DIGITALISERINGSSTYRELSEN

>

# OIO Basic Privilege Profile

Version 1.1

Danish Digitisation Agency
January 2019

# Content >

# Document History

| Version | Date | Initials | Changes |
|---|---|---|---|
| 1.0 | 10-05-2010 | TG/SPN | Document published |
| 1.0.1 | 22-09-2010 | TG | Added the possibility of scoping an access right to a CPR number which is relevant in delegation scenarios. |
| 1.1 | 31-05-2017 | TG | The Intermediate model has been extended with the possibility to define <Constraint> elements describing data restrictions.<br><br>References to standards have been updated to reflect current versions.<br><br>Clarifications to the various chapters (non-normative). |

# Purpose

This document contains a profile of [OIOSAML] and [OIOIDT] defining how basic user privileges can be expressed as attributes in SAML Assertions – in other words a "basic privilege profile". Thus, this profile should be considered an addendum to the [OIOSAML] and [OIOIDT] profiles.

Two different representations of privileges are profiled in this document:

a) A simple model where a user (SAML Subject) is associated with a list of privileges.

b) An intermediate model with richer semantics where a user is associated with privileges which are limited to a defined scope and optionally a set of contraints.

Privileges in this profile are represented as URIs and are often associated with *roles* to achieve role-based access control (RBAC).

Deployers of this profile can choose between the two representations according to their requirements and needs.

This profile does not specify how Identity Providers (token issuers) and Service Providers (token consumers or relying parties) agree on the semantics, rules and governance of the privileges being exchanged or how privileges are administered / assigned to users – it strictly focuses on how to represent privileges in OIOSAML Assertions. In other words, the profile applies to run-time scenarios where a user seeks access to a service by presenting a token containing certain attributes (privileges).

## Background

Due to the wide adoption of the OIOSAML and OIO IDWS standards from the Danish Digitisation Agency an increasing number of service providers in the public sector use SAML assertions as a means to achieve federated access for users to applications and services.

The OIO SAML Web SSO profile does not specify how privileges are represented, and this may potentially create interoperability issues. The goal of this profile is to define a common representation for simple privileges that can be used across the Danish public sector to advance interoperability and simplify implementation.

The profile build on concepts developed for the Virk.dk portal which has defined an approach to specify, administer and convey privileges (corresponding to the intermediate model). Another foundation is the Danish Reference architecture for identity- and access management "Referencearkitektur for brugerstyring 1.0" [RefArk] which defines common terms, architectural building blocks and principles.

The extension of the profile with Constraint elements to define data restrictions on roles has been successfully implemented by KOMBIT as a key element in their framework architecture (*den fælleskommunale rammearkitektur*).

# Terminology and models

In this profile, it is assumed that the token issuer is a SAML 2.0 Identity Provider or WS-Trust "Security Token Service" who has authenticated a user and wants to include privilege information in the SAML Assertion that is going to be issued to a Service Provider who consumes the token and provides the user access based on its content.

It is outside the scope of the profile how privileges are assigned to users (e.g. user administration), how the issuing Identity Provider or Security Token Service knows what privileges the user has been assigned, or what privileges the recipient of the Assertion (application or service) requires (e.g. definition and enforcement of access control policies). In other words, this profile defines a transfer format for privileges independent of the context where privileges are used.

A *privilege* in this profile is simply a unique string identifier (URI) defined by an organization with local semantics. Privileges may also be shared across organizations but this is outside the scope of this profile. A privilege is represented by a URI and is required to be unique, preferably by using the domain name of the defining organization as a prefix, e.g.:

- `urn:dk:mydomain:myapp:myprivilege123` *or*
- `http://mydomain.dk/myapp/myprivilege123`

Note that privileges are identifiers only and need not point to any ressource.

Privilege URIs are normally defined by the application or service that receives a SAML assertion issued by an Identity Provider or Security Token Service. It is a simple identifier that can be understood as a role name, a group name or something else. In some deployments, privileges are referred to as *system roles* (to clarify that they are application-defined roles) or *system profiles* – to avoid confusion with any organizational roles the user may be associated with. Thus, the above definition and representation of privileges is general and can encompass a variety of situations.

In many deployments, privileges are assigned to users by first including individual users in a user group, then assigning abstract / organizational roles to the groups, and finally mapping the roles to privileges in specific applications. E.g.: *user "Kurt Jensen" is a member of the "accountants_head_office" group which is assigned the role "tax_reporter", which is mapped to the privilege "urn:dk:mydomain:app:submit_tax_report"* defined for a specific application.

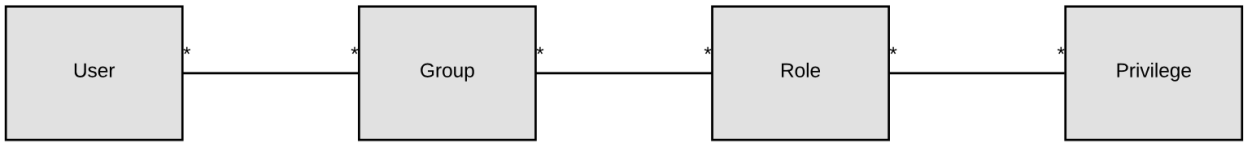


**Figure 1: Example data model**

As stated above, the underlying model for how users are assigned privileges is outside the scope of this profile and may vary with implementations. This profile focuses on the net result which is the set of resulting privileges which are listed in a SAML Assertion.

## Model 1 (Simple model)

In the simple model the user is simply associated with a list of privileges. The privileges are not scoped or restricted via their representation in the SAML Assertion.

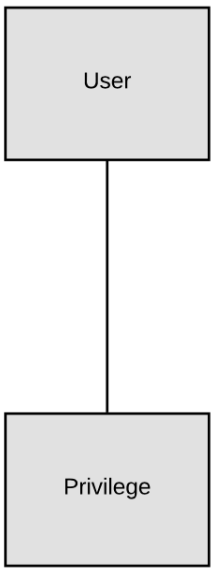


**Figure 2: Simple model**

## Model 2 (Intermediate model)

The intermediate model is a bit more advanced and can express richer access right models. The user is still associated with a set of privileges but each privilege is restricted according to a *scope*. The scope element limits the privilege to a certain context – for example to an organizational unit within the user's company. E.g. *the user "Kurt Jensen" has the privilege "urn:dk:mydomain:app:submit_tax_report" for the scope "company number=20182838".*

This profile defines only scope identifiers for organizational units and natural persons but the model is open and can be extended with other types of scope.

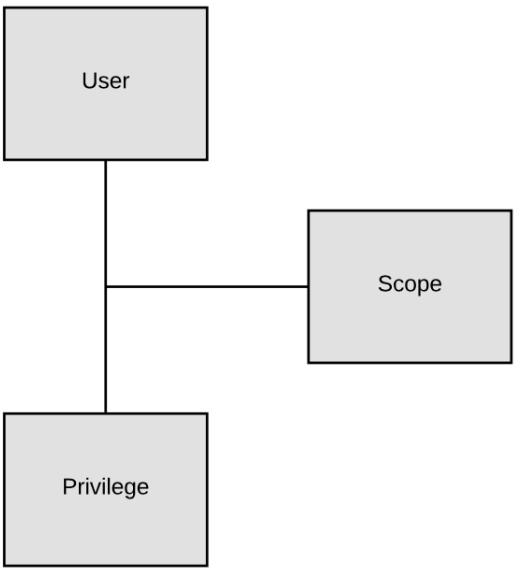


**Figure 3: Intermediate model with scopes**

## Delegation

In many scenarios users can be delegated privileges from other users or organizations. For example, an external auditor (company) can be delegated rights to submit tax reports on behalf of a client. The above models do not express the delegation relation explicitly, but the intermediate model can represent the resulting privileges using the scope element (i.e. the auditor gets the "tax report" privilege with the scope of client's company identifier).

## Model 3 (Intermediate model with constraints)

In some scenarios, a static privilege/role model does not provide sufficient granularity to express the desired access rights of users. Therefore, this profile allows Constraints to be specified which further limit the scope of a privilege. Constraints are essentially key/value pairs where the key identifies the constraint and the value specifies the restriction and they are always applied to a User-Privilege relation (assignment). This construct provides the possibility to use a two-dimension model rather than a one-dimensional model. As an example, privileges can be used to specify roles being actions, functions or methods that can be accessed at a particular system, where constraints can be used to specify the allowed *data objects* where these roles can be applied.

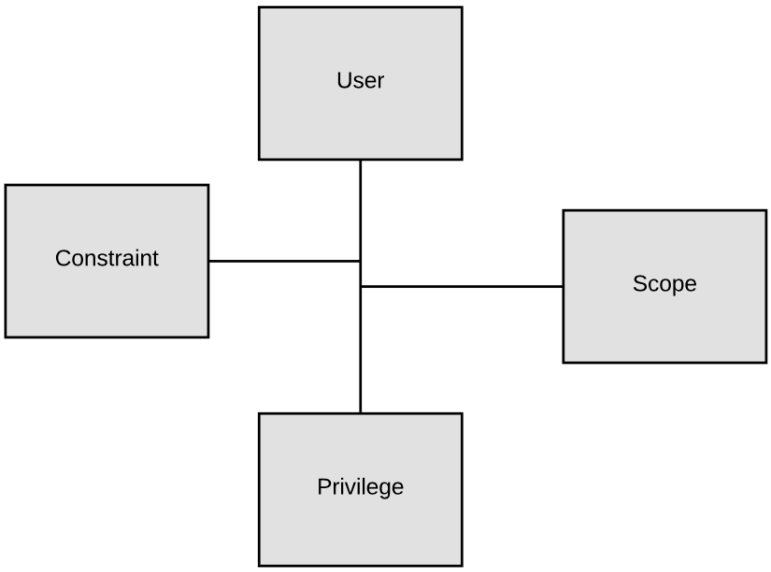


**Figure 4: Intermediate model with Scope and Constraints**

Example:

*A user could have the privilege "urn:dk:mydomain:app:view_case" assigned with the constraint (CaseID=1256153A) associated. This intended result would be that the user has a role which grants access to view a case – but restricted to a specific case with the specified ID – as opposed to viewing all cases.*

Note that a Scope restriction could be modeled as a Constraint, but constraints are kept for reasons of backwards compatibility with the existing and widely deployed intermediate model.

# Representation and processing of Privileges (normative)

This chapter contains normative requirements for representation and processing of privileges in the above two models.

## Privilege

A privilege MUST be represented by a URI that is unique such that name collisions are avoided. It is RECOMMENDED to use the domain name of the defining organization as a prefix e.g.:

- `urn:dk:mydomain:myapp:myprivilege123` *or*
- `http://mydomain.dk/myapp/myprivilege123`

## User

The user to whom the privilege applies MUST be the `<Subject>` of the SAML Assertion. It can both be a human user or a system user.

## Scope

Scope of a privilege MUST (when used) be represented by a URI that is unique such that name collisions are avoided.

This profile defines the following scope URIs (relevant for Danish companies) representing CVR numbers, production unit identifiers ("P-numbers") and SE numbers and CPR numbers:

- `urn:dk:gov:saml:cvrNumberIdentifier:<cvr_number>`
- `urn:dk:gov:saml:productionUnitIdentifier:<p_number>`
- `urn:dk:gov:saml:seNumberIdentifier:<SE_number>`
- `urn:dk:gov:saml:cprNumberIdentifier:<cpr_number>`

A specific scope is indicated by adding the relevant scope value (e.g. CVR number) as a suffix. Scopes can be used to indicate that the user has been assigned access rights on behalf of another person (indicated by the scope) which is useful in delegation scenarios. For example, if the Subject of the Assertion is associated with CVR number *A*, but the Scope of a privilege *P* is set to CVR number *B*, this means that the user in organization *A* has been delegated privilege P on from the organization *B*.

Deployers MAY define other scope URIs provided they are unique. In this case it is RECOMMENDED to use the domain name of the defining organization as a prefix.

## Model 1

In the first model the user is simply associated with a list of privileges. The privileges MUST be represented as multiple `AttributeValue` elements containing the corresponding privilege URIs in an attribute with the name `dk:gov:saml:attribute:Privileges_simple`.

Example:

```
<saml:Attribute FriendlyName="Privileges"
   Name="dk:gov:saml:attribute:Privileges_simple"
   NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:basic">
   <saml:AttributeValue xsi:type="xs:string">
          urn:dk:some_domain:myPrivilege1A
   </saml:AttributeValue>
   <saml:AttributeValue xsi:type="xs:string">
          urn:dk:some_domain:myPrivilege1B
   </saml:AttributeValue>
   <saml:AttributeValue xsi:type="xs:string">
          urn:dk:some_domain:myPrivilege1C
  </saml:AttributeValue>
</saml:Attribute>
```

Note that the attribute values are simple strings and that no XML elements in foreign name spaces are used.

## Model 2

The second model uses two steps to encode the privileges. First, privileges URIs MUST wrapped in `<bpp:Privilege>` elements. Privileges with the same scope MUST be collected in groups via a `<bpp:PrivilegeGroup>`, and the groups MUST subsequently be enumerated in a `<bpp:PrivilegeList>` element as defined below.

Example:

```
<?xml version="1.0" encoding="UTF-8"?>
<bpp:PrivilegeList
   xmlns:bpp="http://itst.dk/oiosaml/basic_privilege_profile"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" >
   <PrivilegeGroup Scope="urn:dk:gov:saml:cvrNumberIdentifier:12345678">
       <Privilege>urn:dk:some_domain:myPrivilege1A</Privilege>
       <Privilege>urn:dk:some_domain:myPrivilege1B</Privilege>
   </PrivilegeGroup>
   <PrivilegeGroup Scope="urn:dk:gov:saml:seNumberIdentifier:27384223">
       <Privilege>urn:dk:some_domain:myPrivilege1C</Privilege>
       <Privilege>urn:dk:some_domain:myPrivilege1D</Privilege>
   </PrivilegeGroup>
</bpp:PrivilegeList>
```

In the second step, the resulting `<PrivilegeList>` element MUST first be converted to bytes using the UTF-8 encoding and then base64-encoded to get a string. The resulting base64 string MUST then be embedded as a SAML attribute with the name `dk:gov:saml:attribute:Privileges_intermediate` as shown below:

```
<saml:Attribute FriendlyName="Privileges"
   Name="dk:gov:saml:attribute:Privileges_intermediate"
   NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:basic">
   <saml:AttributeValue xsi:type="xs:string">
      <base64 encoded value>
  </saml:AttributeValue>
</saml:Attribute>
```

Note that the encoded attribute value is a simple string and that no XML elements in foreign name spaces are used; thus, standard SAML implementations should be able to process the Assertion without problems.

## Model 3

The third model is a variant over Model 2, where Constraint elements can be added within a PrivilegeGroup to restrict privileges in the group. Constraint elements have a Name attribute (whose value MUST be a URI) and a value:

Example:

```
<?xml version="1.0" encoding="UTF-8"?>
<bpp:PrivilegeList
   xmlns:bpp="http://itst.dk/oiosaml/basic_privilege_profile"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" >
   <PrivilegeGroup Scope="urn:dk:gov:saml:cvrNumberIdentifier:12345678">
       <Constraint Name="urn:dk:kombit:KLE">25.*</Constraint>
       <Constraint Name="urn:dk:kombit:sensitivity">3</Constraint>
        <Privilege>urn:dk:kombit:system_xyz:view_case</Privilege>
   </PrivilegeGroup>
</bpp:PrivilegeList>
```

In the above example, the user is assigned the "view_case" privilege, but this is constrained to cases identified with KLE[1]="25.*" and Sensitivity="3", so only cases of a certain type **and** sensitivity can be viewed by the user. In other words, the Constraint elements work as filters.

The second step (base64 encoding) is identical to model 2 and therefore not repeated here.

## Processing rules

All the usual OIOSAML processing rules apply to Assertions that contain privilege attributes, including validating that the signer of the assertion is trusted etc.

A SAML Assertion MUST only use one of the above representations of privileges (simple or intermediate) and MUST not include multiple privilege attributes.

If the user corresponding to the `<Subject>` of the Assertion has no privileges known to the Assertion Issuer, privilege attributes MUST be omitted from the Assertion.

Service Providers who do not understand privilege attributes MAY ignore them entirely.

Service Providers who require privileges as part of their access control policy MUST do the following:

1. Validate the privilege attributes and ensure that all required privileges are present.
2. Unknown privilege URIs MAY be ignored.
3. If a privilege is associated with a scope URI that is not understood, the Service Provider MUST ignore all privileges associated with that scope (i.e. act as if the corresponding privilege group was not present at all).
4. Constraint element (when present) MUST be treated as a restriction on the Privileges in the same group, thus filtering the objects where the privilege can be applied. Missing Constraint elements means that the Privileges are not restricted.
5. When multiple Constraint elements are specified within the same PrivilegeGroup, there is an implied logical AND between them: therefore, the privileges in the group MUST only be applied to objects meeting ALL the Constraints of the PrivelegeGroup.

[1] KLE is the Danish taxonomy for cases (KL Emnesystematik), and 25.* is the main group for tax cases.

6. If a Constraint Name or value is not understood, the entire PrivilegeGroup MUST be ignored (according to the ´default to secure' principle).

# Schema (normative)

Below is an XML schema defining the XML elements for the intermediate model.

```
<?xml version="1.0" encoding="UTF-8"?>
<schema
   xmlns="http://www.w3.org/2001/XMLSchema"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   xmlns:bpp="http://itst.dk/oiosaml/basic_privilege_profile"
   targetNamespace="http://itst.dk/oiosaml/basic_privilege_profile">

  <element name="PrivilegeList" type="bpp:PrivilegeListType"/>

  <complexType name="PrivilegeListType">
    <sequence>
      <element name="PrivilegeGroup" type="bpp:PrivilegeGroupType" minOccurs="1"
        maxOccurs="unbounded"/>
    </sequence>
  </complexType>

  <complexType name="ConstraintType">
     <simpleContent>
        <extension base="string">
           <attribute name="Name" type="anyURI" use="required"/>
         </extension>
     </simpleContent>
  </complexType>

  <complexType name="PrivilegeGroupType">
     <sequence>
          <element name="Privilege" type="anyURI" minOccurs="1" maxOccurs="unbounded"/>
          <element name="Constraint" type="bpp:ConstraintType" minOccurs="0" maxOccurs="unbounded"/>
     </sequence>

     <attribute name="Scope" type="xsd:string" use="required"/>
  </complexType>
</schema>
```

# Appendix A: Architectural Decisions

This chapter contains a number of architectural decisions which provide the rationale behind important choices made in the profile.

## AD 1: Explicit representation of delegation relationship

| | |
|---|---|
| **Problem** | Should the profile explicitly specify how delegation of access rights is represented? (e.g. that a user is acting on behalf of another user) |
| **Assumptions** | Service providers must be able to account for which users that access their systems and data and why access was granted. |
| **Alternatives** | 1. Represent the on-behalf-of relation explicitly in the token.<br>2. Leave delegation relationships out of the token and only represent the privileges that are the result of delegation. |
| **Analysis** | The way delegation is modeled and used will probably vary among services and business domains. Thus, it can be hard to define a model and syntax that is generic across the Danish public sector. Further, no useful standard or profile is known which means that a definition would be proprietary and require additional customization for all federation members to support.<br>Without explicit representation of the delegation relation, the service provider can log the SAML assertion received from Identity Provider / STS where the identity of the user and the privileges are stated. Based on that he can account for the access granted to his services and data. Should a dispute or security incident be investigated further, the service provider can refer to the Identity Provider / STS who must be able to account for how the assertion was issued: which user authenticated using which credentials, and what were the privileges assigned to the user (including any delegations) at the time of token issuance. |
| **Decision** | Leave delegation relationships out of the token and only represent the privileges that are the result of delegation. |

## AD 2: Base64 encoding of XML in intermediate model

| | |
|---|---|
| **Problem** | Should the attribute value containing the privileges be base64 encoded in the intermediate model? |
| **Assumptions** | |
| **Alternatives** | 1. Base 64 encode XML.<br>2. Don’t encode XML. |
| **Analysis** | Previous experience in the Danish federation suggests that some SAML implementation may struggle with complex XML values (i.e. values that are not simple strings) or values that contain XML elements in non-SAML namespaces (as is the case for model 2).<br>These factors suggest that encoding of the attribute value will have the biggest chance of not breaking existing SAML implementations. |

| | |
|---|---|
| | On the other hand, encoding and decoding introduces an extra processing overhead, and the model will not be fully backwards compatible with Virk.dk's existing model. |
| **Decision** | Base64 encode XML in intermediate model. |

# Appendix A: References


**[OIOSAML]** "OIO Web SSO Profile version", Digitaliseringsstyrelsen.
https://www.digitaliser.dk/group/42063/resources

**[OIOIDT]** "OIO SAML Profile for Identity Tokens 1.1", Digitaliseringsstyrelsen,
https://digitaliser.dk/resource/3457606

**[SAMLCore]** "Assertion and Protocols for the OASIS Security Assertion Markup Language 2.0", OASIS Standard.
http://docs.oasis-open.org/security/saml/v2.0/saml-core-2.0-os.pdf

**[RefArk]** "Referencearkitektur for Brugerstyring" – Digitaliseringsstyrelsen, 2017.
https://www.digst.dk/Arkitektur-og-data/It-arkitektur/Brugerstyring

<