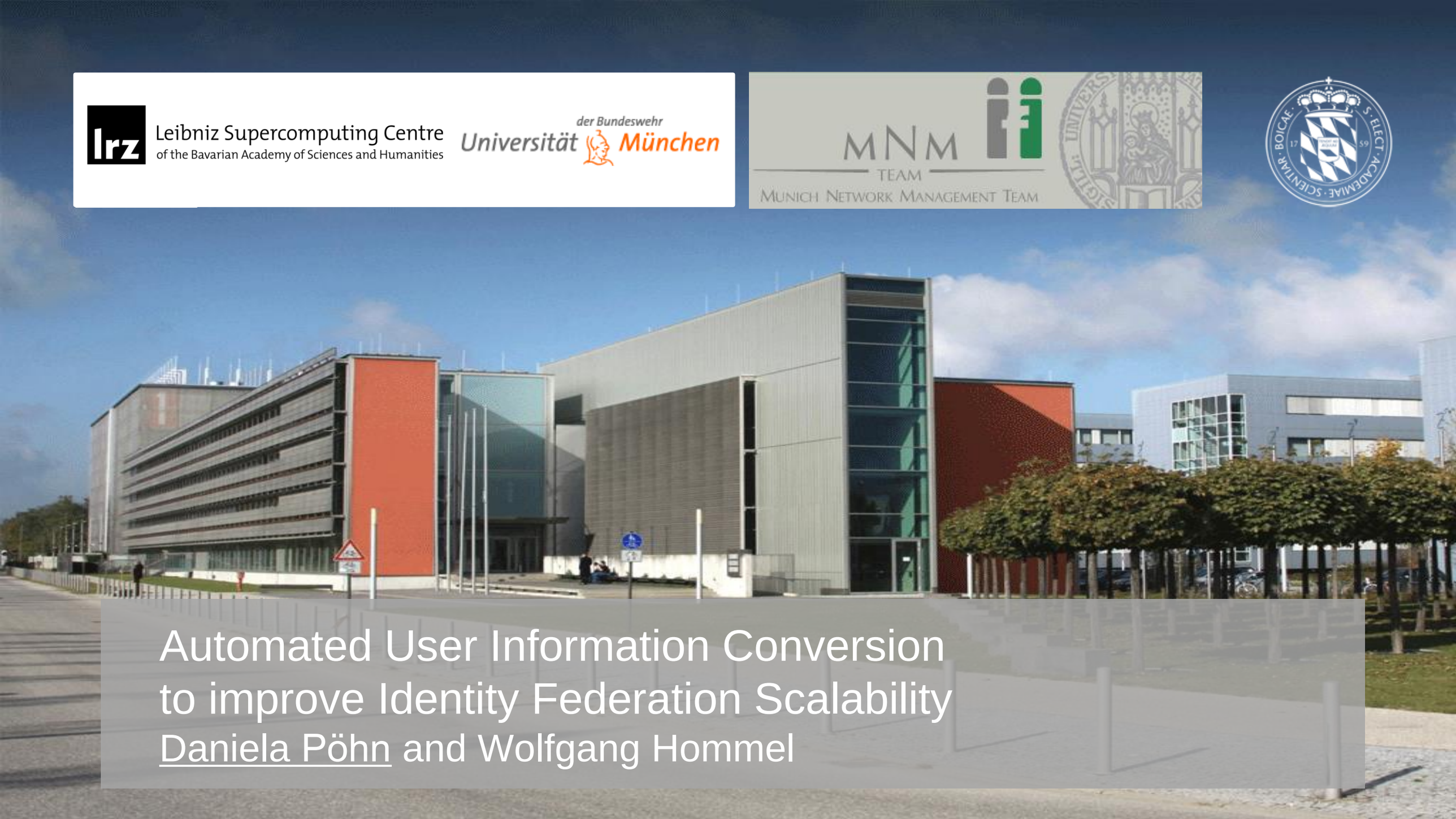


Leibniz Supercomputing Centre
of the Bavarian Academy of Sciences and Humanities

Universität der Bundeswehr München

MNM TEAM
Munich Network Management Team

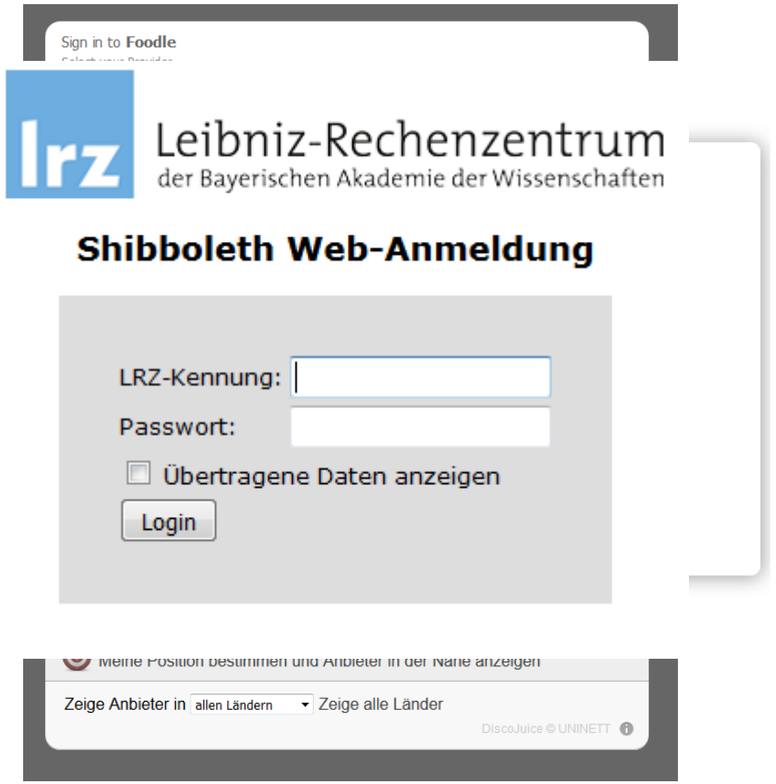
Automated User Information Conversion to improve Identity Federation Scalability

Daniela Pöhn and Wolfgang Hommel

Agenda

- Introduction and Motivation
- Generic Conversion Rule Repository
  - Functionality
  - Workflows
  - Architecture
  - Example
- Conclusion and Outlook

Introduction and Motivation



Introduction and Motivation

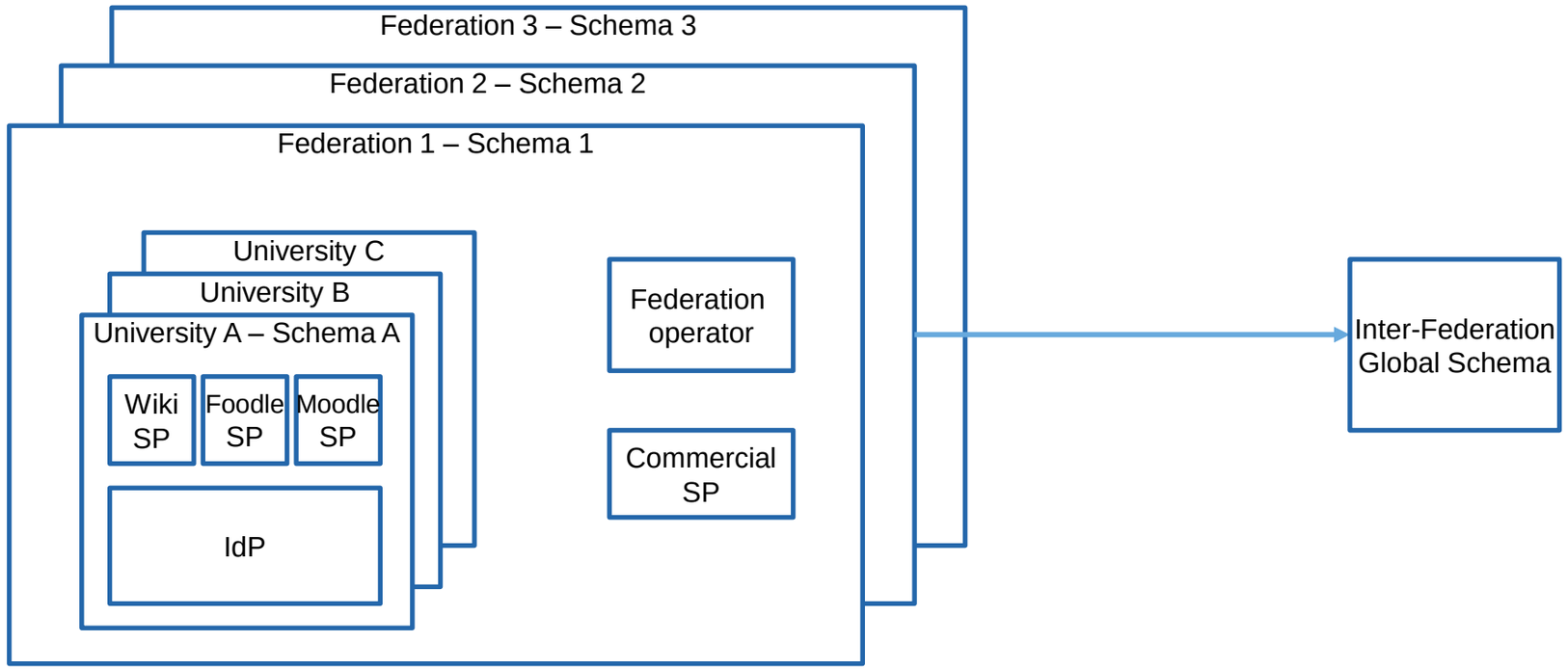
- Problem: User information (attribute) missing
- Attribute: `displayName` or `cn`
- Inform organisation
- Which attribute?
- Update configuration manually
- Waiting time

Introduction and Motivation

Current situation in R&E:

- Local Identity & Access Management (I&AM):
  - LDAP or relational database
  - user information (attributes)
- Federated Identity Management (FIM):
  - Collaborations
  - SAML
  - Identity Provider (IDP)
  - Service Provider (SP)
  - Signed Metadata, aggregated and pre-shared
  - Schema: Semantics and syntax attributes
- Inter-Federated Identity Management (IFIM)

Introduction and Motivation



Introduction and Motivation

Service Providers:

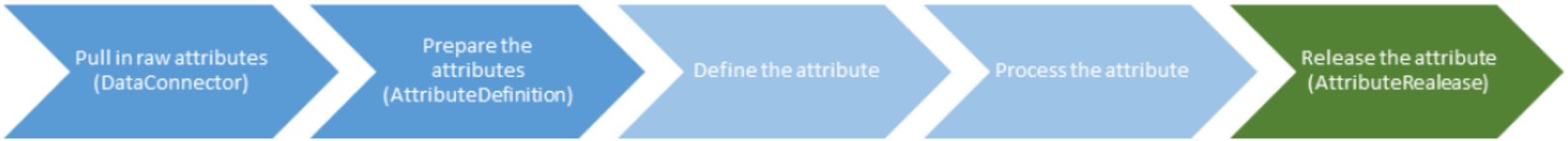
- Can state requested attributes in metadata.
- Can also send a request.
- FriendlyName, Name as OID, NameFormat, and isRequired

```
<RequestedAttribute FriendlyName="eduPersonPrincipalName"
Name="urn:oid:1.3.6.1.4.1.5923.1.1.1.6"
NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri" isRequired="true"/>
```

Introduction and Motivation

Identity Provider:

1. Fetch raw user data into IDP software (`DataConnector`)
2. Define attributes (`AttributeDefinition`)
3. Filter attributes (`AttributeFilter`)
4. Send attributes (`AttributeRelease`)



Introduction and Motivation

Typical conversion rules:

- Renaming, e.g., from `DateofBirth` to `schacDateOfBirth`,
- Merging, e.g., `sn` and `givenName` to `displayName`
- Splitting, e.g. `cn` to `givenName`
- Transforming, e.g., different date formats – `dd.mm.yyyy` to `mm-dd-yyyy`

```
<resolver:AttributeDefinition id="eduPersonPrincipalName" xsi:type="Scoped"
xmlns="urn:mace:shibboleth:2.0:resolver:ad" scope="lrz.de"
sourceAttributeID="uid">
    <resolver:Dependency ref="simauth" />
    <resolver:AttributeEncoder xsi:type="SAML1ScopedString"
    xmlns="urn:mace:shibboleth:2.0:attribute:encoder"
    name="urn:mace:dir:attribute-def:eduPersonPrincipalName" />
    <resolver:AttributeEncoder xsi:type="SAML2ScopedString"
    xmlns="urn:mace:shibboleth:2.0:attribute:encoder"
    name="urn:oid:1.3.6.1.4.1.5923.1.1.1.6" friendlyName="eduPersonPrincipalName"
    />
</resolver:AttributeDefinition>
```

Introduction and Motivation

- Need to be applied **manually** by the IDP administrator.
- Waiting time for users.
- Not scalable
- Shibboleth and other FIM software have pre-defined conversion types.
- Pre-defined conversion types vary.

→ Generic conversion rule repository

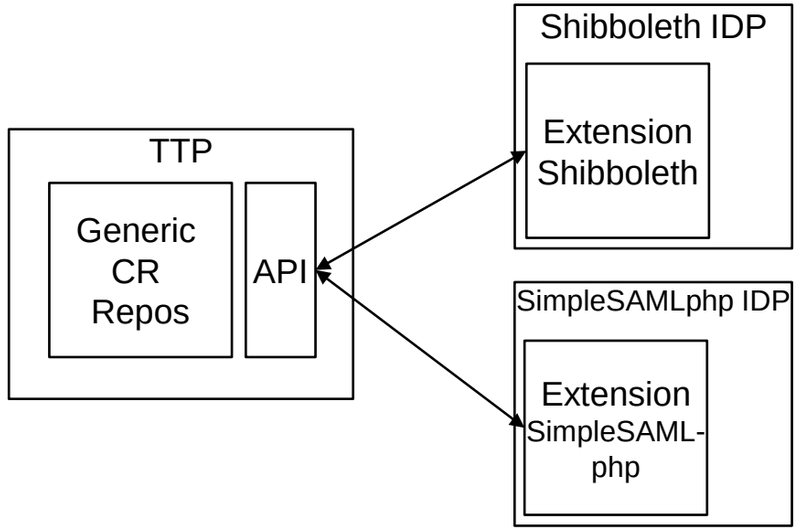
→ Translated into software-specific rules.

Generic Conversion Rule Repository - Functionality

- Known: available and needed attributes

→ Extension searches for conversion rule

- Generic conversion rule at TTP
- Adapted for the FIM software
- Locally integrated



→ Specific and generic conversion rules can be re-used

→ Speeding up setup between IDPs and SPs

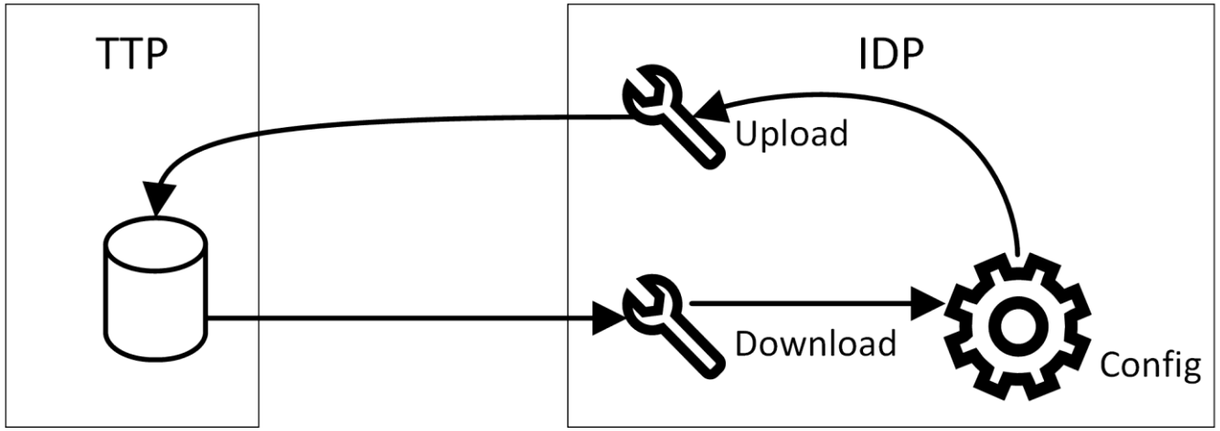
Generic Conversion Rule Repository - Functionality

Workflow:

Known: IDP attributes and needed SP attributes

1. Extension detects that IDP does not have necessary attributes for SP.
2. Extension queries TTP.
3. a. If generic conversion rule is found, rule is downloaded and transformed.
   b. Complex conversion rule with scripts is stored IDP software specific and manually downloaded.
   c. If no conversion rule is found, IDP operator writes new conversion rule.
4. After downloading conversion rule, the generated specific rule is integrated into IDP's local configuration.
5. User can make use of service without problems.

Generic Conversion Rule Repository - Workflow



- TTP stores generic conversion rules in database.
- Generic rule is downloaded, converted, and inserted into configuration.
- If rule is written, it is translated into generic format and uploaded to the TTP.
- Specific rules are also stored in database.

Generic Conversion Rule Repository - Architecture

Database:

- `ConversionRule`:
  Conversion from one or more attributes into another attribute.
- `ConversionKeyword`:
  Inserts keywords for specific conversion rules.
- `ConversionAttribute`:
  Information about source and target attributes for a conversion rule.

Generic Conversion Rule Repository - Architecture

Generic format of simple conversion rules

Shibboleth uses pre-defined operations, e.g.:

- Renaming by mapping of attributes.
- Splitting and other definitions with regular expressions.
- Merging by template attribute definition (Velocity template language).
- Scoping by scoped attribute definition.
- Principal name by principal name attribute definition.

Mapping of different pre-defined operations

→ Generic simple conversion rules

Generic Conversion Rule Repository - Architecture

Following information needed:

- sort of conversion,
- source attributes,
- target attribute, and
- additional information, like regex.

Keywords to apply specific conversion rules:

- `source`,
- `target`,
- `targeturn1`,
- `targeturn2` as well as the transformations
- `regex` respectively `pattern` and
- `conversion`.

Generic Conversion Rule Repository - Architecture

Generic conversion rule:

```
source={source1, source2, ...};
transformation = [renaming, merging, regex, conversion];
target={target, targeturn1, targeturn2};
source(transformation) => target;
```

Renaming:

```
source;
transformation = renaming;
target={target, targeturn1, targeturn2};
```

Generic Conversion Rule Repository - Architecture

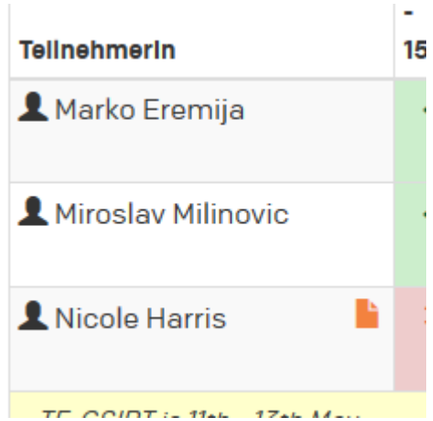
- FIM software specific templates
- Keywords filled with values from generic conversion rule repository
- Federations can operate such a repository

```
<resolver:AttributeDefinition xsi:type="Script“
xmlns="urn:mace:shibboleth:2.0:resolver:ad" id="{{target}}">
    <resolver:Dependency ref="{{source1}}"/>
    <resolver:Dependency ref="{{source2}}"/>
    <resolver:AttributeEncoder xsi:type="SAML1String"
                           xmlns="urn:mace:shibboleth:2.0:attribute:encoder"
                           name="{{targetUrn1}}" />
    <resolver:AttributeEncoder xsi:type="SAML2String"
                           xmlns="urn:mace:shibboleth:2.0:attribute:encoder"
                           name="{{targetUrn2}}"
                           friendlyName="{{target}}" />
...
</resolver:AttributeDefinition>
```

Generic Conversion Rule Repository - Example

```
source={gecos};
transformation = renaming;
target={displayName, targeturn1, targeturn2};

<resolver:AttributeDefinition xsi:type="Simple"
 xmlns="urn:mace:shibboleth:2.0:resolver:ad"
 id="displayName" sourceAttributeID="gecos">
  <resolver:Dependency ref="{{ source|resource }}" />
    <resolver:AttributeEncoder xsi:type="SAML1String"
     xmlns="urn:mace:shibboleth:2.0:attribute:encoder"
     name="{{ targeturn1 }}"/>
    <resolver:AttributeEncoder xsi:type="SAML2String"
     xmlns="urn:mace:shibboleth:2.0:attribute:encoder"
     name="{{ targeturn2 }}"
     friendlyName="displayName" />
</resolver:AttributeDefinition>
```



Conclusion and Outlook

- Generic conversion rule repository improves Shibboleth repository
- Improves Proof-of-Concept implementation of GÉANT TrustBroker
- Allows re-use of conversion rules
- Independent of FIM software
- Speeds up IDP-SP setup
- Reduces waiting time for users

Next steps:

- Test concept with different parties
- Improve and extend repository
- How to generalize scripts or more complex conversion rules?