        >Online Sales</AttributeValue>
    </Attribute>
    <Attribute AttributeId="urn:example:xacml:project-classification"
               IncludeInResult="false">
      <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string"
        >unresticted</AttributeValue>
    </Attribute>
  </Attributes>
</Request>
```

The request would be expected to include other categories and attributes in practice, particularly an action, but since the example ***DA policy set*** does not reference any such attributes they have been omitted from the example.

The ***DAA*** would return the following result from evaluating the ***initial request***:

```
<Result xmlns="&xacml3;core:schema:wd-17">
  <Decision>Permit</Decision>
  <Status>
    <StatusCode Value="&xacml1;status:ok"/>
  </Status>
  <Obligations>
    <Obligation ObligationId="&xacml3;daa:obligation:include">
      <AttributeAssignment
        Category="&xacml1;subject-category:access-subject"
        AttributeId="&xacml2;subject:role"
        DataType="http://www.w3.org/2001/XMLSchema#anyURI"
        >urn:example:xacml:roles:project-member</AttributeAssignment>
    </Obligation>
```

```
    <Obligation ObligationId="&xacml3;daa:obligation:include">
      <AttributeAssignment
        Category="&xacml1;subject-category:access-subject"
        AttributeId="&xacml2;subject:role"
        DataType="http://www.w3.org/2001/XMLSchema#anyURI"
        >urn:example:xacml:roles:project-observer</AttributeAssignment>
    </Obligation>
    <Obligation ObligationId="&xacml3;daa:obligation:exclude">
      <AttributeAssignment
        Category="&xacml1;subject-category:access-subject"
        AttributeId="&xacml2;subject:role"
        DataType="http://www.w3.org/2001/XMLSchema#anyURI"
        >urn:example:xacml:roles:project-owner</AttributeAssignment>
      <AttributeAssignment
        Category="&xacml1;subject-category:access-subject"
        AttributeId="&xacml2;subject:role"
        DataType="http://www.w3.org/2001/XMLSchema#anyURI"
        >urn:example:xacml:roles:project-member</AttributeAssignment>
    </Obligation>
  </Obligations>
</Result>
```

The `enable-project-member-role` rule is applicable, because the subject is a member of the project, and contributes the `include` obligation for the `project-member` role.

The `enable-project-observer-role` rule is applicable, because the subject belongs to the same department as the project, and contributes the `include` obligation for the `project-observer` role.

The `strong-authentication-required` rule is applicable, because the subject has not used a strong authentication method, and contributes the `exclude` obligation for the `project-owner` and `project-member` roles.

None of the other rules are applicable.

The `include` obligations are processed to generate attribute values for the `project-member` and `project-observer` roles that are added to a single value set associated with the `access-subject` category, `subject-role` attribute and `anyURI` data-type. The `exclude` obligation is then processed to remove any attribute values for the `project-owner` (not present) and `project-member` (present) roles. The only remaining value is for the `project-observer` role.

The initial request does not contain the `subject-role` attribute so the remaining value of the value set is added to the request as a value of this attribute. That is, the following attribute is added to the `access-subject` category in the ***initial request*** to produce the ***final request***:

```
<Attribute AttributeId="&xacml2;subject:role" IncludeInResult="false">
  <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#anyURI"
    >urn:example:xacml:roles:project-observer</AttributeValue>
</Attribute>
```

5.1.2 Example Request 2

Suppose that the ***initial request*** contains the following attributes:

```
<Request xmlns="&xacml3;core:schema:wd-17"
         ReturnPolicyIdList="false" CombinedDecision="false">
  <Attributes Category="&xacml1;subject-category:access-subject">
    <Attribute AttributeId="&xacml1;subject:subject-id"
               IncludeInResult="false">
      <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string"
        >Alice</AttributeValue>
    </Attribute>
    <Attribute AttributeId="urn:example:xacml:user-strong-authentication"
```

```
                IncludeInResult="false">
      <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#boolean"
        >true</AttributeValue>
    </Attribute>
    <Attribute AttributeId="urn:example:xacml:user-department"
                IncludeInResult="false">
      <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string"
        >Online Sales</AttributeValue>
    </Attribute>
    <Attribute AttributeId="urn:example:xacml:user-on-leave"
                IncludeInResult="false">
      <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#boolean"
        >true</AttributeValue>
    </Attribute>
  </Attributes>
  <Attributes Category="&xacml3;attribute-category:resource">
    <Attribute AttributeId="&xacml1;resource:resource-id"
                IncludeInResult="false">
      <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string"
        >Enhanced Shopping Cart</AttributeValue>
    </Attribute>
    <Attribute AttributeId="urn:example:xacml:project-owners"
                IncludeInResult="false">
      <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string"
        >Alice</AttributeValue>
    </Attribute>
    <Attribute AttributeId="urn:example:xacml:project-members"
                IncludeInResult="false">
      <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string"
        >Alice</AttributeValue>
      <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string"
        >Bob</AttributeValue>
      <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string"
        >Carol</AttributeValue>
    </Attribute>
    <Attribute AttributeId="urn:example:xacml:project-department"
                IncludeInResult="false">
      <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string"
        >Online Sales</AttributeValue>
    </Attribute>
    <Attribute AttributeId="urn:example:xacml:project-classification"
                IncludeInResult="false">
      <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string"
        >unresticted</AttributeValue>
    </Attribute>
  </Attributes>
</Request>
```

The ***DAA*** would return the following result from evaluating the ***initial request***:

```
<Result xmlns="&xacml3;core:schema:wd-17">
  <Decision>Permit</Decision>
  <Status>
    <StatusCode Value="&xacml1;status:ok"/>
  </Status>
  <Obligations>
    <Obligation ObligationId="&xacml3;daa:obligation:include">
      <AttributeAssignment
        Category="&xacml1;subject-category:access-subject"
        AttributeId="&xacml2;subject:role"
        DataType="http://www.w3.org/2001/XMLSchema#anyURI"
        >urn:example:xacml:roles:project-owner</AttributeAssignment>
    </Obligation>
```

```
    <Obligation ObligationId="&xacml3;daa:obligation:include">
      <AttributeAssignment
        Category="&xacml1;subject-category:access-subject"
        AttributeId="&xacml2;subject:role"
        DataType="http://www.w3.org/2001/XMLSchema#anyURI"
        >urn:example:xacml:roles:project-member</AttributeAssignment>
    </Obligation>
    <Obligation ObligationId="&xacml3;daa:obligation:include">
      <AttributeAssignment
        Category="&xacml1;subject-category:access-subject"
        AttributeId="&xacml2;subject:role"
        DataType="http://www.w3.org/2001/XMLSchema#anyURI"
        >urn:example:xacml:roles:project-observer</AttributeAssignment>
    </Obligation>
    <Obligation ObligationId="&xacml3;daa:obligation:exclude-all-values">
      <AttributeAssignment
        AttributeId="&xacml3;daa:attribute:category"
        DataType="http://www.w3.org/2001/XMLSchema#anyURI"
        >&xacml1;subject-category:access-subject</AttributeAssignment>
      <AttributeAssignment
        AttributeId="&xacml3;daa:attribute:attribute-id"
        DataType="http://www.w3.org/2001/XMLSchema#anyURI"
        >&xacml2;subject:role</AttributeAssignment>
      <AttributeAssignment
        AttributeId="&xacml3;daa:attribute:data-type"
        DataType="http://www.w3.org/2001/XMLSchema#anyURI"
        >http://www.w3.org/2001/XMLSchema#anyURI</AttributeAssignment>
    </Obligation>
  </Obligations>
</Result>
```

The `enable-project-owner-role` rule is applicable, because the subject is an owner of the project, and contributes the `include` obligation for the `project-owner` role.

The `enable-project-member-role` rule is applicable, because the subject is a member of the project, and contributes the `include` obligation for the `project-member` role.

The `enable-project-observer-role` rule is applicable, because the subject belongs to the same department as the project, and contributes the `include` obligation for the `project-observer` role.

The `leave-exclusion` rule is applicable, because the subject is on leave, and contributes the `exclude-all-values` obligation for the `subject-role` attribute.

None of the other rules are applicable.

The `include` obligations are processed to generate attribute values for the `project-owner`, `project-member` and `project-observer` roles that are added to a single value set associated with the `access-subject` category, `subject-role` attribute and `anyURI` data-type. The `exclude-all-values` obligation is then processed to remove all values from the same value set. As there are no remaining values in any value set, nothing is added to the ***initial request*** and the ***final request*** is the same as the ***initial request***.

5.1.3 Example Request 3

Suppose that the ***initial request*** contains the following attributes:

```
<Request xmlns="&xacml3;core:schema:wd-17"
         ReturnPolicyIdList="false" CombinedDecision="false">
  <Attributes Category="&xacml1;subject-category:access-subject">
    <Attribute AttributeId="&xacml1;subject:subject-id"
               IncludeInResult="false">
      <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string"
```

```
        >Grace</AttributeValue>
    </Attribute>
    <Attribute AttributeId="urn:example:xacml:user-strong-authentication"
               IncludeInResult="false">
      <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#boolean"
        >true</AttributeValue>
    </Attribute>
    <Attribute AttributeId="urn:example:xacml:user-department"
               IncludeInResult="false">
      <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string"
        >Management Team</AttributeValue>
    </Attribute>
    <Attribute AttributeId="urn:example:xacml:user-on-leave"
               IncludeInResult="false">
      <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#boolean"
        >false</AttributeValue>
    </Attribute>
  </Attributes>
  <Attributes Category="&xacml3;attribute-category:resource">
    <Attribute AttributeId="&xacml1;resource:resource-id"
               IncludeInResult="false">
      <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string"
        >Demerger Planning</AttributeValue>
    </Attribute>
    <Attribute AttributeId="urn:example:xacml:project-owners"
               IncludeInResult="false">
      <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string"
        >Dave</AttributeValue>
    </Attribute>
    <Attribute AttributeId="urn:example:xacml:project-members"
               IncludeInResult="false">
      <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string"
        >Dave</AttributeValue>
      <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string"
        >Eve</AttributeValue>
      <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string"
        >Frank</AttributeValue>
    </Attribute>
    <Attribute AttributeId="urn:example:xacml:project-department"
               IncludeInResult="false">
      <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string"
        >Management Team</AttributeValue>
    </Attribute>
    <Attribute AttributeId="urn:example:xacml:project-classification"
               IncludeInResult="false">
      <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string"
        >confidential</AttributeValue>
    </Attribute>
  </Attributes>
</Request>
```

The ***DAA*** would return the following result from evaluating the ***initial request***:

```
<Result xmlns="&xacml3;core:schema:wd-17">
  <Decision>Permit</Decision>
  <Status>
    <StatusCode Value="&xacml1;status:ok"/>
  </Status>
  <Obligations>
    <Obligation ObligationId="&xacml3;daa:obligation:include">
      <AttributeAssignment
        Category="&xacml1;subject-category:access-subject"
        AttributeId="&xacml2;subject:role"
```



```
        DataType="http://www.w3.org/2001/XMLSchema#anyURI"
        >urn:example:xacml:roles:project-observer</AttributeAssignment>
    </Obligation>
    <Obligation ObligationId="&xacml3;daa:obligation:exclude-matching-values">
      <AttributeAssignment
        AttributeId="&xacml3;daa:attribute:category"
        DataType="http://www.w3.org/2001/XMLSchema#anyURI"
        >&xacml1;subject-category:access-subject</AttributeAssignment>
      <AttributeAssignment
        AttributeId="&xacml3;daa:attribute:attribute-id"
        DataType="http://www.w3.org/2001/XMLSchema#anyURI"
        >&xacml2;subject:role</AttributeAssignment>
      <AttributeAssignment
        AttributeId="&xacml3;daa:attribute:data-type"
        DataType="http://www.w3.org/2001/XMLSchema#anyURI"
        >http://www.w3.org/2001/XMLSchema#anyURI</AttributeAssignment>
      <AttributeAssignment
        AttributeId="&xacml3;daa:attribute:value"
        DataType="http://www.w3.org/2001/XMLSchema#string"
        >urn:example:xacml:roles:.*-observer</AttributeAssignment>
      <AttributeAssignment
        AttributeId="&xacml3;daa:attribute:function-id"
        DataType="http://www.w3.org/2001/XMLSchema#anyURI"
        >&xacml2;function:anyURI-regexp-match</AttributeAssignment>
    </Obligation>
  </Obligations>
</Result>
```

The `enable-project-observer-role` rule is applicable, because the subject belongs to the same department as the project, and contributes the `include` obligation for the `project-observer` role.

The `hide-confidential-projects` rule is applicable, because the project is classified as confidential, and contributes the `exclude-matching-values` obligation.

None of the other rules are applicable.

The `include` obligation is processed to generate an attribute value for the `project-observer` role that is added to a single value set associated with the `access-subject` category, `subject-role` attribute and `anyURI` data-type. The `exclude-matching-values` obligation is then processed to remove any values from the same value set that match according to the `anyURI-regexp-match` function with `urn:example:xacml:roles:.*-observer` as the regular expression. The value for the `project-observer` role matches and is removed from the value set. The matching that is performed is equivalent to this expression:

```
<Apply FunctionId="&xacml2;function:anyURI-regexp-match">
  <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string"
    >urn:example:xacml:roles:.*-observer</AttributeValue>
  <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#anyURI"
    >urn:example:xacml:roles:project-observer</AttributeValue>
</Apply>
```

As there are no remaining values in any value set, nothing is added to the ***initial request*** and the ***final request*** is the same as the ***initial request***.

5.2 Data Transformation Example

This example shows a ***DA policy*** being used to harmonize weight measurements between pounds and kilograms in a heterogeneous collection of PIPs and PEPs (perhaps the result of a recent corporate merger or a consequence of operating in diverse markets).

Suppose that a resource may have the following additional attributes:

- `urn:example:xacml:weight-lb`

Weight measurements for the resource in pounds.

- `urn:example:xacml:weight-kg`

  Weight measurements for the resource in kilograms.

The conversion rules are contained in a single ***DA policy***:

```
<Policy xmlns="&xacml3;core:schema:wd-17"
  PolicyId="http://example.com/DA/convert-weights" Version="1.0"
  RuleCombiningAlgId="&xacml3;rule-combining-algorithm:deny-overrides">
  <Description>
    Weight conversion rules.
    One pound equals 0.45359237 kilograms.
  </Description>
  <Target/>

  <Rule RuleId="pounds-to-kilograms" Effect="Permit">
    <Description>
      Convert any weight in pounds into kilograms.
    </Description>
    <Condition>
      <Apply FunctionId="&xacml1;function:integer-greater-than">
        <Apply FunctionId="&xacml1;function:double-bag-size">
          <AttributeDesignator
            Category="&xacml3;attribute-category:resource"
            AttributeId="urn:example:xacml:weight-lb"
            DataType="http://www.w3.org/2001/XMLSchema#double"
            MustBePresent="false"/>
        </Apply>
        <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#integer"
          >0</AttributeValue>
      </Apply>
    </Condition>
    <ObligationExpressions>
      <ObligationExpression ObligationId="&xacml3;daa:obligation:include"
        FulfillOn="Permit">
        <AttributeAssignmentExpression
          Category="&xacml3;attribute-category:resource"
          AttributeId="urn:example:xacml:weight-kg">
          <Apply FunctionId="&xacml3;function:map">
            <Function FunctionId="&xacml1;function:double-multiply"/>
            <AttributeDesignator
              Category="&xacml3;attribute-category:resource"
              AttributeId="urn:example:xacml:weight-lb"
              DataType="http://www.w3.org/2001/XMLSchema#double"
              MustBePresent="false"/>
            <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#double"
              >0.45359237</AttributeValue>
          </Apply>
        </AttributeAssignmentExpression>
      </ObligationExpression>
    </ObligationExpressions>
  </Rule>

  <Rule RuleId="kilograms-to-pounds" Effect="Permit">
    <Description>
      Convert any weight in kilograms into pounds.
    </Description>
    <Condition>
      <Apply FunctionId="&xacml1;function:integer-greater-than">
        <Apply FunctionId="&xacml1;function:double-bag-size">
          <AttributeDesignator
            Category="&xacml3;attribute-category:resource"
```

```
            AttributeId="urn:example:xacml:weight-kg"
            DataType="http://www.w3.org/2001/XMLSchema#double"
            MustBePresent="false"/>
        </Apply>
        <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#integer"
          >0</AttributeValue>
      </Apply>
    </Condition>
    <ObligationExpressions>
      <ObligationExpression ObligationId="&xacml3;daa:obligation:include"
        FulfillOn="Permit">
        <AttributeAssignmentExpression
          Category="&xacml3;attribute-category:resource"
          AttributeId="urn:example:xacml:weight-lb">
          <Apply FunctionId="&xacml3;function:map">
            <Function FunctionId="&xacml1;function:double-divide"/>
            <AttributeDesignator
              Category="&xacml3;attribute-category:resource"
              AttributeId="urn:example:xacml:weight-kg"
              DataType="http://www.w3.org/2001/XMLSchema#double"
              MustBePresent="false"/>
            <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#double"
              >0.45359237</AttributeValue>
          </Apply>
        </AttributeAssignmentExpression>
      </ObligationExpression>
    </ObligationExpressions>
  </Rule>

</Policy>
```

Suppose that the ***initial request*** contains the following attributes:

```
<Request xmlns="&xacml3;core:schema:wd-17"
         ReturnPolicyIdList="false" CombinedDecision="false">
  <Attributes Category="&xacml3;attribute-category:resource">
    <Attribute AttributeId="&xacml1;resource:resource-id"
               IncludeInResult="false">
      <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string"
        >Sugar</AttributeValue>
    </Attribute>
    <Attribute AttributeId="urn:example:xacml:weight-lb"
               IncludeInResult="false">
      <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#double"
        >1.0</AttributeValue>
      <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#double"
        >2.0</AttributeValue>
      <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#double"
        >4.0</AttributeValue>
    </Attribute>
  </Attributes>
</Request>
```

The request would be expected to include other categories and attributes in practice, particularly a subject and an action, but since the example ***DA policy*** does not reference any such attributes they have been omitted from the example.

The ***DAA*** would return the following result from evaluating the ***initial request***:

```
<Result xmlns="&xacml3;core:schema:wd-17">
  <Decision>Permit</Decision>
  <Status>
```

```
    <StatusCode Value="&xacml1;status:ok"/>
  </Status>
  <Obligations>
    <Obligation ObligationId="&xacml3;daa:obligation:include">
      <AttributeAssignment
        Category="&xacml3;attribute-category:resource"
        AttributeId="urn:example:xacml:weight-kg"
        DataType="http://www.w3.org/2001/XMLSchema#double"
        >0.45359237</AttributeAssignment>
      <AttributeAssignment
        Category="&xacml3;attribute-category:resource"
        AttributeId="urn:example:xacml:weight-kg"
        DataType="http://www.w3.org/2001/XMLSchema#double"
        >0.90718474</AttributeAssignment>
      <AttributeAssignment
        Category="&xacml3;attribute-category:resource"
        AttributeId="urn:example:xacml:weight-kg"
        DataType="http://www.w3.org/2001/XMLSchema#double"
        >1.81436948</AttributeAssignment>
    </Obligation>
  </Obligations>
</Result>
```

The `pounds-to-kilograms` rule is applicable, because the resource contains the `weight-lb` attribute, and contributes the `include` obligation.

The other rule is not applicable.

The `include` obligation is processed to generate attribute values that are added to a single value set associated with the `resource` category, `weight-kg` attribute and `double` data-type.

The initial request does not contain this attribute so the values of the value set are added to the request as values of this attribute. That is, the following attribute is added to the `resource` category in the ***initial request*** to produce the ***final request***:

```
<Attribute AttributeId="urn:example:xacml:weight-kg" IncludeInResult="false">
  <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#double"
    >0.45359237</AttributeValue>
  <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#double"
    >0.90718474</AttributeValue>
  <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#double"
    >1.81436948</AttributeValue>
</Attribute>
```

5.3 Alternative Final Processing

In the final processing step a dynamic attribute generated from the obligations returned by the ***DAA*** replaces the same attribute in the ***initial request*** if it has been provided by the PEP, i.e., the dynamic attribute overrides the attribute provided by the PEP. It is possible to achieve different effects by composing ***DA policies*** and ***DA policy sets*** in particular ways and through the judicious use of the issuer field. A different effect can be chosen for each dynamic attribute.

For illustration, the examples show the different effects being applied to an example attribute in the subject category with the identifier `urn:example:xacml:subject:generic`. The attributes provided by the PEP in the initial request will be called PEP attributes and attributes fetched from one or more PIPs will be called PIP attributes.

5.3.1 PEP and PIP Override DAA

There is a simple way to have PEP and PIP attributes override dynamic attributes that takes advantage of ***DA policies*** being designed to never produce "`Deny`" decisions. The ***DA policies*** applicable to a dynamic attribute that has no dependencies on any other dynamic attribute can generally be combined

into a single ***DA policy set***. That policy set can be made the second child of an additional ***DA policy set*** that uses the first-applicable policy combining rule. The first child is then set to a policy that evaluates to "`Permit`" if the attribute is not empty.

The additional ***DA policy set*** would look something like this:

```
<PolicySet xmlns="&xacml3;core:schema:wd-17"
  PolicySetId="http://example.com/DA/PEP-and-PIP-override-generic"
  Version="1.0"
  PolicyCombiningAlgId="&xacml1;policy-combining-algorithm:first-applicable">

  <Target/>

  <Policy PolicyId="http://example.com/DA/generic-present" Version="1.0"
    RuleCombiningAlgId="&xacml3;rule-combining-algorithm:deny-overrides">
    <Description>
      The policy is applicable if the generic attribute is provided by the
      PEP or can be fetched from a PIP.
    </Description>
    <Target/>

    <Rule RuleId="generic-present" Effect="Permit">
      <Description>
        The rule is applicable if the generic attribute has
        at least one value.
      </Description>
      <Condition>
        <Apply FunctionId="&xacml1;function:string-greater-than">
          <Apply FunctionId="&xacml1;function:string-bag-size">
            <AttributeDesignator
              Category="&xacml1;subject-category:access-subject"
              AttributeId="urn:example:xacml:subject:generic"
              DataType="http://www.w3.org/2001/XMLSchema#string"
              MustBePresent="false"/>
          </Apply>
          <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#integer"
            >0</AttributeValue>
        </Apply>
      </Condition>
    </Rule>

  </Policy>

  <PolicySet
    PolicySetId="http://example.com/DA/generic-attribute" Version="1.0"
    PolicyCombiningAlgId="&xacml3;policy-combining-algorithm:deny-overrides">
    <Description>
      Policies to generate the generic attribute. This policy set is not
      evaluated if the generic attribute already has values.
    </Description>
    <Target/>

    <!-- DA policies and DA policy sets returning obligations for the generic
        attribute go here. -->

  </PolicySet>

</PolicySet>
```

If the `generic` attribute has values provided by the PEP in the ***initial request*** or obtained by the context handler from a PIP, then the first policy in the policy set evaluates to "`Permit`" with no obligations and the policy set with all the policies for generating the `generic` dynamic attribute is not evaluated at all. No inclusion or exclusion obligations for the `generic` attribute are returned and there is no value set for the

`generic` attribute (not even an empty one). If the PEP provided the `generic` attribute in the ***initial request***, then it is retained for the ***final request***. If the PIP provided the `generic` attribute, then it will be provided again when the ***final request*** is evaluated.

If the `generic` attribute is not provided by the PEP in the ***initial request***, nor obtained by the context handler from any PIP, then the first policy in the policy set evaluates to "`NotApplicable`" and the policy set with all the policies for generating the `generic` dynamic attribute will be evaluated. If the final processing ends up with a non-empty value set for the `generic` attribute, then its values are added to the ***final request***.

It may be that the ***DA policies*** applicable to the `generic` attribute cannot be readily isolated into a single ***DA policy set*** because of interdependencies with other dynamic attributes. In this case the `exclude-all-values` obligation and issuer field can be exploited. The obligation expressions in the ***DA policies*** applicable to the `generic` attribute are altered to specify a distinct issuer string that identifies the ***DAA***, e.g., "DAA". An additional ***DA policy*** is created that returns the `exclude-all-values` obligation for the `generic` attribute with "DAA" as issuer if the `generic` attribute is not empty.

The additional ***DA policy*** would look something like this:

```
<Policy xmlns="&xacml3;core:schema:wd-17"
  PolicyId="http://example.com/DA/generic-present-exclude" Version="1.0"
  RuleCombiningAlgId="&xacml3;rule-combining-algorithm:deny-overrides">
  <Description>
    The policy is applicable if the generic attribute is provided by the
    PEP or can be fetched from a PIP.
  </Description>
  <Target/>

  <Rule RuleId="generic-present" Effect="Permit">
    <Description>
      The rule is applicable if the generic attribute has
      at least one value.
    </Description>
    <Condition>
      <Apply FunctionId="&xacml1;function:string-greater-than">
        <Apply FunctionId="&xacml1;function:string-bag-size">
          <AttributeDesignator
            Category="&xacml1;subject-category:access-subject"
            AttributeId="urn:example:xacml:subject:generic"
            DataType="http://www.w3.org/2001/XMLSchema#string"
            MustBePresent="false"/>
        </Apply>
        <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#integer"
          >0</AttributeValue>
      </Apply>
    </Condition>
    <ObligationExpressions>
      <ObligationExpression
        ObligationId="&xacml3;daa:obligation:exclude-all-values"
        FulfillOn="Permit">
        <AttributeAssignmentExpression
          AttributeId="&xacml3;daa:attribute:category">
          <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#anyURI"
            >&xacml1;subject-category:access-subject</AttributeValue>
        </AttributeAssignmentExpression>
        <AttributeAssignmentExpression
          AttributeId="&xacml3;daa:attribute:attribute-id">
          <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#anyURI"
            >urn:example:xacml:subject:generic</AttributeValue>
        </AttributeAssignmentExpression>
        <AttributeAssignmentExpression
          AttributeId="&xacml3;daa:attribute:data-type">
          <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#anyURI"
```

```
            >http://www.w3.org/2001/XMLSchema#string</AttributeValue>
        </AttributeAssignmentExpression>
        <AttributeAssignmentExpression
          AttributeId="&xacml3;daa:attribute:issuer">
          <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string"
            >DAA</AttributeValue>
        </AttributeAssignmentExpression>
      </ObligationExpression>
    </ObligationExpressions>
  </Rule>

</Policy>
```

If the `generic` attribute has values provided by the PEP in the ***initial request*** or obtained by the context handler from a PIP, then the additional policy evaluates to "`Permit`" and returns the `exclude-all-values` obligation for the `generic` attribute with "DAA" as issuer. The policies for generating the `generic` dynamic attribute are evaluated where applicable and may return further inclusion or exclusion obligations for the `generic` attribute also with "DAA" as issuer. The final processing ends up with an empty value set for the `generic` attribute with "DAA" as issuer because of the `exclude-all-values` obligation. The `generic` attribute with "DAA" as issuer should not have been provided in the ***initial request*** so there is nothing to replace and no dynamic attribute values are added to the ***final request***. If the PEP provided the `generic` attribute (without an issuer) in the ***initial request***, then it is retained for the ***final request***. It is not affected by the empty value set because the issuer is different. If the PIP provided the `generic` attribute, then it will be provided again when the ***final request*** is evaluated.

If the `generic` attribute is not provided by the PEP in the ***initial request***, nor obtained by the context handler from any PIP, then the additional policy evaluates to "`NotApplicable`". The policies for generating the `generic` dynamic attribute are evaluated where applicable and may return inclusion or exclusion obligations for the `generic` attribute with "DAA" as the issuer. If the final processing ends up with a non-empty value set for the `generic` attribute with "DAA" as the issuer, then its values are added to the ***final request***. Assuming that the attribute designators in conventional policies and policy sets reference the `generic` attribute without specifying an issuer they will nonetheless select the values of the `generic` attribute with "DAA" as the issuer in the ***final request***.

5.3.2 PEP Overrides DAA

If it is desired that the dynamic attribute is only overridden by an attribute explicitly provided by the PEP in the ***initial request*** and not by an attribute fetched from a PIP, then this can be achieved by the PEP specifying a distinct issuer string for the `generic` attribute that identifies the PEP, e.g., "PEP". The examples in the preceding section can be used except with an `Issuer` XML Attribute with the value "PEP" added to the `<AttributeDesignator>` elements. In this case the attribute designators will not select any PIP attribute values.

The PEP can also use the `Issuer` XML Attribute to control the override for an attribute. If it sets the issuer to "PEP", then the attribute it provides will override the dynamic attribute. If it omits the issuer, then the dynamic attribute will override the provided attribute if the dynamic attribute has any values.

5.3.3 PEP Attributes and Dynamic Attributes Merge

If it is desired that any dynamic attribute values should be added to any PEP attribute values, then this can be achieved by specifying a distinct issuer that identifies the ***DAA*** in the obligation expressions in the ***DA policies***. The dynamic attribute will not replace the PEP attribute because the issuer is different, and both bags of values will appear in the ***final request***. Assuming that the attribute designators in conventional policies and policy sets reference the attribute without specifying an issuer, they will select both bags of values.

5.3.4 Merge with Exclusions

The exclusion obligations only affect values added by inclusion obligations. If it is desired that exclusion obligations should also apply to any attribute values provided by the PEP or PIP, then this can be achieved with an additional ***DA policy*** that copies PEP and PIP attributes into the dynamic attribute with an `include` obligation and a distinct issuer that identifies the ***DAA***. The obligation expressions in the ***DA policies*** applicable to the `generic` attribute are altered to also specify this same issuer.

The additional ***DA policy*** would look something like this:

```
<Policy xmlns="&xacml3;core:schema:wd-17"
  PolicyId="http://example.com/DA/generic-present-copy" Version="1.0"
  RuleCombiningAlgId="&xacml3;rule-combining-algorithm:deny-overrides">
  <Description>
    The policy is applicable if the generic attribute is provided by the
    PEP or can be fetched from a PIP.
  </Description>
  <Target/>

  <Rule RuleId="generic-present" Effect="Permit">
    <Description>
      The rule is applicable if the generic attribute has
      at least one value.
    </Description>
    <Condition>
      <Apply FunctionId="&xacml1;function:string-greater-than">
        <Apply FunctionId="&xacml1;function:string-bag-size">
          <AttributeDesignator
            Category="&xacml1;subject-category:access-subject"
            AttributeId="urn:example:xacml:subject:generic"
            DataType="http://www.w3.org/2001/XMLSchema#string"
            MustBePresent="false"/>
        </Apply>
        <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#integer"
          >0</AttributeValue>
      </Apply>
    </Condition>
    <ObligationExpressions>
      <ObligationExpression ObligationId="&xacml3;daa:obligation:include"
        FulfillOn="Permit">
        <AttributeAssignmentExpression
          Category="&xacml1;subject-category:access-subject"
          AttributeId="urn:example:xacml:subject:generic" Issuer="DAA">
          <AttributeDesignator
            Category="&xacml1;subject-category:access-subject"
            AttributeId="urn:example:xacml:subject:generic"
            DataType="http://www.w3.org/2001/XMLSchema#string"
            MustBePresent="false"/>
        </AttributeAssignmentExpression>
      </ObligationExpression>
    </ObligationExpressions>
  </Rule>

</Policy>
```

If the `generic` attribute has values provided by the PEP in the ***initial request*** or obtained by the context handler from a PIP, then the additional policy evaluates to "`Permit`" and returns an `include` obligation for the `generic` attribute with "DAA" as the issuer. The policies for generating the `generic` dynamic attribute are evaluated where applicable and may return further inclusion or exclusion obligations for the `generic` attribute with "DAA" as the issuer. The attribute designators in conventional policies and policy sets must reference the `generic` attribute specifying "DAA" as the issuer. This is to prevent the PIP attributes begin provided again when the ***final request*** is evaluated (whether or not this would occur is an

implementation choice of the context handler as defined in the XACML core specification **XACML-v3.0-Errata01-complete]**).

Instead of outputting obligations for the `generic` attribute with “DAA” as the issuer, the ***DA policies*** (including the one above) could output a different attribute identifier, e.g., `generic:daa`, without an issuer, that the attribute designators in conventional policies and policy sets could reference instead.

The `<Condition>` element is not actually required. If it is omitted, then the `include` obligation will always be returned, but it will contain no `<AttributeAssignment>` elements, and have no effect, if the `generic` attribute has no values.

6 Architectural Considerations

[Informative]

Although this profile aims to maximize the reuse of existing XACML capabilities, it is still necessary to have an implementation of a context handler (call this the principal context handler) that is able to send a request to the ***DAA*** and process the response appropriately. However, the ***DAA*** itself could be a standalone, unmodified, standard XACML context handler and PDP. There is nothing special it needs to do with respect to this profile and it doesn't need to know it is acting as a ***DAA***. It would see the principal context handler as just another PEP. If this profile is used in this way, then it is important to note that the principal context handler and the ***DAA*** context handler are not necessarily accessing the same collection of PIPs. This means that an attribute designator in a conventional policy may evaluate to a different set of attribute values from the same attribute designator evaluated by the ***DAA***, simply because the PIP supplying the attribute is different in each case. Arranging for the principal context handler and the ***DAA*** context handler to have equivalent access to *exactly* the same PIPs would prevent this situation; otherwise, the ***DAA*** policy writers should be made aware of the potential difference in behavior. The principal context handler and the ***DAA*** context handler may also have different values for environment attributes, including slightly different values for the current-time attribute. The principal context handler may be able to mitigate this by adding its environment attributes to the ***initial request*** to override the attribute values the ***DAA*** context handler would otherwise use.

An alternative to using a standalone ***DAA*** is for the principal context handler to use its own facilities and the co-located PDP to act as the ***DAA***. In this case it is necessary to distinguish the policies and policy sets in some manner so that ***DA policies*** and ***DA policy sets*** are only used for ***DAA*** processing and conventional policies and policy sets are not. Implementors are free to achieve this in whatever way they choose. For example, they might use separate internal caches for the two classes of policy, or they might use a single cache but label the policies and policy sets as either conventional policies and policy sets or ***DA policies*** and ***DA policy sets*** and have the principal context handler indicate to the PDP which labelled collection it should be using at each invocation. The principal context handler can, however, use a shared internal instance of the request context provided it notes which attribute values came from the ***initial request*** so that these values can be removed in the final processing, if required. With respect to other attributes, an attribute designator in a conventional policy will evaluate to the same set of attribute values as the same attribute designator evaluated by a ***DA policy*** because the principal context handler is handling both cases using the same request context, the same PIPs and the same values for the environment attributes.

An off the shelf policy administration point (PAP) can be used to manage ***DA policies*** and ***DA policy sets***, though the policy writer does not require, and does not benefit from, the full expressive power of XACML policies and policy sets. When composing ***DA policies*** and ***DA policy sets*** we are not interested in an access control decision. The permits, denies and combining algorithms are just a means to an end and a custom ***DAA*** PAP could be created that presents a simpler conceptual model to the user:

- Users don't need to be able to choose, or even see, the value of the `Effect` attribute in a rule of a ***DA policy***; it should always be `Permit`.
- In the absence of "`Deny`" results the deny-overrides, ordered-deny-overrides and permit-unless-deny combining algorithms are suitable for the ***DAA*** and produce equivalent results. A custom ***DAA*** PAP could just pick one, e.g., deny-overrides, and omit the other two. The remaining useful combining rule is first-applicable, so it boils down to a choice between deny-overrides (say) and first-applicable, which could be presented to the policy writer as a Boolean choice between all applicable and first applicable.
- Users don't need to be able to choose, or even see, the value of the `FulfillOn` attribute of an `<ObligationExpression>` element; it should always be `Permit`.
- Obligations and advice that are defined elsewhere without reference to this profile are generally not useful in ***DA policies*** and ***DA policy sets***. A custom ***DAA*** PAP could drop all support for advice and provide simplified, customized dialogs for managing only the specific obligations defined in this profile. The detailed construction of ***DA category***, ***DA identifier***, ***DA type*** and ***DA issuer*** attribute assignments is something that can be hidden from the user.

- The overall user experience provided by a custom ***DAA*** PAP could be aligned to the task of writing rules for generating attributes rather than writing policies for generating decisions, and use terminology appropriate to that task.

Where the ***DAA*** is being used for a specific purpose, such as role enablement, a custom ***DAA*** PAP could be further simplified. For example, for role enablement the ***DA obligations*** can be assumed to be for the subject-role attribute with the anyURI data-type in the access-subject category. A ***DAA*** PAP for role enablement might also be integrated with a system for describing and managing the available roles.

One use case for the ***DAA*** is to transform attributes between heterogeneous systems, and not necessarily for the purpose of making authorization decisions. This need might arise through mergers and acquisitions, partnering with other organizations or sourcing applications from different vendors. The ***DAA*** PAP might be largely invisible within a management console leveraging data dictionaries to integrate disparate applications. At a high level the user makes linkages between items in the data dictionaries and specifies transformation rules for attribute values, and the user interface takes care of representing that intent as ***DA obligations***, ***DA policies*** and ***DA policy sets***.

Just as ***DAA*** users don't need to use the full expressive power of XACML, a pure ***DAA*** implementation doesn't need to be a complete XACML implementation. An implementation could use simplified representations of XACML rules, policies and policy sets internally and would only ever need to use the standard representation for interchange with other systems. If such implementations become widespread then there would be value in formally defining a specific framework for dynamic attribute generation, with artifacts that leave out the unused and unneeded parts of XACML and use nomenclature appropriate for generating attributes instead of authorization.

7 Conformance

An implementation claiming conformance with this specification MUST support the extended data flow model in Section 2 and the obligations defined in Section 3.

Appendix A. References

This appendix contains the normative and informative references that are used in this document. Normative references are specific (identified by date of publication and/or edition number or Version number) and Informative references are either specific or non-specific.

While any hyperlinks included in this appendix were valid at the time of publication, OASIS cannot guarantee their long term validity.

A.1 Normative References

The following documents are referenced in such a way that some or all of their content constitutes requirements of this document.

[RFC2119]

Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

[RFC8174]

Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <http://www.rfc-editor.org/info/rfc8174>.

[**XACML-v3.0-Errata01-complete**]

eXtensible Access Control Markup Language (XACML) Version 3.0 Plus Errata 01. Edited by Erik Rissanen. 12 July 2017. OASIS Standard incorporating Approved Errata. http://docs.oasis-open.org/xacml/3.0/errata01/os/xacml-3.0-core-spec-errata01-os-complete.html. Latest version: http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-en.html.

[xacml-json-v1.1]

JSON Profile of XACML 3.0 Version 1.1. Edited by David Brossard and Steven Legg. 20 June 2019. OASIS Standard. https://docs.oasis-open.org/xacml/xacml-json-http/v1.1/os/xacml-json-http-v1.1-os.html. Latest version: https://docs.oasis-open.org/xacml/xacml-json-http/v1.1/xacml-json-http-v1.1.html.

[xacml-3.0-multiple-v1.0]

XACML v3.0 Multiple Decision Profile Version 1.0. Edited by Erik Rissanen. 18 May 2014. OASIS Committee Specification 02. http://docs.oasis-open.org/xacml/3.0/multiple/v1.0/cs02/xacml-3.0-multiple-v1.0-cs02.html. Latest version: http://docs.oasis-open.org/xacml/3.0/multiple/v1.0/xacml-3.0-multiple-v1.0.html.

A.2 Informative References

The following referenced documents are not required for the application of this document but may assist the reader with regard to a particular subject area.

[XACML-3.0-RBAC]

XACML v3.0 Core and Hierarchical Role Based Access Control (RBAC) Profile Version 1.0. Edited by Erik Rissanen. 23 October 2014. OASIS Committee Specification 02. http://docs.oasis-open.org/xacml/3.0/rbac/v1.0/cs02/xacml-3.0-rbac-v1.0-cs02.html. Latest version: http://docs.oasis-open.org/xacml/3.0/rbac/v1.0/xacml-3.0-rbac-v1.0.html.

[RFC3552]

Rescorla, E. and B. Korver, "Guidelines for Writing RFC Text on Security Considerations", BCP 72, RFC 3552, DOI 10.17487/RFC3552, July 2003, <https://www.rfc-editor.org/info/rfc3552>.

[xacml-3.0-nested-ent-v1.0]

XACML v3.0 Related and Nested Entities Profile Version 1.0. Edited by Steven Legg. 16 February 2021. OASIS Committee Specification 02. https://docs.oasis-open.org/xacml/xacml-3.0-related-entities/v1.0/cs02/xacml-3.0-related-entities-v1.0-cs02.html. Latest stage: https://docs.oasis-open.org/xacml/xacml-3.0-related-entities/v1.0/xacml-3.0-related-entities-v1.0.html.

Appendix B. Security and Privacy Considerations

[Informative]

DA policies and ***DA policy sets*** are subject to the same security concerns as conventional policies and policy sets. The reader should refer to the security and privacy considerations in the XACML core specification **XACML-v3.0-Errata01-complete**.

By injecting dynamic attributes into the ***final request***, ***DA policies*** and ***DA policy sets*** are able to alter the outcome of the original request, which could be used by an attacker to obtain unauthorized access. For this reason, the ***DA policies*** and ***DA policy sets*** must be at least as well protected from unauthorized modification as the conventional policies and policy sets.

Although largely internal to the context handler, the value of a dynamic attribute can be discovered by external parties. The Security Considerations of the Related and Nested Entities Profile **[xacml-3.0-nested-ent-v1.0]**

[xacml-3.0-nested-ent-v1.0]

discusses various ways in which the value of an attribute can be discovered by unauthorized users and possible ways to prevent that discovery.

Appendix C. Acknowledgments

C.1 Special Thanks

Substantial contributions to this document from the following individuals are gratefully acknowledged:

Steven Legg, ViewDS Identity Solutions
Hal Lockhart, Individual Member
Bill Parducci, Individual Member

C.2 Participants

The following individuals were members of this Technical Committee during the creation of this document and their contributions are gratefully acknowledged:

Steven Legg, ViewDS Identity Solutions
Hal Lockhart, Individual Member
Bill Parducci, Individual Member

Appendix D. Revision History

Revisions made since the initial stage of this numbered Version of this document may be tracked here.

Revision	Date	Editor	Changes Made
WD 01	29 March 2021	Steven Legg	Initial draft.
WD 02	30 April 2021	Steven Legg	Added role enablement and data transformation examples.
WD 03	23 July 2021	Steven Legg	Expanded the Architectural Considerations section and subsumed the Considerations for a Future Version section. Added the security and privacy considerations.
WD 04	29 September 2021	Steven Legg	Fixed syntax errors in the XML examples. Changed the final processing so that dynamic attributes override attributes in the initial request instead of being merged. Added an informative section describing how to achieve effects different from the default final processing. Added a normative reference to the JSON profile.
WD 05	5 November 2021	Steven Legg	Fixed syntax errors in the enable-roles example policy set. Fixed the combining algorithm in the generic-attribute example policy set. Added missing mandatory XML attributes to the example <Request> elements. Gave the last two policy examples distinct identifiers.

Appendix E. Notices

All capitalized terms in the following text have the meanings assigned to them in the OASIS Intellectual Property Rights Policy (the "OASIS IPR Policy"). The full Policy may be found at the OASIS website: [https://www.oasis-open.org/policies-guidelines/ipr/].

This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published, and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this section are included on all such copies and derivative works. However, this document itself may not be modified in any way, including by removing the copyright notice or references to OASIS, except as needed for the purpose of developing any document or deliverable produced by an OASIS Technical Committee (in which case the rules applicable to copyrights, as set forth in the OASIS IPR Policy, must be followed) or as required to translate it into languages other than English.

The limited permissions granted above are perpetual and will not be revoked by OASIS or its successors or assigns.

This document and the information contained herein is provided on an "AS IS" basis and OASIS DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY OWNERSHIP RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. OASIS AND ITS MEMBERS WILL NOT BE LIABLE FOR ANY DIRECT, INDIRECT, SPECIAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF ANY USE OF THIS DOCUMENT OR ANY PART THEREOF.

As stated in the OASIS IPR Policy, the following three paragraphs in brackets apply to OASIS Standards Final Deliverable documents (Committee Specifications, OASIS Standards, or Approved Errata).

[OASIS requests that any OASIS Party or any other party that believes it has patent claims that would necessarily be infringed by implementations of this OASIS Standards Final Deliverable, to notify OASIS TC Administrator and provide an indication of its willingness to grant patent licenses to such patent claims in a manner consistent with the IPR Mode of the OASIS Technical Committee that produced this deliverable.]

[OASIS invites any party to contact the OASIS TC Administrator if it is aware of a claim of ownership of any patent claims that would necessarily be infringed by implementations of this OASIS Standards Final Deliverable by a patent holder that is not willing to provide a license to such patent claims in a manner consistent with the IPR Mode of the OASIS Technical Committee that produced this OASIS Standards Final Deliverable. OASIS may include such claims on its website, but disclaims any obligation to do so.]

[OASIS takes no position regarding the validity or scope of any intellectual property or other rights that might be claimed to pertain to the implementation or use of the technology described in this OASIS Standards Final Deliverable or the extent to which any license under such rights might or might not be available; neither does it represent that it has made any effort to identify any such rights. Information on OASIS' procedures with respect to rights in any document or deliverable produced by an OASIS Technical Committee can be found on the OASIS website. Copies of claims of rights made available for publication and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this OASIS Standards Final Deliverable, can be obtained from the OASIS TC Administrator. OASIS makes no representation that any information or list of intellectual property rights will at any time be complete, or that any claims in such list are, in fact, Essential Claims.]

The name "OASIS" is a trademark of OASIS, the owner and developer of this document, and should be used only to refer to the organization and its official outputs. OASIS welcomes reference to, and implementation and use of, documents, while reserving the right to enforce its marks against misleading uses. Please see https://www.oasis-open.org/policies-guidelines/trademark/ for above guidance.