JSON Representation of ACAL Version 1.0 (JACAL)

Committee Specification Draft 01

18 February 2026

This version

- https://docs.oasis-open.org/xacml/acal/jacal/core/v1.0/csd01/acal-core-json-v1.0-csd01.html (Authoritative) -
- https://docs.oasis-open.org/xacml/acal/jacal/core/v1.0/csd01/acal-core-json-v1.0-csd01.pdf \
- https://docs.oasis-open.org/xacml/acal/acal/profiles/xpath/v1.0/csd01/acal-xpath-v1.0-csd01.md

Previous version

N/A

Latest version

- https://docs.oasis-open.org/xacml/acal/jacal/core/v1.0/csd01/acal-core-json-v1.0-csd01.html (Authoritative) \
- https://docs.oasis-open.org/xacml/acal/jacal/core/v1.0/csd01/acal-core-json-v1.0-csd01.pdf \
- https://docs.oasis-open.org/xacml/acal/acal/profiles/xpath/v1.0/csd01/acal-xpath-v1.0-csd01.md

Technical Committee:

OASIS eXtensible Access Control Markup Language (XACML) TC

Chairs:

- Bill Parducci (bill@parducci.net), Individual

Secretaries

- Bill Parducci (bill@parducci.net), Individual

Editors:

- Steven Legg (steven.legg@viewds.com), ViewDS Identity Solutions
- Cyril Dangerville (cyril.dangerville@thalesgroup.com), THALES

Additional artifacts:

This document is one component of a Work Product that also includes:

- Core JSON schema: acal-core-json-v1.0-schema.json
- Short identifier set: acal-core-json-v1.0-identifiers.json

Abstract:

This specification defines Version 1.0 of the JSON Representation Profile of the ACAL (JACAL) Version 1.0.

Citation format:

When referencing this specification the following citation format should be used:

[**JACAL-Core-1.0**] *ACAL v1.0 JSON Representation Profile (JACAL) Version 1.0.* Edited by Steven Legg and Cyril Dangerville. 18 February 2026. OASIS Committee Specification Draft 01. https://docs.oasis-open.org/xacml/acal/jacal/core/v1.0/csd01/acal-core-json-v1.0-csd01.html . Latest stage: https://docs.oasis-open.org/xacml/acal/jacal/core/v1.0/csd01/acal-core-json-v1.0-csd01.html.

Related work:

This document is related to:

- *Attribute-Centric Authorization Language (ACAL) Version 1.0*. Edited by Steven Legg and Cyril Dangerville. 18 February 2026. OASIS Committee Specification Draft 01.

License, Document Status, and Notices

Table of Contents

---

1 Scope

This specification defines the JSON syntax of the [ACAL-Core-1.0] model and any JSON-specific syntax, semantics and processing instructions that are not already specified by [ACAL-Core-1.0]. For more information on the scope, please refer to [ACAL-Core-1.0].

---

2 Definitions and Acronyms

2.1 Definitions

2.1.1 Terms Defined Elsewhere

None.

2.1.2 Terms Defined in this Document

None.

2.1.3 Related terms

None.

2.2 Abbreviations and Acronyms

This document uses the following abbreviations and acronyms:

JACAL
: JSON representation/syntax of ACAL as specified in this document.

JSLT
: JSON Query and Transformation Language [JSLT]

JSON
: JavaScript Object Notation [RFC8259]

JSONPath
: XPath equivalent for JSON as defined in [RFC9535]

---

3 Document Conventions

3.1 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] [RFC8174] when, and only when, they appear in all capitals, as shown here.

3.2 Typographical Conventions

This specification contains schema conforming to JSON Schema and normative text to describe the syntax and semantics of JSON-encoded ACAL objects.

```
Listings of JSON schema and code listings appear like this.
```

This specification uses the following typographical conventions in text: `JSONPropertyName`, `JSONDataType`, `OtherCode`. Terms in ***bold-face italic*** are intended to have the meaning defined in Section 2.

3.3 Schema organization and identifier(s)

The JACAL syntax is defined in a JSON Schema associated with the following identifier:

`urn:oasis:names:tc:jacal:1.0:core:schema`

---

4 Introduction (non-normative)

4.1 Requirements

The JSON representation (JACAL) should be as aligned as possible with [ACAL-Core-1.0].

4.2 Abstraction Layer

In the case where the native request/response format is specified in JSON Schema (e.g. an OpenID-Connect-conformant PEP), the transformation between the native format and the ACAL context may be specified in the form of a JSON Query And Transformation Language expression [JSLT].

Similarly, in the case where the resource to which access is requested is a JSON document, the resource itself may be included in, or referenced by, the request context. Then, through the use of JSONPath expressions [RFC9535] in the policy, values in the resource may be included in the policy evaluation. The use of JSONPath expressions is not specified here but in the JSONPath Profile of ACAL.

4.3 Example Short Identifier Set

A set of Short Identifiers with the Id `urn:oasis:names:tc:acal:1.0:core:identifiers` is defined by ACAL in the JSON Representation according to this specification, for the various identifiers assigned by ACAL, and provided attached to this profile. However, a deployment will usually have need for additional identifiers, especially for locally-defined attributes, so it is usually desirable to define a set of additional short identifiers to use in the deployment, that may import the first set.

The following short-identifier set defines an JSON representation of short identifiers for the additional attributes in this example and also imports the standardized set.

```
{
  "Id": "urn:oasis:names:tc:acal:1.0:example:identifiers",
  "ShortIdSetReference": ["urn:oasis:names:tc:acal:1.0:core:identifiers"],
  "ShortId": [
    { "Name": "patient-number", "Value": "urn:oasis:names:tc:acal:1.0:example:attribute:patient-number" },
    { "Name": "collection", "Value": "urn:oasis:names:tc:acal:1.0:example:attribute:collection" }
  ]
}
```

Short Identifiers can reuse other Short Identifiers in their values, typically as prefix, for example, given the following short identifiers:

```
{
  "ShortId": [
    {"Name": "xs", "Value": "urn:oasis:names:tc:acal:1.0:data-type:"},
    {"Name": "string", "Value": "{xs}string"}
  ]
}
```

the following `IdentifierType` values are all equivalent and evaluate to the URI of the string data type:

```
"string"
"{string}"
"{xs}string"
```

4.4 Changes From the Previous Version

The list of changes from the previous version and any revision history can be found in Appendix 2.

---

5 Syntax (normative, with the exception of the schema fragments)

The next sections describe the rules that SHALL be applied for mapping the [ACAL-Core-1.0] agnostic model (UML-based) to JSON schema Draft 2020-12 definitions for this JSON representation (JACAL). These rules have been applied to produce JACAL's core JSON schema in Annex D (also in the Core JSON schema file accompanying this document) from [ACAL-Core-1.0] core model.

We consider `PolicyType`, `BundleType`, `RequestType` and `ResponseType` as the root JSON objects to be used by JACAL users, therefore the final JACAL core schema has the following structure:

```
{
    "$id": "urn:oasis:names:tc:jacal:1.0:core:schema",
    "$schema": "https://json-schema.org/draft/2020-12/schema",
    "title": "JSON schema of ACAL Version 1.0 (JACAL)",
    "anyOf": [
        {
            "$ref": "#/$defs/PolicyType"
        },
    {
            "$ref": "#/$defs/BundleType"
        },
        {
            "$ref": "#/$defs/RequestType"
        },
        {
            "$ref": "#/$defs/ResponseType"
        }
    ],
    "$defs": {
    <subschema 1>,
    <subschema 2>,
    ...
    <subschema N>
  }
}
```

where `<subschema 1>`,`<subschema 2>`, etc. are the reusable JSON subschemas created by applying the ACAL model mapping rules described in sections 5.1 and 5.2. For the rest of the document, the JSON object containing these subschemas is simply referred to as *the `$defs` object*.

5.1 Mapping ACAL primitive types

For each primitive type (stereotyped `<<primitive>>` or `<<enumeration>>`) in [ACAL-Core-1.0] model, apply the mapping rules in the next subsections.

5.1.1 Primitive types mapped to native JSON schema definitions

The ACAL primitive types in the following table have a direct JSON equivalent in the JSON schema Draft 2020-12 specification, provided in the second column, and that is used for the JSON representation:

Table 1: Mapping ACAL primitive types to JSON schema

ACAL (UML)	JSON schema (Draft 2020-12)
String	`{"type": "string"}`
Boolean	`{"type": "boolean"}`
Double	`{"type": "number"}`
Integer	`{"type": "integer"}`
NonNegativeInteger	`{"type": "integer", "minimum": 0}`

ACAL (UML)	JSON schema (Draft 2020-12)
URI	`{"type": "string", "format": "uri-reference"}`

Contrary to the above subschemas which are not added as reusable schemas to the *$defs object*, the ACAL `Name` type's corresponding subschema is added to the *`$defs` object*:

```
{
  "$defs": {
    "Name": {"type": "string", "pattern": "^[_:A-Za-z][-._:A-Za-z0-9]*$"}
    ...
  }
}
```

5.1.2 Restricted String types (UML stereotype <<restrictedString>>)

Each ACAL primitive type `FooType` with stereotype `<<restrictedString>>`, i.e. with a given `pattern` property set to a regular expression , is mapped to a subschema in the *`$defs` object* as follows:

```
{
  "$defs": {
    ...
    "FooType": { "type": "string", "pattern": "<REGEX>" }
  }
}
```

`FooType` may also have a (optional) `minLength` property set to a (strictly) positive integer `N`, in which case the subschema becomes:

```
{
  "$defs": {
    ...
    "FooType": { "type": "string", "pattern": "<REGEX>", "minLength": <N> }
  }
}
```

For example, ACAL `VersionType` translates to the following subschema definition (backslashes must be escaped in JSON):

```
{
  "$defs": {
    ...
    "VersionType": { "type": "string", "pattern": "^(0|[1-9]\\d*)(\\.(0|[1-9]\\d*)){0,3}$" }
  }
}
```

5.1.3 Enum types (UML stereotype <<enumeration>>)

Each ACAL enumerated type `FooType` (stereotyped `<<enumeration>>`) with enum values *V1, V2, ... Vn* is mapped to the following subschema:

```
{
  "$defs": {
    ...
    "FooType": { "enum": ["V1", "V2", ..., "Vn"] }
    ...
  }
}
```

For example, ACAL `DecisionType` translates to the following subschema:

```
{
  "$defs": {
    ...
    "DecisionType": { "enum": ["Permit", "Deny", "Indeterminate", "NotApplicable"] }
    ...
  }
}
```

5.2 Mapping complex ACAL types (UML stereotype <<dataType>>)

For each complex type (stereotyped `<<dataType>>`) in [ACAL-Core-1.0] model, apply the mapping rules in the next subsections.

5.2.1 AnyType mapping rule

The ACAL `AnyType` used in `ContentType` objects is mapped to the following subschema:

```
{"type": ["string", "object"]}
```

The `object` type is used for JSON object (which can be used to wrap a JSON array as well), and the `string` type for non-JSON structured data, e.g. XML, possibly escaped or encoded to fit in a JSON string. See the Content Types section for examples.

WARNING: for safety/security reasons, in production, ACAL implementers should add further restrictions to this JSON schema and/or enforce security measures in the JSON processor to mitigate possible security issues that may occur when allowing any JSON object as input.

5.2.2 ValueType mapping rules

The `ValueType` and subtypes from [ACAL-Core-1.0] section 7.23 are mapped to JSON as described in the next subsections.

5.2.2.1 Primitive value mappings

1. A `LiteralBooleanType` object is represented as a JSON boolean. The ACAL data-type is implicitly set to `urn:oasis:names:tc:acal:1.0:data-type:boolean`.
2. A `LiteralIntegerType` object is represented as a JSON integer as defined by JsonSchemaValidation section 6.1.1 (JSON number with a zero fractional part). The ACAL data-type is implicitly set to `urn:oasis:names:tc:acal:1.0:data-type:integer`.
3. A `LiteralDoubleType` object is represented as a JSON number with a non-zero fractional part. The ACAL data-type is implicitly set to `urn:oasis:names:tc:acal:1.0:data-type:double`.
4. A `LiteralStringType` object is represented as a JSON string without any JSON property named `DataType`. If a `DataType` property is present at an upper level, i.e. in the parent or an ancestor object (e.g. `AttributeType` object), its value MUST be `urn:oasis:names:tc:acal:1.0:data-type:string`. Else the ACAL data-type is implicitly set to `urn:oasis:names:tc:acal:1.0:data-type:string`.
5. A `LiteralRestrictedStringType` object, which may be used for any primitive type with a lexical representation, is represented in either of two forms:
   1. If the `DataType` property is already present at an upper level, i.e. in the parent or an ancestor object (e.g. an `AttributeType` object), then this object may be represented simply as a JSON string. The ACAL data-type is inferred from the aforementioned `DataType` property.
   2. Else it is wrapped in a JSON object made of two string properties `DataType` and `$` holding the actual value:
      `{"DataType": "<LiteralRestrictedStringType object's DataType>", "Value": "<LiteralRestrictedStringType object's Valu`

If no support for structured data-types is needed, the following JSON subschema MAY be used for `ValueType` objects in general and added to the *`$defs` object* of the JACAL schema:

```
"ValueType":
{
  "anyOf": [
    {
      "$comment": "Case when the DataType property is unnecessary because implicit or already defined by the parent object",
      "type": ["boolean", "integer", "number", "string"]
    },
    {
      "$comment": "Case when the DataType property is required (not the case for boolean / integer / number which have a fixed
      "type": "object",
      "properties": {
        "DataType": {"$ref": "#/$defs/IdentifierType"},
        "Value": {"type": "string"}
      },
      "required": ["DataType", "Value"],
      "unevaluatedProperties": false
    }
  ]
}
```

For supporting structured data-types (as schema extensions), a more generic subschema is provided in the next section.

5.2.2.2 Structured value mappings `StructuredValueType` objects, which are used for structured values, are represented as JSON objects. Since `StructuredValueType` is an abstract type, it is the responsibility of

ACAL profiles (e.g. XPath profile) or ACAL users / implementers to define concrete subtypes as needed, with their own ACAL data-type identifier; and also to define their respective **JSON schema** if they need to have a JSON representation. For a given concrete subtype of `StructuredValueType` named `FooStructValueType`, a `FooStructValueType` object is represented in JSON in one of two forms of JSON objects (with a `DataType` property or not):

1. If the `DataType` property is already present at an upper level, i.e. in the parent or an ancestor object (e.g. an `AttributeType` object), then this object may be represented as a JSON object compliant with `FooStructValueType`'s JSON schema. The ACAL data-type is inferred from the aforementioned `DataType` property.
2. Else a string property `DataType` is added to the FooStructValueType's JSON object:
   ```
   {
    "DataType": "<FooStructValueType's DataType identifier>",
    <Properties according to FooStructValueType's JSON schema>
   }
   ```

The subtype `FooStructValueType` of `StructuredValueType` SHALL NOT (re)define any `DataType` property in its JSON representation, as the property is already defined in the core schema.

Therefor, the `ValueType`'s JSON schema SHALL be enhanced as follows in order to support structured data-types:

```
"ValueType": {
    "anyOf": [
      {
        "$comment": "Case when the DataType property is unnecessary (implicit or already defined by parent) and the value is pri
        "type": ["boolean", "integer", "number", "string"]
      },
      {
        "$comment": "Same case as above but for structured values",
        "$ref": "#/$defs/StructuredValueTypeTree"
      },
      {
        "$comment": "Case when DataType property is required (not the case for boolean / integer / number which have a fixed imp
        "allOf": [
          {
            "type": "object",
            "properties": {
              "DataType": { "$ref": "#/$defs/IdentifierType" }
            },
            "required": ["DataType"]
          },
          {
            "anyOf": [
              {
                "properties": {
                  "Value": {
                    "$comment": "For DataTypes represented as string (besides the standard string type itself).",
                    "type": "string"
                  }
                },
                "required": ["Value"]
              },
              { "$ref": "#/$defs/StructuredValueTypeTree" }
            ]
          }
        ],
        "unevaluatedProperties": false
      }
    ]
  },
  "StructuredValueTypeTree": {
    "$dynamicRef": "#StructuredValueTypeExtensions"
  },
  "StructuredValueTypeTreeEmpty": {
    "$dynamicAnchor": "StructuredValueTypeExtensions",
    "$comment": "No StructuredValueType extension by default. Create your own implementation-specific schema that overrides thi
    "not": true
  }
```

This enables ACAL implementers to extend `StructuredValueType` with new concrete subtypes of their own or from standard JACAL Profiles, by overriding the `$dynamicAnchor` (redeclare with the same name) in a implementation-specific JSON schema, as described in section 5.4. That section gives an example where `StructuredValueType` is extended with the `XPathExpressionValueType` (`urn:oasis:names:tc:acal:1.0:data-type:xpathExpression` data-type) from the XPath Profile.

5.2.3 Default mapping rules for complex ACAL types (other than ValueType)

For each complex ACAL type `FooType` that does not fall under any of the previous cases (section 5.2.1 and 5.2.2), apply the mappings rules defined in this section.

Definitions:

- The term *concrete type* is used as synonym for *non-abstract type*.
- An ACAL type is said *empty* iff it is an abstract type with no property.
- An ACAL type is said *final* iff it is a concrete type with no subtype.
- For a given non-final ACAL type *FooType*, the name *FooTypeTree* - as in *FooType*'s inheritance *Tree* - is used for the schema composed of all its subtypes' subschemas, and, if *FooType* is a (non-empty) concrete type, the subschema of *FooType* itself. Recursively, a similar *FooSubTypeTree* subschema will be included via reference (using `$ref` keyword) for each non-final subtype `FooSubType` of `FooType`.
- Let `FooSub1FinalType`, `FooSub2FinalType`, etc. be referred to as the final subtypes of `FooType` if there is any.
- Let `FooSub1NonFinalType`, `FooSub2NonFinalType`, etc. be referred to as the non-final subtypes of `FooType` if there is any.
- Let *pn* be the n-th property defined in *FooType* class model specifically (not already defined in a supertype of *FooType* if any).
- Let <*si*> be the JSON subschema obtained by applying Property mapping rules of section 5.2.4 to the property *pn*.
- Let <*LR*> be the list of all required properties *pn*, i.e. such that the lower bound of *pn*'s multiplicity is 1.

Mapping rules:

1. If `FooType` is abstract (italicized title in the UML diagram), then:
   - 1.1. If `FooType` is not *empty*, then:
     - 1.1.1. If `FooType` inherits from a non-empty type `BarType` (DataType), then add the following JSON subschema to the ***$defs** object* (the `BarType` subschema has been or will be created/added when applying the very same rules of this section to `BarType`, the order of declaration does not matter):
       ```
       "FooType": {
          "$comment": "Used by subtypes of FooType to combine FooType properties (via $ref) with their own properties",
          "allOf": [
            { "$ref": "#$defs/BarType" },
            {
              "type": "object",
              "properties": {
                "p1": <s1>,
                "p2": <s2>,
                ...
              },
              "required": [ <LR> ]
            }
          ]
       }
       ```
     - 1.1.2. Else (`FooType` is abstract and does not inherit a non-empty type) add the following JSON subschema instead:
       ```
       "FooType": {
          "$comment": "Used by subtypes of FooType to combine FooType properties (via $ref) with their own properties",
          "type": "object",
          "properties": {
            "p1": <s1>,
            "p2": <s2>,
            ...
          },
          "required": [ <LR> ]
       }
       ```

   The case when ***FooType*** *is empty and inherits a non-empty (abstract) class should not occur in the ACAL model, and therefore it is ignored here.*

   - 1.2. *Iff* there is at least one ACAL type with a property of type `FooType` or `FooSuperType` where `FooSuperType` may be any supertype of `FooType`, then:
     - 1.2.1. If `FooType` does not have any subtype (defined in ACAL core model), then add also the following JSON subschemas to the ***$defs** object*:
       ```
       "FooTypeTree": {
          "$dynamicRef": "#FooTypeExtensions"
       ```

```
},
"FooTypeTreeEmpty": {
  "$dynamicAnchor": "FooTypeExtensions",
  "$comment": "No FooType extension by default in the core schema. But one may define an implementation-specific
  "not": true
}
```

This enables ACAL implementers to extend `FooType` with concrete subtypes of their own or from standard JACAL Profiles, by overriding the `$dynamicAnchor` with the same name in a new (implementation-specific) JSON schema. See section 5.4 for more information.

- 1.2.2. Else (`FooType` has one or more subtypes in ACAL core model), then add also the following JSON subschema to the *`$defs` object*:

```
"FooTypeTree": {
  "$comment": "FooType's subtypes",
  "anyOf": [
    {
      "type": "object",
      "properties": {
        "$comment": "FooSub1FinalType is final (no 'FooSub1FinalTypeTree' subschema)",
        "FooSub1Final": {
          "$ref": "#/$defs/FooSub1FinalType"
        }
      },
      "required": [
          "FooSub1Final"
      ],
      "additionalProperties": false
    },
    {
      "type": "object",
      "properties": {
        "$comment": "FooSub2FinalType is final",
        "FooSub2Final": {
          "$ref": "#/$defs/FooSub2FinalType"
        }
      },
      "required": [
        "FooSub2Final"
      ],
      "additionalProperties": false
    },
    ... other final subtypes' subschemas ...
    {
      "$comment": "FooSub1NonFinalType is non-final",
      "$ref": "#/$defs/FooSub1NonFinalTypeTree"
    },
    {
      "$comment": "FooSub2NonFinalType is non-final",
      "$ref": "#/$defs/FooSub2NonFinalTypeTree"
    }
    ... other non-final subtypes' subschemas...
  ]
}
```

(Else it is only used as a base type for other ACAL types, in which case the `FooType` subschema created by the next rules is enough.)

2. Else (`FooType` is a *concrete* type):

- 2.1. If `FooType` is *final* (no subtype), then:

  - 2.1.1. If `FooType` inherits from a non-empty type `BarType` (DataType), then add the following subschema to the *`$defs` object* (`BarType` has been or will be created when applying the very same rules of this section to `BarType`):

```
"FooType": {
   "allOf": [
     { "$ref": "#$defs/BarType" },
     {
       "type": "object",
       "properties": {
         "p1": <s1>,
         "p2": <s2>,
         ...
       },
       "required": [ <LR> ]
     }
```

```
  ],
  "unevaluatedProperties": false
}
```

- 2.1.2. Else (`FooType` is concrete, final and does not inherit from a non-empty type) add the following JSON subschema to the ***$defs** object*:

```
"FooType": {
   "type": "object",
   "properties": {
     "p1": <s1>,
     "p2": <s2>,
     ...
   },
   "required": [ <LR> ],
   "unevaluatedProperties": false
 }
```

- 2.2. Else (`FooType` is concrete and non-final):
  - 2.2.1. If `FooType` inherits from a non-empty type `BarType`, same mapping as rule 1.1.1.
  - 2.2.2. Else (`FooType` is concrete, non-final and does not inherit a non-empty class) same mapping as rule 1.1.2.
  - 2.2.3. *Iff* there is at least one ACAL type with a property of type `FooType`, then add the following JSON subschema to the ***$defs** object* (similar to rule 1.2 except the subschema of `FooType` itself is included because it is a concrete type that may be used for a JSON object property):

```
"FooTypeTree": {
  "anyOf": [
    {
      "type": "object",
      "properties":
      {
        "Foo": {
          "$ref": "#$defs/FooType",
          "additionalProperties": false
        }
      },
      "required": ["Foo"],
      "additionalProperties": false
    },
    {
      "type": "object",
      "properties": {
        "$comment": "FooSub1FinalType is final (no 'FooSub1FinalTypeTree' subschema)",
        "FooSub1Final": {
          "$ref": "#/$defs/FooSub1FinalType"
        }
      },
      "required": [
        "FooSub1Final"
      ],
      "additionalProperties": false
    },
    {
      "type": "object",
      "properties": {
        "$comment": "FooSub2FinalType is final",
        "FooSub2Final": {
          "$ref": "#/$defs/FooSub2FinalType"
        }
      },
      "required": [
        "FooSub2Final"
      ],
      "additionalProperties": false
    },
    ... other final subtypes' subschemas ...
    {
      "$comment": "FooSub1NonFinalType is non-final",
      "$ref": "#/$defs/FooSub1NonFinalTypeTree"
    },
    {
      "$comment": "FooSub2NonFinalType is non-final",
      "$ref": "#/$defs/FooSub2NonFinalTypeTree"
    }
    ... other non-final subtypes' subschemas...
```

```
        ]
      }
```

5.2.4 Property mapping rules

For each of an ACAL Datatype's property *Prop* with value type *PropType*, the corresponding JSON subschema is obtained as follows:

- Map *PropType* to a JSON subschema according to the mapping rules of previous sections 5.1, 5.2.1, 5.2.2 and 5.2.3.
- 1. If *Prop* is single-valued, the final JSON subschema of the property value is the obtained previously, unless it has a defined default primitive value, represented in JSON as `<DEFAULT>`, in which case the final schema is:

```
{ <PropTypeSchema_without_opening_and_closing_braces>, "default": <DEFAULT> }
```

- 2. Else (*Prop* is multivalued):
  - 2.1. If the property has a **Value type uniqueness constraint** as defined in [ACAL-Core-1.0] section 7.1.1.1.1.2 (`self->isUnique(oclType())`), then map to the following subschema:

    ```
    {
      "type": "object",
      "properties": {
        "Item1": <Item1TypeSchema>,
        "Item2": <Item2TypeSchema>
        ...
      },
      "unevaluatedProperties": false
    }
    ```

    where *Item1Type*, *Item2Type*, etc. are all possible (and distinct) concrete subtypes of *PropType* (*ItemXTypeSchema* is the JSON subschema corresponding to the item's type *ItemXType*).
  - 2.2. Else (no *Value type uniqueness constraint*), map to an array type as follows:

    ```
    {
        "type": "array",
        "items": <PropTypeSchema>,
        "minItems": <min>,
        "uniqueItems": <is_unique>
    }
    ```

    where:

    - `<min>` is set to the lower bound of *Prop*'s multiplicity unless the lower bound is zero, in which case `<min>` is set 1 regardless, since the lower bound zero is already achieved by making the JSON property optional;
    - `<is_unique>` is set to `true` if and only if the `unique` constraint is specified on the ACAL property, else `false`.

    The standard `uniqueItems` keyword does not allow to enforce uniqueness of array items based on a specific key when such items are JSON objects, in the current latest JSON schema draft (version 2020-12).

    Therefore, the mapping of **property-based uniqueness constraints** - defined in [ACAL-Core-1.0] section 7.1.1.1.1.2 - on properties of complex/structured type (mapped to JSON object) is left implementation-defined by this specification, since there is no standard mechanism in the current latest JSON schema standard to enforce the such constraints. However, as a general guidance, implementations MAY use the third-party ArrayExt extension vocabulary and more particulary the `uniqueKeys` keyword (instead of `uniqueItems`) to implement this feature.

5.2.5 Mapping ACAL object-level constraints (OCL)

ACAL object-level constraints defined in [ACAL-Core-1.0] section 7.1.1.1.2 may be translated into JSON subschema(s) to be added the corresponding JSON schema definition of the ACAL Datatype, according to the table below:

Table 2: ACAL/UML constraints mapped to JSON schema

ACAL - UML constraint (OCL)	JSON schema equivalent
`X or Y` *(X, Y can be any of the predicates below)*	`"if": {"not": <X_subschema>}, "then": <Y_subschema>`
`prop <> null` *(prop is single-valued)*	`{"required": ["prop"]}`
`prop = null` *(prop is single-valued)*	`{"not": {"required": ["prop"]}}` *(any resulting* `{"not": {"not": <`
`prop->notEmpty()` *(prop is multivalued)*	`{"required": ["prop"]}` *(minItems is already set by rule 2.2.2 (previo*

5.3 Content Types and Body representations (optional)

Although this specification defines an JSON representation, both JSON and non-JSON data may be represented in a `Content` object (corresponding to an ACAL `ContentType` object). This specification defines the following `Content` types in order to support ACAL Profiles with AttributeSelector and/or DataType extensions based on such Content (e.g. XPath and JSONPath Profiles):

- JSON object *(note that a JSON array can be wrapped in a JSON object if there is a need to support JSON arrays)*:
  - `MediaType` property SHALL be set to `application/json` (default value);
  - `Encoding` attribute unused;
  - `Body` property is set to the JSON object itself.
- XML document:
  - `MediaType` attribute SHALL be set to `application/xml`;
  - `Encoding` attribute is either unused/undefined or set to `base64`.
  - `Body` property is set to a JSON string containing the XML document in one of the following forms:
    1) **Escaped:** if `Encoding` is undefined, the XML is escaped to be a valid JSON string using escaping rules described in section 7 of [RFC8259], i.e. in particular the double quote (`"`), backslash (`\`) and control characters are escaped with a backslash `\` (the new line escaped as `\n`, the carriage return as `\r`, and the horizontal tab as `\t`). For example:
    ```
    {
     "MediaType": "application/xml",
     "Body": "<?xml version=\"1.0\"?><catalog><book id=\"bk101\"><author>Gambardella, Matthew</author><title>XML Develope
    }
    ```
    2) **Base64-encoded:** if `Encoding` is `base64`, the XML is Base64-encoded as per [BASE64]. For example:
    ```
    {
     "MediaType": "application/xml",
     "Encoding": "base64",
     "Body": "PD94bWwgdmVyc2lvbj0iMS4wIj8+DQo8Y2F0YWxvZz48Ym9vayBpZD0iYmsxMDEiPjxhdXRob3I+R2FtYmFyZGVsbGEsIE1hdHRoZXc8L2
    }
    ```

The implementation SHALL support a given content type in this list if and only if there is an ACAL Profile that makes it mandatory (refer to the Profile's specification for more information).

5.4 JACAL Extension Mechanism

If the implementation does not support any extension, you may use the JACAL core schema (provided with this specification) as is, and ignore this section.

Extending JACAL syntax means extending JACAL core JSON schema (obtained from the mapping rules in the previous section). As explained in the mapping rule 1.2.1 of section 5.2.3, JACAL core schema uses a `$dynamicRef` for any extensible ACAL type that may be extended by a separate JSON schema (overriding a matching `$dynamicAnchor`), depending on which extensions the ACAL implementation shall support.

To explain how to use this extension mechanism, we go through various concrete examples in the next sections.

5.4.1 Example using extensions from a single ACAL Profile

In this example, we consider an ACAL implementation that supports the `AttributeSelectorType` extension from the standard JSONPath Profile of ACAL. In this case, the implementation may use as root schema the following combining schema, which combines the core schema with the supported extension's schema:

```
{
    "$schema": "https://json-schema.org/draft/2020-12/schema",
    "$id": "urn:example:root:schema:with:jsonpath:profile",
    "$defs": {
        "EnabledAttributeSelectorExtensions": {
```

```
            "$dynamicAnchor": "AttributeSelectorTypeExtensions",
            "$ref": "urn:oasis:names:tc:jacal:1.0:jsonpath:schema#/$defs/JSONPathAttributeSelectorTypeTree"
        },
        "EnabledEntityAttributeSelectorExtensions": {
            "$dynamicAnchor": "EntityAttributeSelectorTypeExtensions",
            "$ref": "urn:oasis:names:tc:jacal:1.0:jsonpath:schema#/$defs/JSONPathEntityAttributeSelectorTypeTree"
        }
    },
    "$ref": "urn:oasis:names:tc:jacal:1.0:core:schema"
}
```

This schema refers to (and therefore depends on) the JSONPath Profile's JSON schema by its identifier `urn:oasis:names:tc:jacal:1.0:jsonpath:schema`, which is also provided by the XACML TC with the core schema.

5.4.2 Example using extensions from multiple ACAL Profiles

In this example, we consider an ACAL implementation that supports various extensions (RequestDefaultsType, PolicyDefaultsType, AttributeSelectorType, StructuredValueType) from the XPath Profile of ACAL, and the `AttributeSelectorType` extension from the JSONPath Profile of ACAL. In this case, the implementation should use as root schema the following combining schema, which combines the core schema with the supported extensions' schemas:

```
{
    "$schema": "https://json-schema.org/draft/2020-12/schema",
    "$id": "urn:example:implementation:specific:root:schema",
    "$defs": {
        "EnabledRequestDefaultsExtensions": {
            "$dynamicAnchor": "RequestDefaultsTypeExtensions",
            "$ref": "urn:oasis:names:tc:jacal:1.0:xpath:schema#/$defs/XPathRequestDefaultsTypeTree"
        },
        "EnabledPolicyDefaultsExtensions": {
            "$dynamicAnchor": "PolicyDefaultsTypeExtensions",
            "$ref": "urn:oasis:names:tc:jacal:1.0:xpath:schema#/$defs/XPathPolicyDefaultsTypeTree"
        },
        "EnabledAttributeSelectorExtensions": {
            "$dynamicAnchor": "AttributeSelectorTypeExtensions",
            "anyOf": [
                {
                    "$ref": "urn:oasis:names:tc:jacal:1.0:xpath:schema#/$defs/XPathAttributeSelectorTypeTree"
                },
                {
                    "$ref": "urn:oasis:names:tc:jacal:1.0:jsonpath:schema#/$defs/JSONPathAttributeSelectorTypeTree"
                }
            ]
        },
        "EnabledEntityAttributeSelectorExtensions": {
            "$dynamicAnchor": "EntityAttributeSelectorTypeExtensions",
            "anyOf": [
                {
                    "$ref": "urn:oasis:names:tc:jacal:1.0:xpath:schema#/$defs/XPathEntityAttributeSelectorTypeTree"
                },
                {
                    "$ref": "urn:oasis:names:tc:jacal:1.0:jsonpath:schema#/$defs/JSONPathEntityAttributeSelectorTypeTree"
                }
            ]
        },
        "EnabledStructuredValueTypeExtensions": {
            "$dynamicAnchor": "StructuredValueTypeExtensions",
            "$ref": "urn:oasis:names:tc:jacal:1.0:xpath:schema#/$defs/XPathExpressionValueType"
        }
    },
    "$ref": "urn:oasis:names:tc:jacal:1.0:core:schema"
}
```

This schema refers to (and therefore depends on) to both the JSONPath Profile's JSON schema by its identifier `urn:oasis:names:tc:jacal:1.0:jsonpath:schema` and XPath Profile's JSON schema by its identifier `urn:oasis:names:tc:jacal:1.0:xpath:schema`, which are also provided by the XACML TC with the core schema.

5.4.3 Example combining a custom extension with a standard profile-defined extension

In this example, we consider an ACAL implementation that supports two extensions of some type `FooType` defined in the core schema: one custom extension `CustomFooSubType` and another extension `SomeProfileFooSubType` defined in an existing JACAL Profile with a schema identified `urn:some:profile:schema`. In this case, the implementation should use as root schema the following combining schema, which combines the core schema with the supported extensions' schemas:

```
{
  "$schema": "https://json-schema.org/draft/2020-12/schema",
  "$id": "urn:my:implementation:specific:root:schema",
  "$defs": {
      "EnabledFooTypeExtensions": {
          "$dynamicAnchor": "FooTypeExtensions",
          "anyOf": [
              {
                "$ref": "#/$defs/CustomFooSubTypeTree"
              },
              {
                "$ref": "urn:some:profile:schema#/$defs/SomeProfileFooSubTypeTree"
              }
          ]
      },
      "CustomFooSubType": {...},
      "CustomFooSubTypeTree": {
        "type": "object",
        "required": [
          "CustomFooSub"
        ],
        "properties": {
          "CustomFooSub": {
            "$ref": "#/$defs/CustomFooSubType"
          }
        },
        "unevaluatedProperties": false
      }
      ...
  },
  "$ref": "urn:oasis:names:tc:jacal:1.0:core:schema"
}
```

---

6 Safety, Security and Privacy Considerations (non-normative)

This section identifies possible security and privacy compromise scenarios that should be considered when implementing this profile. The section is informative only. It is left to the implementer to decide whether these compromise scenarios are practical in their environment and to select appropriate safeguards.

6.1 Threat model

Refer to [ACAL-Core-1.0] section 11.1.

6.2 Safeguards

Refer to [ACAL-Core-1.0] section 11.2 for general considerations.

6.2.1 Policy confidentiality

Where the policy is represented in JSON Representation (JACAL) defined by this profile, the *RFC 7516 - JSON Web Encryption (JWE)* standard can be used to encrypt all or parts of a JSON document. This specification is recommended for use with JACAL.

6.2.2 Policy integrity

The selection of the appropriate mechanisms is left to the implementers. However, when ***policy*** is distributed between organizations to be acted on at a later time, or when the ***policy*** travels with the protected ***resource***, it would be useful to sign the ***policy***. In these cases and when the JSON representation of policies (according to this profile) is used, the *RFC 7515 - JSON Web Signature (JWS)* is recommended to be used with JACAL.

7 Conformance

7.1 Introduction

The JACAL specification addresses the following aspect of conformance:

The JACAL specification defines a number of functions, etc. that have somewhat special applications, therefore they are not required to be implemented in an implementation that claims to conform with the OASIS standard.

7.2 Conformance tables

This section lists those portions of the specification that MUST be included in an implementation of a *PDP* that claims to conform to JACAL 1.0.
Note: "M" means mandatory-to-implement. "O" means optional.

The implementation MUST follow Section 5 and Annex C where they apply to implemented items in the following tables.

7.2.1 Schema objects

The implementation MUST support those JSON schema objects that are marked `M`.

Object name	M/O
Apply	M
ApplicablePolicyReference	O
Attribute	M
AttributeAssignment	M
AttributeAssignmentExpression	M
AttributeDesignator	M
AttributeSelector	O
Bundle	O
Category	M
Condition	M
Content	O
Description	M
EntityAttributeDesignator	O
EntityAttributeSelector	O
Expression	M
ForAll	O
ForAny	O
Function	M
Map	O
MissingAttributeDetail	M
MultiRequests	O
Notice	M
NoticeExpression	M
Policy	M
PolicyDefaults	O
PolicyReference	M
PolicyIssuer	O
PolicyPatternMatchReference	O
Request	M
RequestAttribute	M
RequestDefaults	O
RequestEntity	M
RequestEntityReference	O
RequestReference	O
Response	M
Result	M

Object name	M/O
ResultEntity	M
Rule	M
Select	O
SharedVariableDefinition	O
SharedVariableReference	O
ShortId	M
ShortIdSet	M
ShortIdSetReference	M
Status	M
StatusCode	M
StatusDetail	O
StatusMessage	O
Target	M
Value	M
VariableDefinition	M
VariableReference	M

Annex A License, Document Status and Notices

(This annex forms an integral part of this Specification.)

A.1 Document 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 version" 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://groups.oasis-open.org/communities/tc-community-home2?CommunityKey=67afe552-0921-49b7-9a85-018dc7d3ef1d#technical.

TC members should send comments on this document 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/.

NOTE: 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.

A.2 License and Notices

Annex B References

(This annex forms an integral part of this Specification.)

This section 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. For specific references, only the cited version applies. For non-specific references, the latest version of the reference document (including any amendments) applies. While any hyperlinks included in this section were valid at the time of publication, OASIS cannot guarantee their long term validity.

B.1 Normative References

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

[ACAL-Core-1.0]

Attribute-Centric Authorization Language (ACAL) Version 1.0. Edited by Steven Legg and Cyril Dangerville. 18 February 2026. OASIS Committee Specification Draft 01.

[RFC8259]

RFC 8259, Tim Bray, *The JavaScript Object Notation (JSON) Data Interchange Format*, December 2017, https://www.rfc-editor.org/info/rfc8259 .

[JsonSchemaCore] A. Wright and al., *JSON Schema: A Media Type for Describing JSON Documents*, June 2022, https://json-schema.org/draft/2020-12/draft-bhutton-json-schema-01

[JsonSchemaValidation] A. Wright and al., *JSON Schema Validation: A Vocabulary for Structural Validation of JSON*, June 2022, json-schema.org/draft/2020-12/json-schema-validation

[CMF]

Martin J. Dürst et al, eds., Character Model for the World Wide Web 1.0: Fundamentals, W3C Recommendation 15 February 2005, https://www.w3.org/TR/2005/REC-charmod-20050215/

[DS]

D. Eastlake et al., XML-Signature Syntax and Processing, https://www.w3.org/TR/xmldsig-core/, World Wide Web Consortium.

[exc-c14n]

J. Boyer et al, eds., Exclusive XML Canonicalization, Version 1.0, W3C Recommendation 18 July 2002, https://www.w3.org/TR/2002/REC-xml-exc-c14n-20020718/

[Hancock]

Hancock, Polymorphic Type Checking, in Simon L. Peyton Jones, Implementation of Functional Programming Languages, Section 8, Prentice-Hall International, 1987.

[Hier]

XACML v3.0 Hierarchical Resource Profile Version 1.0. 11 March 2010. Committee Specification Draft 03. https://docs.oasis-open.org/xacml/3.0/xacml-3.0-hierarchical-v1-spec-cd-03-en.html

[IEEE754]

IEEE Standard for Binary Floating-Point Arithmetic 1985, ISBN 1-5593-7653-8, IEEE Product No. SH10116-TBR.

[INFOSET]

XML Information Set (Second Edition), W3C Recommendation, 4 February 2004, https://www.w3.org/TR/xml-infoset/

[ISO10181-3]

ISO/IEC 10181-3:1996 Information technology – Open Systems Interconnection -- Security frameworks for open systems: Access control framework.

[JSLT]

Schibsted Media AS, *JSLT: JSON Query and Transformation Language*, 2022, https://github.com/schibsted/jslt

[Kudo00]

Kudo M and Hada S, XML document security based on provisional authorization, Proceedings of the Seventh ACM Conference on Computer and Communications Security, Nov 2000, Athens, Greece, pp 87-96.

[LDAP-1]

RFC2256, A summary of the X500(96) User Schema for use with LDAPv3, Section 5, M Wahl, December 1997, https://www.ietf.org/rfc/rfc2256.txt

[LDAP-2]

RFC2798, Definition of the inetOrgPerson, M. Smith, April 2000, https://www.ietf.org/rfc/rfc2798.txt

[MathML]

Mathematical Markup Language (MathML), Version 2.0, W3C Recommendation, 21 October 2003, https://www.w3.org/TR/2003/REC-MathML2-20031021/

[Multi]

OASIS Committee Draft 03, XACML v3.0 Multiple Decision Profile Version 1.0, 11 March 2010, https://docs.oasis-open.org/xacml/3.0/xacml-3.0-multiple-v1-spec-cd-03-en.doc

[Perritt93]

Perritt, H. Knowbots, Permissions Headers and Contract Law, Conference on Technological Strategies for Protecting Intellectual Property in the Networked Multimedia Environment, April 1993. Available at: https://www.cni.org/resources/historical-resources/technological-strategies-for-protecting-intellectual-property-in-the-networked-multimedia-environment/permission-headers-and-contract-law

[RBAC]

David Ferraiolo and Richard Kuhn, Role-Based Access Controls, 15th National Computer Security Conference, 1992.

[RFC2119]

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

[RFC2396]

RFC 2396, Berners-Lee T, Fielding R, Masinter L, Uniform Resource Identifiers (URI): Generic Syntax, https://www.ietf.org/rfc/rfc2396.txt

[RFC2732]

RFC 2732, Hinden R, Carpenter B, Masinter L, Format for Literal IPv6 Addresses in URL's, https://www.ietf.org/rfc/rfc2732.txt

[RFC3198]

IETF RFC 3198: Terminology for Policy-Based Management, November 2001. https://www.ietf.org/rfc/rfc3198.txt

[RFC8174]

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

[RFC9535]

RFC 9535, *JSONPath: Query Expressions for JSON*, February 2024. [Online]. Available: https://datatracker.ietf.org/doc/html/rfc9535

[UAX15]

Mark Davis, Martin Dürst, Unicode Standard Annex #15: Unicode Normalization Forms, Unicode 5.1, https://unicode.org/reports/tr15/

[UTR36]

Davis, Mark, Suignard, Michel, Unicode Technical Report #36: Unicode Security Considerations, https://www.unicode.org/reports/tr36/

[XF]

W3C XQuery, XPath, and XSLT Functions and Operators Namespace Document (XPath and XQuery Functions and Operators 3.1) 21 March 2017, https://www.w3.org/2005/xpath-functions/

[XS]

XML Schema, parts 1 and 2. Available at: https://www.w3.org/TR/xmlschema-1/ and https://www.w3.org/TR/xmlschema-2/

B.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.

[CM]

Character Model for the World Wide Web: String Matching W3C Working Group Note 11 August 2021, https://www.w3.org/TR/charmod-norm/, World Wide Web Consortium.

[Hinton94]

Hinton, H, M, Lee, E, S, The Compatibility of Policies, Proceedings 2nd ACM Conference on Computer and Communications Security, Nov 1994, Fairfax, Virginia, USA.

[Sloman94]

Sloman, M. Policy Driven Management for Distributed Systems. Journal of Network and Systems Management, Volume 2, part 4. Plenum Press. 1994.

Annex C JACAL identifiers (normative)

This section defines standard identifiers for commonly used entities.

C.1 JACAL schema identifier

The JACAL core schema is defined using this identifier (`$id`):

`urn:oasis:names:tc:jacal:1.0:core:schema`

Annex D JSON Schema (normative)

This section includes the JSON Schema for the JACAL syntax defined in this specification, more particularly in section 5 (i.e. obtained by applying the ACAL-to-JSON mapping rules):

```
{
        "$schema": "https://json-schema.org/draft/2020-12/schema",
        "$id": "urn:oasis:names:tc:jacal:1.0:core:schema",
        "title": "JSON schema for the JSON Representation of ACAL Core Version 1.0 (JACAL)",
        "description": "Copyright © OASIS Open 2026. All Rights Reserved.  For license and copyright information, and complete 
        "examples": [
                {
                        "Bundle": {}
                }
        ],
        "anyOf": [
                {
                        "type": "object",
                        "required": [
                                "Policy"
                        ],
                        "properties": {
                                "Policy": {
                                        "$ref": "#/$defs/PolicyType"
                                }
                        },
                        "additionalProperties": false
                },
                {
                        "type": "object",
                        "required": [
                                "Bundle"
                        ],
                        "properties": {
                                "Bundle": {
                                        "$ref": "#/$defs/BundleType"
                                }
                        },
                        "additionalProperties": false
                },
                {
                        "type": "object",
                        "required": [
                                "Request"
                        ],
                        "properties": {
                                "Request": {
                                        "$ref": "#/$defs/RequestType"
                                }
                        },
                        "additionalProperties": false
                },
                {
                        "type": "object",
                        "required": [
                                "Response"
                        ],
                        "properties": {
                                "Response": {
                                        "$ref": "#/$defs/ResponseType"
                                }
                        },
                        "additionalProperties": false
                }
        ],
        "$defs": {
                "Name": {
                        "type": "string",
                        "pattern": "^[_:A-Za-z][-._:A-Za-z0-9]*$"
                },
                "VersionType": {
                        "description": "Policy version (backslashes must be escaped in JSON)",
                        "type": "string",
                        "pattern": "^(0|[1-9]\\d*)(\\.(0|[1-9]\\d*)){0,3}$"
                },
                "VersionMatchType": {
                        "description": "Policy version match pattern (backslashes must be escaped in JSON)",
                        "type": "string",
                        "pattern": "^(0|[1-9]\\d*|\\*)(\\.(0|[1-9]\\d*|\\*|\\+)){0,3}$"
                },
                "ShortIdNameType": {
                        "description": "Short Identifier alias",
                        "type": "string",
                        "pattern": "^[A-Za-z][0-9A-Za-z]*(-[0-9A-Za-z]+)*$"
                },
                "ShortIdValueType": {
                        "description": "Short Identifier value (long identifier)",
                        "type": "string",
                        "pattern": "^[!#-;=?-\\[\\]_a-z~]*(\\{[A-Za-z][0-9A-Za-z]*(-[0-9A-Za-z]+)*\\}[!#-;=?-\\[\\]_a-z~]*)*$",
                        "minLength": 1
                },
                "IdentifierType": {
                        "description": "Identifier supporting Short Identifier names (Pattern needs to be improved as this allo
                        "type": "string",
                        "pattern": "^[^{}]*(\\{[A-Za-z][0-9A-Za-z]*(-[0-9A-Za-z]+)*\\}[^{}]*)*$"
                },
                "LocalIdentifierType": {
                        "description": "Local identifier, unique only within a Request, Policy or Rule",
                        "type": "string",
                        "pattern": "^_*[A-Za-z][A-Za-z_0-9]*([-.]_*[A-Za-z_0-9]*)*$"
                },
                "AttributeSelectorPathType": {
                        "description": "AttributeSelector Path type",
                        "type": "string",
                        "pattern": "^\\S(.*\\S)?$"
                },
                "EffectType": {
                        "enum": [
                                "Permit",
                                "Deny"
                        ]
                },
                "DecisionType": {
                        "enum": [
                                "Permit",
                                "Deny",
                                "Indeterminate",
                                "NotApplicable"
                        ]
                },
                "MediaType": {
                        "description": "Content media type (RFC 6838). The standard MediaType values defined in JACAL core spec
                        "$comment": "The pattern is based on the ABNF syntax from section 4.2 of RFC 6838 (with the recommendat
                        "type": "string",
                        "pattern": "^[A-Za-z0-9][A-Za-z0-9!#$&\\-\\^_.+]{0,63}/[A-Za-z0-9][A-Za-z0-9!#$&\\-\\^_.+]{0,63}$"
                },
                "ContentEncodingType": {
                        "description": "Content encoding name (RFC 2045). The standard ContentEncodingType value(s) defined in
                        "type": "string",
                        "pattern": "^[a-z0-9]+(-[a-z0-9]+)*$"
                },
                "ShortIdType": {
                        "type": "object",
                        "required": [
                                "Name",
                                "Value"
                        ],
                        "properties": {
                                "Name": {
                                        "$ref": "#/$defs/ShortIdNameType"
                                },
                                "Value": {
                                        "$ref": "#/$defs/ShortIdValueType"
                                }
                        },
                        "additionalProperties": false
                },
                "ShortIdSetType": {
                        "type": "object",
                        "required": [
                                "Id"
                        ],
                        "properties": {
                                "Id": {
                                        "type": "string",
                                        "format": "uri-reference"
                                },
                                "ShortIdSetReference": {
                                        "type": "array",
                                        "minItems": 1,
                                        "uniqueItems": true,
                                        "items": {
                                                "type": "string",
                                                "format": "uri-reference"
                                        }
                                },
                                "ShortId": {
                                        "$comment": "TODO: add 'uniqueKeys': ['/Name'] from ArrayExt vocabulary - https://docs.
                                        "type": "array",
                                        "minItems": 1,
                                        "items": {
                                                "$ref": "#/$defs/ShortIdType"
                                        }
                                }
                        },
                        "additionalProperties": false
                },
                "PolicyDefaultsTypeTree": {
                        "$comment": "Using object type to have one and only one property per type of PolicyDefaults (i.e. one p
                        "$dynamicRef": "#PolicyDefaultsTypeExtensions"
                },
                "PolicyDefaultsTypeTreeEmpty": {
                        "$dynamicAnchor": "PolicyDefaultsTypeExtensions",
                        "$comment": "No PolicyDefaults extension supported by default. Create your own implementation-specific
                        "not": true
                },
                "ParameterType": {
                        "type": "object",
                        "required": [
                                "Name"
                        ],
                        "properties": {
                                "Name": {
                                        "$ref": "#/$defs/LocalIdentifierType"
                                },
                                "DataType": {
                                        "default": "urn:oasis:names:tc:acal:1.0:data-type:string",
                                        "$ref": "#/$defs/IdentifierType"
                                },
                                "isBag": {
```

```
                                "default": false,
                                "type": "boolean"
                        },
                        "Description": {
                                "type": "string"
                        },
                        "Expression": {
                                "$ref": "#/$defs/ExpressionTypeTree",
                                "dependentSchemas": {
                                        "Value": {
                                                "$comment": "ACAL constraint (textual) on ParameterType's Expression pr
                                                "properties": {
                                                        "Value": {
                                                                "dependentSchemas": {
                                                                        "DataType": {
                                                                                "not": true
                                                                        }
                                                                }
                                                        }
                                                }
                                        }
                                }
                        }
                },
                "additionalProperties": false
        },
        "BooleanExpressionType": {
                "$ref": "#/$defs/NonLiteralExpressionTypeTree"
        },
        "IdReferenceType": {
                "$comment": "ACAL abstract type",
                "type": "object",
                "required": [
                        "Id"
                ],
                "properties": {
                        "Id": {
                                "type": "string",
                                "format": "uri-reference"
                        }
                }
        },
        "ExactMatchIdReferenceType": {
                "$comment": "Final subtype of IdReferenceType",
                "allOf": [
                        {
                                "$ref": "#/$defs/IdReferenceType"
                        },
                        {
                                "properties": {
                                        "Version": {
                                                "$ref": "#/$defs/VersionType"
                                        }
                                },
                                "required": [
                                        "Version"
                                ]
                        }
                ],
                "unevaluatedProperties": false
        },
        "PatternMatchIdReferenceType": {
                "$comment": "Non-final subtype of IdReferenceType",
                "allOf": [
                        {
                                "$ref": "#/$defs/IdReferenceType"
                        },
                        {
                                "properties": {
                                        "Version": {
                                                "$ref": "#/$defs/VersionMatchType"
                                        }
                                }
                        }
                ]
        },
"PolicyReferenceType": {
	"$comment": "Final subtype of PatternMatchIdReferenceType",
	"allOf": [
		{
			"$ref": "#/$defs/PatternMatchIdReferenceType"
		},
		{
			"type": "object",
			"required": [],
			"properties": {
				"Expression": {
					"type": "array",
					"minItems": 1,
					"uniqueItems": false,
					"items": {
						"$ref": "#/$defs/ExpressionTypeTree",
						"dependentSchemas": {
							"Value": {
								"$comment": "ACAL constraint (textual) on Polic
								"properties": {
									"Value": {
										"dependentSchemas": {
											"DataType": {
												"not": true
											}
										}
									}
								}
							}
						}
					}
				}
			}
		}
	],
	"unevaluatedProperties": false
},
"VariableDefinitionType": {
	"type": "object",
	"required": [
		"VariableId",
		"Expression"
	],
	"properties": {
		"VariableId": {
			"$ref": "#/$defs/LocalIdentifierType"
		},
		"Expression": {
			"$ref": "#/$defs/ExpressionTypeTree"
		}
	},
	"additionalProperties": false
},
"VariableReferenceType": {
	"$comment": "Inheritance relationship to ExpressionType is implemented in ExpressionTypeTree subschema"
	"type": "object",
	"required": [
		"VariableId"
	],
	"properties": {
		"VariableId": {
			"$ref": "#/$defs/LocalIdentifierType"
		}
	},
	"additionalProperties": false
},
"SharedVariableDefinitionType": {
	"type": "object",
	"required": [
		"Id",
		"Version",
		"Expression"
	],
	"properties": {
		"Id": {
			"$ref": "#/$defs/LocalIdentifierType"
```

```
                },
                "Version": {
                        "$ref": "#/$defs/VersionType"
                },
                "Description": {
                        "type": "string"
                },
                "ShortIdSetReference": {
                        "type": "array",
                        "minItems": 1,
                        "uniqueItems": true,
                        "items": {
                                "type": "string",
                                "format": "uri-reference"
                        }
                },
                "Parameter": {
                        "$comment": "TODO: add 'uniqueKeys': ['/Name'] from ArrayExt vocabulary - https://docs.
                        "type": "array",
                        "minItems": 1,
                        "items": {
                                "$ref": "#/$defs/ParameterType"
                        }
                },
                "Expression": {
                        "$ref": "#/$defs/ExpressionTypeTree"
                }
        },
        "additionalProperties": false
},
"SharedVariableReferenceType": {
        "$comment": "Inheritance relationship to ExpressionType is implemented in ExpressionTypeTree subschema"
        "type": "object",
        "required": [
                "Id"
        ],
        "properties": {
                "Id": {
                        "$ref": "#/$defs/LocalIdentifierType"
                },
                "Version": {
                        "$ref": "#/$defs/VersionMatchType"
                },
                "Expression": {
                        "type": "array",
                        "minItems": 1,
                        "uniqueItems": false,
                        "items": {
                                "$ref": "#/$defs/ExpressionTypeTree",
                                "dependentSchemas": {
                                        "Value": {
                                                "$comment": "ACAL constraint (textual) on SharedVariableReferen
                                                "properties": {
                                                        "Value": {
                                                                "dependentSchemas": {
                                                                        "DataType": {
                                                                                "not": true
                                                                        }
                                                                }
                                                        }
                                                }
                                        }
                                }
                        }
                }
        },
        "additionalProperties": false
},
"RuleType": {
        "type": "object",
        "required": [
                "Id",
                "Effect"
        ],
        "properties": {
                "Id": {
```

```
                                                "$ref": "#/$defs/LocalIdentifierType"
                                        },
                                        "Description": {
                                                "type": "string"
                                        },
                                        "VariableDefinition": {
                                                "$comment": "TODO: add 'uniqueKeys': ['/VariableId'] from ArrayExt vocabulary - https:/
                                                "type": "array",
                                                "minItems": 1,
                                                "items": {
                                                        "$ref": "#/$defs/VariableDefinitionType"
                                                }
                                        },
                                        "Condition": {
                                                "$ref": "#/$defs/BooleanExpressionType"
                                        },
                                        "Effect": {
                                                "$ref": "#/$defs/EffectType"
                                        },
                                        "NoticeExpression": {
                                                "type": "array",
                                                "minItems": 1,
                                                "items": {
                                                        "$ref": "#/$defs/NoticeExpressionType"
                                                }
                                        }
                                },
                                "additionalProperties": false
                        },
                        "PolicyType": {
                                "type": "object",
                                "required": [
                                        "PolicyId",
                                        "Version",
                                        "CombiningAlgId"
                                ],
                                "properties": {
                                        "PolicyId": {
                                                "type": "string",
                                                "format": "uri-reference"
                                        },
                                        "Version": {
                                                "$ref": "#/$defs/VersionType"
                                        },
                                        "Description": {
                                                "type": "string"
                                        },
                                        "ShortIdSetReference": {
                                                "type": "array",
                                                "minItems": 1,
                                                "uniqueItems": true,
                                                "items": {
                                                        "type": "string",
                                                        "format": "uri-reference"
                                                }
                                        },
                                        "MaxDelegationDepth": {
                                                "type": "integer",
                                                "minimum": 0
                                        },
                                        "PolicyIssuer": {
                                                "$ref": "#/$defs/EntityType"
                                        },
                                        "PolicyDefaults": {
                                                "$ref": "#/$defs/PolicyDefaultsTypeTree"
                                        },
                                        "Parameter": {
                                                "$comment": "TODO: add 'uniqueKeys': ['/Name'] from ArrayExt vocabulary - https://docs.
                                                "type": "array",
                                                "minItems": 1,
                                                "items": {
                                                        "$ref": "#/$defs/ParameterType"
                                                }
                                        },
                                        "VariableDefinition": {
                                                "$comment": "TODO: add 'uniqueKeys': ['/VariableId'] from ArrayExt vocabulary - https:/
                                                "type": "array",
                                                "minItems": 1,
                                                "items": {
                                                        "$ref": "#/$defs/VariableDefinitionType"
                                                }
                                        },
                                        "Target": {
                                                "$ref": "#/$defs/BooleanExpressionType"
                                        },
                                        "CombiningAlgId": {
                                                "$ref": "#/$defs/IdentifierType"
                                        },
                                        "CombinerInput": {
                                                "type": "array",
                                                "minItems": 1,
                                                "items": {
                                                        "$ref": "#/$defs/CombinerInputTypeTree"
                                                }
                                        },
                                        "NoticeExpression": {
                                                "type": "array",
                                                "minItems": 1,
                                                "items": {
                                                        "$ref": "#/$defs/NoticeExpressionType"
                                                }
                                        }
                                },
                                "additionalProperties": false
                        },
                        "CombinerInputTypeTree": {
                                "$comment": "CombinerInputType's direct subtypes",
                                "anyOf": [
                                        {
                                                "type": "object",
                                                "required": [
                                                        "Policy"
                                                ],
                                                "properties": {
                                                        "Policy": {
                                                                "$ref": "#/$defs/PolicyType"
                                                        }
                                                },
                                                "additionalProperties": false
                                        },
                                        {
                                                "type": "object",
                                                "required": [
                                                        "PolicyReference"
                                                ],
                                                "properties": {
                                                        "PolicyReference": {
                                                                "$ref": "#/$defs/PolicyReferenceType"
                                                        }
                                                },
                                                "additionalProperties": false
                                        },
                                        {
                                                "type": "object",
                                                "required": [
                                                        "Rule"
                                                ],
                                                "properties": {
                                                        "Rule": {
                                                                "$ref": "#/$defs/RuleType"
                                                        }
                                                },
                                                "additionalProperties": false
                                        }
                                ]
                        },
                        "ValueType": {
                                "anyOf": [
                                        {
                                                "$comment": "Case when the DataType property is unnecessary (implicit or already define
                                                "type": [
                                                        "boolean",
```

```json
                                "integer",
                                "number",
                                "string"
                        ]
                },
                {
                        "$comment": "Same case as above but for structured values",
                        "$ref": "#/$defs/StructuredValueTypeTree"
                },
                {
                        "$comment": "Case when the DataType property is required (not the case for boolean / in
                        "allOf": [
                                {
                                        "type": "object",
                                        "properties": {
                                                "DataType": {
                                                        "$ref": "#/$defs/IdentifierType"
                                                }
                                        },
                                        "required": [
                                                "DataType"
                                        ]
                                },
                                {
                                        "anyOf": [
                                                {
                                                        "properties": {
                                                                "Value": {
                                                                        "$comment": "For DataTypes represented 
                                                                        "type": "string"
                                                                }
                                                        },
                                                        "required": [
                                                                "Value"
                                                        ]
                                                },
                                                {
                                                        "$ref": "#/$defs/StructuredValueTypeTree"
                                                }
                                        ]
                                }
                        ],
                        "unevaluatedProperties": false
                }
        ]
},
"StructuredValueTypeTree": {
        "$dynamicRef": "#StructuredValueTypeExtensions"
},
"StructuredValueTypeTreeEmpty": {
        "$dynamicAnchor": "StructuredValueTypeExtensions",
        "$comment": "No StructuredValueType extension supported by default. Create your own implementation-spec
        "not": true
},
"ApplyType": {
        "$comment": "ApplyType's inheritance relationship to ExpressionType is implemented in ExpressionTypeTre
        "type": "object",
        "required": [
                "FunctionId"
        ],
        "properties": {
                "Description": {
                        "type": "string"
                },
                "FunctionId": {
                        "$ref": "#/$defs/IdentifierType"
                },
                "Expression": {
                        "type": "array",
                        "minItems": 1,
                        "uniqueItems": false,
                        "items": {
                                "$ref": "#/$defs/ExpressionTypeTree"
                        }
                }
        },
        "additionalProperties": false
},
"FunctionType": {
        "$comment": "FunctionType's inheritance relationship to ExpressionType is implemented in ExpressionType
        "type": "object",
        "required": [
                "Id"
        ],
        "properties": {
                "Id": {
                        "$ref": "#/$defs/IdentifierType"
                }
        },
        "additionalProperties": false
},
"NamedAttributeDesignatorType": {
        "$comment": "Abstract ACAL type",
        "type": "object",
        "required": [
                "AttributeId"
        ],
        "properties": {
                "AttributeId": {
                        "$ref": "#/$defs/IdentifierType"
                },
                "DataType": {
                        "default": "urn:oasis:names:tc:acal:1.0:data-type:string",
                        "$ref": "#/$defs/IdentifierType"
                },
                "Issuer": {
                        "$ref": "#/$defs/Name"
                },
                "MustBePresent": {
                        "default": false,
                        "type": "boolean"
                }
        }
},
"AttributeDesignatorType": {
        "allOf": [
                {
                        "$ref": "#/$defs/NamedAttributeDesignatorType"
                },
                {
                        "type": "object",
                        "required": [
                                "Category"
                        ],
                        "properties": {
                                "Category": {
                                        "$ref": "#/$defs/IdentifierType"
                                }
                        }
                }
        ],
        "unevaluatedProperties": false
},
"EntityAttributeDesignatorType": {
        "allOf": [
                {
                        "$ref": "#/$defs/NamedAttributeDesignatorType"
                },
                {
                        "type": "object",
                        "required": [
                                "Expression"
                        ],
                        "properties": {
                                "Expression": {
                                        "$ref": "#/$defs/ExpressionTypeTree"
                                }
                        }
                }
        ],
        "unevaluatedProperties": false
},
"NamedAttributeDesignatorTypeTree": {
        "$comment": "NamedAttributeDesignatorType's direct subtypes",
        "anyOf": [
                {
                        "type": "object",
                        "required": [
                                "AttributeDesignator"
                        ],
                        "properties": {
                                "AttributeDesignator": {
                                        "$ref": "#/$defs/AttributeDesignatorType"
                                }
                        },
                        "additionalProperties": false
                },
                {
                        "type": "object",
                        "required": [
                                "EntityAttributeDesignator"
                        ],
                        "properties": {
                                "EntityAttributeDesignator": {
                                        "$ref": "#/$defs/EntityAttributeDesignatorType"
                                }
                        },
                        "additionalProperties": false
                }
        ]
},
"BaseAttributeSelectorType": {
        "$comment": "Abstract ACAL type",
        "type": "object",
        "required": [
                "Path"
        ],
        "properties": {
                "Path": {
                        "$ref": "#/$defs/AttributeSelectorPathType"
                },
                "DataType": {
                        "default": "urn:oasis:names:tc:acal:1.0:data-type:string",
                        "$ref": "#/$defs/IdentifierType"
                },
                "MustBePresent": {
                        "default": false,
                        "type": "boolean"
                }
        }
},
"AttributeSelectorType": {
        "allOf": [
                {
                        "$ref": "#/$defs/BaseAttributeSelectorType"
                },
                {
                        "type": "object",
                        "required": [
                                "Category"
                        ],
                        "properties": {
                                "Category": {
                                        "$ref": "#/$defs/IdentifierType"
                                }
                        }
                }
        ],
        "unevaluatedProperties": false
},
"EntityAttributeSelectorType": {
        "allOf": [
                {
                        "$ref": "#/$defs/BaseAttributeSelectorType"
                },
                {
                        "type": "object",
                        "required": [
                                "Expression"
                        ],
                        "properties": {
                                "Expression": {
                                        "$ref": "#/$defs/ExpressionTypeTree"
                                }
                        }
                }
        ],
        "unevaluatedProperties": false
},
"BaseAttributeSelectorTypeTree": {
        "$comment": "BaseAttributeSelectorType's direct subtypes",
        "anyOf": [
                {
                        "$ref": "#/$defs/AttributeSelectorTypeTree"
                },
                {
                        "$ref": "#/$defs/EntityAttributeSelectorTypeTree"
                }
        ]
},
"AttributeSelectorTypeTree": {
        "$dynamicRef": "#AttributeSelectorTypeExtensions"
},
"AttributeSelectorTypeTreeEmpty": {
        "$dynamicAnchor": "AttributeSelectorTypeExtensions",
        "$comment": "No AttributeSelector extension supported by default. Create your own implementation-specifi
        "not": true
},
"EntityAttributeSelectorTypeTree": {
        "$dynamicRef": "#EntityAttributeSelectorTypeExtensions"
},
"EntityAttributeSelectorTypeTreeEmpty": {
        "$dynamicAnchor": "EntityAttributeSelectorTypeExtensions",
        "$comment": "No EntityAttributeSelector extension supported by default. Create your own implementation-
        "not": true
},
"QuantifiedExpressionType": {
        "type": "object",
        "required": [
                "VariableId",
                "Domain",
                "Iterant"
        ],
        "properties": {
                "VariableId": {
                        "$ref": "#/$defs/LocalIdentifierType"
                },
                "Domain": {
                        "$ref": "#/$defs/NonLiteralExpressionTypeTree"
                },
                "Iterant": {
                        "$ref": "#/$defs/ExpressionTypeTree"
                }
        }
},
"QuantifiedExpressionTypeTree": {
        "$comment": "QuantifiedExpressionType's direct subtypes",
        "anyOf": [
                {
                        "type": "object",
                        "required": [
                                "ForAny"
                        ],
                        "properties": {
                                "ForAny": {
                                        "$ref": "#/$defs/QuantifiedExpressionType"
                                }
                        },
                        "additionalProperties": false
                },
                {
                        "type": "object",
                        "required": [
                                "ForAll"
                        ],
                        "properties": {
                                "ForAll": {
                                        "$ref": "#/$defs/QuantifiedExpressionType"
                                }
                        },
                        "additionalProperties": false
                },
                {
                        "type": "object",
                        "required": [
                                "Map"
                        ],
                        "properties": {
                                "Map": {
                                        "$ref": "#/$defs/QuantifiedExpressionType"
                                }
                        },
                        "additionalProperties": false
                },
                {
                        "type": "object",
                        "required": [
                                "Select"
                        ],
                        "properties": {
                                "Select": {
                                        "$ref": "#/$defs/QuantifiedExpressionType"
                                }
                        },
                        "additionalProperties": false
                }
        ]
},
"ExpressionTypeTree": {
        "$comment": "ExpressionType's direct subtypes",
        "anyOf": [
                {
                        "type": "object",
                        "required": [
                                "Value"
                        ],
                        "properties": {
                                "Value": {
                                        "$ref": "#/$defs/ValueType"
                                }
                        },
                        "additionalProperties": false
                },
                {
                        "type": "object",
                        "required": [
                                "Function"
                        ],
                        "properties": {
                                "Function": {
                                        "$ref": "#/$defs/FunctionType"
                                }
                        }
                },
                {
                        "$ref": "#/$defs/NonLiteralExpressionTypeTree"
                }
        ]
},
"NonLiteralExpressionTypeTree": {
        "$comment": "ExpressionType's direct subtypes except ValueType",
        "anyOf": [
                {
                        "type": "object",
                        "required": [
                                "VariableReference"
                        ],
                        "properties": {
                                "VariableReference": {
                                        "$ref": "#/$defs/VariableReferenceType"
```

```
                                                }
                                        },
                                        "additionalProperties": false
                                },
                                {
                                        "type": "object",
                                        "required": [
                                                "SharedVariableReference"
                                        ],
                                        "properties": {
                                                "SharedVariableReference": {
                                                        "$ref": "#/$defs/SharedVariableReferenceType"
                                                }
                                        },
                                        "additionalProperties": false
                                },
                                {
                                        "type": "object",
                                        "required": [
                                                "Apply"
                                        ],
                                        "properties": {
                                                "Apply": {
                                                        "$ref": "#/$defs/ApplyType"
                                                }
                                        }
                                },
                                {
                                        "$ref": "#/$defs/NamedAttributeDesignatorTypeTree"
                                },
                                {
                                        "$comment": "TODO: remove this subschema if (Entity)AttributeSelectorType is not suppor
                                        "$ref": "#/$defs/BaseAttributeSelectorTypeTree"
                                },
                                {
                                        "$comment": "TODO: remove this subschema if QuantifiedExpressionType is not supported b
                                        "$ref": "#/$defs/QuantifiedExpressionTypeTree"
                                }
                        ]
                },
                "AttributeAssignmentExpressionType": {
                        "type": "object",
                        "required": [
                                "AttributeId",
                                "Expression"
                        ],
                        "properties": {
                                "AttributeId": {
                                        "$ref": "#/$defs/IdentifierType"
                                },
                                "Category": {
                                        "$ref": "#/$defs/IdentifierType"
                                },
                                "Issuer": {
                                        "$ref": "#/$defs/Name"
                                },
                                "Expression": {
                                        "$ref": "#/$defs/ExpressionTypeTree"
                                }
                        },
                        "additionalProperties": false
                },
                "NoticeExpressionType": {
                        "type": "object",
                        "required": [
                                "Id"
                        ],
                        "properties": {
                                "Id": {
                                        "$ref": "#/$defs/IdentifierType"
                                },
                                "IsObligation": {
                                        "type": "boolean"
                                },
                                "AppliesTo": {
                                        "$ref": "#/$defs/EffectType"
                                },
                                "Condition": {
                                        "$ref": "#/$defs/BooleanExpressionType"
                                },
                                "AttributeAssignmentExpression": {
                                        "$comment": "TODO: add 'uniqueKeys': ['/AttributeId', '/Category'] from ArrayExt vocabul
                                        "type": "array",
                                        "minItems": 1,
                                        "items": {
                                                "$ref": "#/$defs/AttributeAssignmentExpressionType"
                                        }
                                }
                        },
                        "additionalProperties": false
                },
                "NoticeType": {
                        "type": "object",
                        "required": [
                                "Id"
                        ],
                        "properties": {
                                "Id": {
                                        "$ref": "#/$defs/IdentifierType"
                                },
                                "IsObligation": {
                                        "default": false,
                                        "type": "boolean"
                                },
                                "AttributeAssignment": {
                                        "$comment": "TODO: add 'uniqueKeys': ['/AttributeId', '/Category'] from ArrayExt vocabul
                                        "type": "array",
                                        "minItems": 1,
                                        "items": {
                                                "$ref": "#/$defs/AttributeAssignmentType"
                                        }
                                }
                        },
                        "additionalProperties": false
                },
                "AttributeType": {
                        "type": "object",
                        "required": [
                                "AttributeId",
                                "Value"
                        ],
                        "properties": {
                                "AttributeId": {
                                        "$ref": "#/$defs/IdentifierType"
                                },
                                "Issuer": {
                                        "$ref": "#/$defs/Name"
                                },
                                "DataType": {
                                        "default": "urn:oasis:names:tc:acal:1.0:data-type:string",
                                        "$ref": "#/$defs/IdentifierType"
                                },
                                "Value": {
                                        "type": "array",
                                        "minItems": 1,
                                        "items": {
                                                "$ref": "#/$defs/ValueType",
                                                "dependentSchemas": {
                                                        "DataType": {
                                                                "$comment": "ACAL constraint (textual) on AttributeType's Value
                                                                "not": true
                                                        }
                                                }
                                        }
                                }
                        }
                },
                "AttributeAssignmentType": {
                        "allOf": [
                                {
                                        "$ref": "#/$defs/AttributeType"
                                },
                                {
                                    "type": "object",
                                    "required": [],
                                    "properties": {
                                        "Category": {
                                            "$ref": "#/$defs/IdentifierType"
                                        }
                                    }
                                }
                            ],
                            "unevaluatedProperties": false
                        },
                        "ResultEntityType": {
                            "type": "object",
                            "required": [
                                "Category",
                                "Attribute"
                            ],
                            "properties": {
                                "Category": {
                                    "$ref": "#/$defs/IdentifierType"
                                },
                                "Id": {
                                    "$ref": "#/$defs/LocalIdentifierType"
                                },
                                "Attribute": {
                                    "$comment": "TODO: add 'uniqueKeys': ['/AttributeId'] from ArrayExt vocabulary - https:
                                    "type": "array",
                                    "minItems": 1,
                                    "items": {
                                        "$ref": "#/$defs/AttributeType"
                                    }
                                }
                            }
                        },
                        "RequestDefaultsTypeTree": {
                            "$comment": "Using object type to have one and only one property per type of RequestDefaults (i.e. one
                            "$dynamicRef": "#RequestDefaultsTypeExtensions"
                        },
                        "RequestDefaultsTypeTreeEmpty": {
                            "$dynamicAnchor": "RequestDefaultsTypeExtensions",
                            "$comment": "No RequestDefaults extension supported by default. Create your own implementation-specific
                            "not": true
                        },
                        "RequestType": {
                            "type": "object",
                            "required": [
                                "RequestEntity"
                            ],
                            "properties": {
                                "ShortIdSetReference": {
                                    "type": "array",
                                    "minItems": 1,
                                    "uniqueItems": true,
                                    "items": {
                                        "type": "string",
                                        "format": "uri-reference"
                                    }
                                },
                                "RequestDefaults": {
                                    "$ref": "#/$defs/RequestDefaultsTypeTree"
                                },
                                "RequestEntity": {
                                    "$comment": "TODO: add 'uniqueKeys': ['/Id'] from ArrayExt vocabulary - https://docs.js
                                    "type": "array",
                                    "minItems": 1,
                                    "items": {
                                        "$ref": "#/$defs/RequestEntityType"
                                    }
                                },
                                "MultiRequests": {
                                    "$ref": "#/$defs/MultiRequestsType"
                                },
                                "ReturnPolicyIdList": {
                                    "default": false,
                                    "type": "boolean"
```

```
                                        },
                                        "CombinedDecision": {
                                                "default": false,
                                                "type": "boolean"
                                        }
                                },
                                "additionalProperties": false
                        },
                        "RequestEntityType": {
                                "type": "object",
                                "required": [
                                        "Category"
                                ],
                                "properties": {
                                        "Category": {
                                                "$ref": "#/$defs/IdentifierType"
                                        },
                                        "Id": {
                                                "$ref": "#/$defs/LocalIdentifierType"
                                        },
                                        "Content": {
                                                "$ref": "#/$defs/ContentType"
                                        },
                                        "RequestAttribute": {
                                                "$comment": "TODO: add 'uniqueKeys': ['/AttributeId'] from ArrayExt vocabulary - https:
                                                "type": "array",
                                                "minItems": 1,
                                                "items": {
                                                        "$ref": "#/$defs/RequestAttributeType"
                                                }
                                        }
                                },
                                "additionalProperties": false
                        },
                        "ContentType": {
                                "$comment": "TODO: supporting this is optional. Remove this subschema if your implementation does not s
                                "type": "object",
                                "properties": {
                                        "MediaType": {
                                                "$ref": "#/$defs/MediaType",
                                                "default": "application/json"
                                        },
                                        "Encoding": {
                                                "$ref": "#/$defs/ContentEncodingType"
                                        },
                                        "Body": {
                                                "$comment": "'object' type is used for JSON object, 'string' for others - e.g. XML - po
                                                "type": [
                                                        "string",
                                                        "object"
                                                ]
                                        }
                                },
                                "required": [
                                        "Body"
                                ],
                                "additionalProperties": false
                        },
                        "RequestAttributeType": {
                                "allOf": [
                                        {
                                                "$ref": "#/$defs/AttributeType"
                                        },
                                        {
                                                "type": "object",
                                                "required": [],
                                                "properties": {
                                                        "IncludeInResult": {
                                                                "default": false,
                                                                "type": "boolean"
                                                        }
                                                }
                                        }
                                ],
                                "unevaluatedProperties": false
                        },
                        "ResponseType": {
                                "type": "object",
                                "required": [
                                        "Result"
                                ],
                                "properties": {
                                        "ShortIdSetReference": {
                                                "type": "array",
                                                "minItems": 1,
                                                "uniqueItems": true,
                                                "items": {
                                                        "type": "string",
                                                        "format": "uri-reference"
                                                }
                                        },
                                        "Result": {
                                                "type": "array",
                                                "minItems": 1,
                                                "items": {
                                                        "$ref": "#/$defs/ResultType"
                                                }
                                        }
                                },
                                "additionalProperties": false
                        },
                        "ResultType": {
                                "type": "object",
                                "required": [
                                        "Decision"
                                ],
                                "properties": {
                                        "Decision": {
                                                "$ref": "#/$defs/DecisionType"
                                        },
                                        "Status": {
                                                "$ref": "#/$defs/StatusType"
                                        },
                                        "Notice": {
                                                "type": "array",
                                                "minItems": 1,
                                                "items": {
                                                        "$ref": "#/$defs/NoticeType"
                                                }
                                        },
                                        "ResultEntity": {
                                                "$comment": "TODO: add 'uniqueKeys': ['/Category'] from ArrayExt vocabulary - https://d
                                                "type": "array",
                                                "minItems": 1,
                                                "items": {
                                                        "$ref": "#/$defs/ResultEntityType"
                                                }
                                        },
                                        "ApplicablePolicyReference": {
                                                "$comment": "TODO: add 'uniqueKeys': ['/Id'] from ArrayExt vocabulary - https://docs.js
                                                "type": "array",
                                                "minItems": 1,
                                                "items": {
                                                        "$ref": "#/$defs/ExactMatchIdReferenceType"
                                                }
                                        }
                                },
                                "additionalProperties": false
                        },
                        "MultiRequestsType": {
                                "$comment": "TODO: remove this if your implementation does not support MultiRequests (the Multiple Deci
                                "type": "object",
                                "required": [
                                        "RequestReference"
                                ],
                                "properties": {
                                        "RequestReference": {
                                                "$comment": "TODO: translate/enforce ACAL constraint: {OCL} self->isUnique(RequestEntit
                                                "type": "array",
                                                "minItems": 1,
                                                "items": {
                                                        "$ref": "#/$defs/RequestReferenceType"
                                                }
                                        }
                                },
                                "additionalProperties": false
                        },
                        "RequestReferenceType": {
                                "$comment": "TODO: remove this subschema if your implementation does not support MultiRequests (the Mul
                                "type": "object",
                                "required": [
                                        "RequestEntityReference"
                                ],
                                "properties": {
                                        "RequestEntityReference": {
                                                "$comment": "TODO: add 'uniqueKeys': ['/Id'] from ArrayExt vocabulary - https://docs.js
                                                "type": "array",
                                                "minItems": 1,
                                                "items": {
                                                        "$ref": "#/$defs/RequestEntityReferenceType"
                                                }
                                        }
                                },
                                "additionalProperties": false
                        },
                        "RequestEntityReferenceType": {
                                "type": "object",
                                "required": [
                                        "Id"
                                ],
                                "properties": {
                                        "Id": {
                                                "$ref": "#/$defs/LocalIdentifierType"
                                        }
                                },
                                "additionalProperties": false
                        },
                        "StatusType": {
                                "type": "object",
                                "required": [
                                        "StatusCode"
                                ],
                                "properties": {
                                        "StatusCode": {
                                                "$ref": "#/$defs/StatusCodeType"
                                        },
                                        "StatusMessage": {
                                                "type": "string"
                                        },
                                        "StatusDetail": {
                                                "$ref": "#/$defs/StatusDetailType"
                                        }
                                },
                                "additionalProperties": false
                        },
                        "StatusCodeType": {
                                "type": "object",
                                "required": [
                                        "Value"
                                ],
                                "properties": {
                                        "Value": {
                                                "$ref": "#/$defs/IdentifierType"
                                        },
                                        "StatusCode": {
                                                "$ref": "#/$defs/StatusCodeType"
                                        }
                                },
                                "additionalProperties": false
                        },
                        "StatusDetailType": {
                                "$comment": "TODO: supporting this is optional. Remove this subschema if your implementation does not s
                                "type": "object",
                                "properties": {
                                        "MissingAttributeDetail": {
                                                "type": "array",
                                                "minItems": 1,
                                                "items": {
```

```
                                                "$ref": "#/$defs/MissingAttributeDetailType"
                                        }
                                }
                        }
                },
                "MissingAttributeDetailType": {
                        "type": "object",
                        "required": [
                                "Category",
                                "AttributeId",
                                "DataType"
                        ],
                        "properties": {
                                "Category": {
                                        "$ref": "#/$defs/IdentifierType"
                                },
                                "AttributeId": {
                                        "$ref": "#/$defs/IdentifierType"
                                },
                                "Issuer": {
                                        "$ref": "#/$defs/Name"
                                },
                                "DataType": {
                                        "$ref": "#/$defs/IdentifierType"
                                },
                                "Value": {
                                        "type": "array",
                                        "minItems": 1,
                                        "items": {
                                                "$ref": "#/$defs/ValueType",
                                                "dependentSchemas": {
                                                        "DataType": {
                                                                "$comment": "ACAL constraint (textual) on MissingAttributeDetai
                                                                "not": true
                                                        }
                                                }
                                        }
                                }
                        }
                },
                "EntityType": {
                        "type": "object",
                        "required": [],
                        "minProperties": 1,
                        "properties": {
                                "Content": {
                                        "$ref": "#/$defs/ContentType"
                                },
                                "Attribute": {
                                        "$comment": "TODO: add 'uniqueKeys': ['/AttributeId'] from ArrayExt vocabulary - https:
                                        "type": "array",
                                        "minItems": 1,
                                        "items": {
                                                "$ref": "#/$defs/AttributeType"
                                        }
                                }
                        },
                        "if": {
                                "not": {
                                        "required": [
                                                "Content"
                                        ]
                                }
                        },
                        "then": {
                                "required": [
                                        "Attribute"
                                ]
                        }
                },
                "BundleType": {
                        "type": "object",
                        "required": [],
                        "properties": {
                                "ShortIdSet": {
                                        "$comment": "TODO: add 'uniqueKeys': ['/Id'] from ArrayExt vocabulary - https://docs.js
                        "type": "array",
                        "minItems": 1,
                        "items": {
                                "$ref": "#/$defs/ShortIdSetType"
                        }
                },
                "SharedVariableDefinition": {
                        "$comment": "TODO: add 'uniqueKeys': ['/Id'] from ArrayExt vocabulary - https://docs.js
                        "type": "array",
                        "minItems": 1,
                        "items": {
                                "$ref": "#/$defs/SharedVariableDefinitionType"
                        }
                },
                "Policy": {
                        "$comment": "TODO: add 'uniqueKeys': ['/PolicyId'] from ArrayExt vocabulary - https://d
                        "type": "array",
                        "minItems": 1,
                        "items": {
                                "$ref": "#/$defs/PolicyType"
                        }
                },
                "PolicyReference": {
                        "$ref": "#/$defs/PolicyReferenceType"
                }
        },
        "dependentRequired": {
                "PolicyReference": [
                        "Policy"
                ]
        }
    }
  }
}
```

Annex E How to generate HTML and PDF versions

Prerequisites

Install Pandoc on your system; or simply use Docker with the following shell alias:

```
$ alias pandoc='docker run --rm --volume "$(pwd):/data" pandoc/extra'
```

OASIS staff are currently using pandoc 3.0 from https://github.com/jgm/pandoc/releases/tag/3.0.

Git clone or get a local copy of OASIS XACML TC Github repository, open a terminal and **change your working directory to the root directory of your local copy of the repository**.

CSS stylesheet

The generation command uses a CSS stylesheet file (`-c` argument) provided by OASIS. It may be changed to one of these (or the local version in the `styles` folder) to get a different style of output:

- https://docs.oasis-open.org/templates/css/markdown-styles-v1.7.3.css
- https://docs.oasis-open.org/templates/css/markdown-styles-v1.7.3a.css (this one produces HTML that resembles the github display more closely, especially for blocks of code) This template already includes a reference (in HTML code) to this .css file.
- https://docs.oasis-open.org/templates/css/markdown-styles-v1.8.1-cn_final.css

HTML generation

Run the following command line to generate HTML from this markdown file (named `acal-core-json-v1.0-csd01.md`) to an output file `/tmp/acal-core-json-v1.0-csd01.html` :

```
$ pandoc -s --embed-resources -f gfm+definition_lists -c styles/markdown-styles-v1.7.3a.css -F pandoc-include -M lang=en -M tit
```

Note this command generates a Table of Contents (TOC) in HTML which is located at the top of the HTML document, and which requires additional editing in order to be published in the expected OASIS style. This editing will be handled by OASIS staff during publication.

PDF generation

For PDF output, the command line is the following (different `-t` and `-H` arguments, and output file `/tmp/acal-core-json-v1.0-csd01.pdf`):

```
$ pandoc -s --embed-resources -f gfm+definition_lists -c styles/markdown-styles-v1.7.3a.css -F pandoc-include -H pandoc/custom_
```

Appendix 1. Acknowledgments

(This appendix does not form an integral part of this Specification and is informational.)

Leadership

The following individuals have had significant leadership positions during the development of this document, not just this version of the document, and they are gratefully acknowledged:

- Chairs
  - Bill Parducci, Individual
- Secretaries
  - Bill Parducci, Individual
- Editors
  - Steven Legg, ViewDS Identity Solutions
  - Cyril Dangerville, THALES

Special Thanks

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

Steven Legg, ViewDS Identity Solutions
Cyril Dangerville, THALES

Participants

The following individuals were members of this committee during the creation of this document, not just this version of the document, and their contributions are gratefully acknowledged:

XACML TC Members:

- Hal Lockhart, Individual
- Bill Parducci, Individual
- Steven Legg, ViewDS Identity Solutions
- Cyril Dangerville, THALES

Appendix 2 Changes From Previous Version

(This appendix does not form an integral part of this Specification and is informational.)

None. This is the first version of the document.

Revision History

Latest revision history can be obtained from OASIS XACML TC's github repository.