



13-03-2015

Open Call Deliverable OCJ-DS4.1.1

GÉANT-TrustBroker Implementation with Documentation (GÉANT-TrustBroker)

Open Call Deliverable OCJ-DS4.1.1

Grant Agreement No.: 605243
Activity: NA1
Task Item: 10
Nature of Deliverable: R (Report)
Dissemination Level: PU (Public)
Lead Partner: BADW-LRZ
Document Code: GN3PLUS14-1292-56
Authors: Daniela PÖHN, Michael GRABATIN, Stefan METZGER, David SCHMITZ, Wolfgang HOMMEL

© GEANT Limited on behalf of the GN3plus project.

The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7 2007–2013) under Grant Agreement No. 605243 (GN3plus).

Abstract

This deliverable first documents the usage as well as installation and configuration procedures of the three GÉANT-TrustBroker project implementation parts: The GNTB core service, the Shibboleth IDP extension, and the Shibboleth SP extension. It refers to the proof-of-concept testbed, which is the basis for the GÉANT-TrustBroker demonstrator that was presented at the GÉANT Symposium 2015, as an example and is intended to foster further deployments in GN4P1 to continue development and prepare for pilot operations. The source code is published via the TrustBroker repository on svn.geant.net. Afterwards, this document discusses how the project met its original requirements, has been completed as per the project proposal submitted, and analyses the project impact.

Document Revision History

Version	Date	Description of change	Person
1	14-02-15	First draft issued	Pöhn, Metzger, Grabatin, Hommel
2	04-03-15	Minor improvements	Grabatin, Pöhn, Metzger, Hommel
3	06-03-15	Various improvements	Hommel
4	09-03-15	Review	R. Poortinga-van Wijnen
5	10-03-15	Improvements based on review	Hommel
	10-03-15	Approved	R. Poortinga-van Wijnen

Table of Contents

Executive Summary	1
1 Introduction	2
2 Using GÉANT-TrustBroker	4
2.1 GNTB management workflow	5
2.2 GNTB core workflow (DAME)	8
2.3 GNTB conversion rule workflow	12
2.4 GNTB complete core workflow	14
3 GÉANT-TrustBroker implementation overview	15
3.1 GNTB extension for Shibboleth IDP	16
3.2 DAME extension for Shibboleth SP	17
3.3 Getting the source code	18
4 GÉANT-TrustBroker installation and configuration	19

4.1	GNTB core service	19
4.2	GNTB extension of the Shibboleth IDP	22
4.3	DAME extension of the Shibboleth SP	25
4.4	Modification of the Shibboleth Embedded Discovery Service	27
5	GÉANT-TrustBroker project assessment	29
5.1	Comparison with the objectives of the Open Call text	29
5.2	Comparison with the original project proposal	30
5.3	Discussion of project impact	31
6	Summary and outlook	33
	References	43
	Glossary	43

Table of Figures

Figure 2-1: Demonstrator setup	5
Figure 2-2: Screenshot of the GNTB web-interface for adding a new provider	6
Figure 2-3: Screenshot of the web-interface for uploading metadata	8
Figure 2-4: DAME core workflow	9
Figure 2-5: Screenshot of the SP's Embedded Discovery Service showing the option of using GNTB	9
Figure 2-6: Screenshot of the IDP selection at the GNTB core service	10
Figure 2-7: Screenshot of the collaboration service without the skypeID	11
Figure 2-8: Screenshot of the web-interface for conversion rules	13
Figure 2-9: Screenshot of the collaboration service after automatic conversion of the skypeID	14
Figure 3-1: Overview of the implementation of the IDP extension	16
Figure 3-2: Overview of the inter-process communication used in the Shibboleth SP implementation	18

Executive Summary

The GÉANT-TrustBroker open call project enables the fully automated, user-triggered, on-demand establishment of technical trust between service providers (SPs) and identity providers (IDPs) through dynamic automated SAML metadata exchange (DAME). It includes basic AccountChooser (Where Are You From/WAYF) functionality for users and minimizes the system-administrative workload for IDP administrators by enabling the sharing and re-use of user attribute conversion rules. GÉANT-TrustBroker is intended for scenarios in which the a-priori - e.g. federation- or eduGAIN-based - exchange of SAML metadata is neither practical nor scalable. GÉANT-TrustBroker is not intended for scenarios where a-priori exchange of metadata is mandatory for non-technical aspects, such as contract-based trust building between IDP and SP.

Roughly one third of the project was spent on implementing a proof of the GÉANT-TrustBroker concepts as they have been documented in M.1.1.1, D.2.1.1, D.2.2.1, and M.3.1.1. The results have been presented as a demonstrator during the 2015 GÉANT Symposium; the development and testbed infrastructure are operated locally at BADW-LRZ, but the source code of the implementation is available publically through GÉANT's Subversion-based software development repository.

This deliverable is primarily intended as accompanying documentation of the implementation, therefore describing its usage as well as the installation and configuration procedures of the GÉANT-TrustBroker core service and the extensions required for Shibboleth IDPs and SPs. To make this documentation more hands-on and tangible, the demonstrator setup is used as an example; the goals are 1) to enable developers from other project partners in GN4 to quickly set up similar development and testing environments and 2) to create a basis for software packaging and documentation as it will be required for preparing pilot operations in GN4.

As this document is also the final deliverable of the open call project, it includes a discussion of how the original requirements of the open call text and the open call projects' overall scope have been met, analyses to which degree the project achieved its original plans as documented in the description of work and project proposal, and draws a conclusion regarding the impact of the project and its results.

1 Introduction

This deliverable documents the proof-of-concept implementation developed during the GN3plus open call project GÉANT-TrustBroker. The source code is available from <http://svn.geant.net/GEANT/TrustBroker/>.

Section 2 shows how the implementation can be used from the perspectives of IDP and SP administrators as well as users; it is intended for a broader audience interested in the GÉANT-TrustBroker project results. Then, Sections 3 and 4 go into the technical details of the GÉANT-TrustBroker implementation, including documentation for installing and configuring the core service as well as the Shibboleth IDP and SP extensions; they are intended for GÉANT developers who will continue the work on GÉANT-TrustBroker in GN4. Afterwards, Section 5 evaluates the project's results by comparing them with the open call text and the original project proposal / description of work; it also discusses the impact of the project. Section 6 concludes with a summary and a short outlook to what has been planned for the first phase of the GN4 project. Longer code and configuration file excerpts have been moved to appendix A to improve readability.

This document uses the GÉANT-TrustBroker demonstrator, which has been presented at the GÉANT Symposium 2015, as a hands-on example. A demo video is available in the Documents section of the project's GÉANT Intranet website and it is recommended to watch it in order to set expectations of what following the installation, configuration, and usage steps described in this document leads to.

To avoid content duplication, readers of this document are supposed to be roughly familiar with the concepts and design of GÉANT-TrustBroker, which have been documented in

- M.1.1.1: GÉANT-TrustBroker core functionality, requirements, and workflows
- D.2.1.1: GÉANT-TrustBroker technical specification, including API and data model design
- D.2.2.1 or the IETF Internet-Draft on the GÉANT-TrustBroker core protocol [1], called DAME (Dynamic Automated Metadata Exchange).

As the GÉANT-TrustBroker implementation is an extension of the open source software Shibboleth, readers who want to validate the project results by re-creating the demonstrator setup in their own infrastructure are also assumed to have basic Shibboleth installation and administration experience, namely regarding the Shibboleth IDP, Shibboleth SP, Shibboleth Centralized Discovery Service, and Shibboleth Embedded Discovery Service.

The two primary distinguishing features of GÉANT-TrustBroker are the dynamic automated Security Assertion Markup Language (SAML) metadata exchange and the user attribute conversion rule approach to handle heterogeneous federation schemas. Therefore, readers interested in technical details should also know about SAML metadata and SAML attributes as well as how Shibboleth, especially the IDP implementation, handles them (metadata providers, attribute filters, and so on).

The GÉANT-TrustBroker team at BADW-LRZ would like to thank the technical project coordinator, Remco Poortinga-van Wijnen, and the Open Calls team at DANTE/the GÉANT Association, especially Annabel Grant and Nathalie McKenzie, for their excellent and uncomplaining support, as well as everyone from GN3plus JRA3 & SA5, REFEDS, and IETF who gave encouraging feedback and therefore contributed to the improvement of GÉANT-TrustBroker.



2 Using GÉANT-TrustBroker

This section shows how GÉANT-TrustBroker can be used in practice. It uses the open call project's demonstrator setup as a hands-on example; Sections 3 and 4 of this document provide the necessary information to set up such a scenario in one's own infrastructure.

The acronym DAME refers to the IETF Internet-Draft about Dynamic Automated Metadata Exchange, which is the vendor-, project-, and implementation-neutral term for the GÉANT-TrustBroker core workflow protocol and one of the key deliverables of the open call project: <https://datatracker.ietf.org/doc/draft-poehn-dame/>. In general, GÉANT-TrustBroker implements a superset of the functionality described in the DAME Internet-Draft. This document uses the acronym DAME whenever the related functionality is covered by what will one day hopefully be a standard (IETF RFC), and the term GNTB whenever GÉANT-TrustBroker-specific functionality is involved.

GNTB has been designed and implemented with scalability in mind; as SAML metadata is only exchanged on-demand, it scales up to large numbers of SPs and IDPs easily compared to huge SAML metadata files in inter-federations such as eduGAIN. However, the simplest scenario that still can demonstrate all key features of GNTB consists of the GNTB core service, one SP, and two IDPs. The project's demonstrator implements such a simple scenario and gives a tangible impression of the benefits of using GNTB.

The demonstrator scenario is about a fictitious international research project, named COLORado. There are three project partners: Blue University, Yellow University, and Grey Services. The two universities are assumed to be from different countries, i.e., they are not members of the same national/NREN federation, and Grey Services is assumed to be an industry partner; they do not share membership in inter-federations, such as eduGAIN, and have taken no a-priori measures to exchange SAML metadata. While both universities are IDPs, Grey Services is an SP hosting a web-based collaboration platform for the project. In this context, the IDPs use the SCHAC user attribute schema, while the SP uses a proprietary user attribute data format. The goal of the demonstrator is to use GNTB to exchange SAML metadata between each IDP and the SP, and to implement and share a user attribute conversion rule at one IDP, which the other IDP then re-uses without manual implementation efforts. Users from both IDPs get access to the SP immediately, which would not be possible without the use of GNTB.

Figure 2-1 on the next page shows the setup:

- The TTP GÉANT-TrustBroker (GNTB core service) implements the DAME-based SAML metadata exchange and facilitates the sharing and re-use of user attribute conversion rules. It is assumed to be operated by GÉANT, i.e., research projects are not supposed to operate their own GNTB core service.
- Two IDPs: Blue University and Yellow University have never cooperated before but now are partners in the COLORado project. They are not members of a common federation, as mentioned before. However, to show the re-use of user attribute conversion rules, we assume that they both can provide the user attributes defined in the SCHAC schema.
- One SP: Grey Services provides a collaboration platform that both universities would like to use to manage their project; it has never cooperated with either IDP before and does not share any federation membership with the IDPs.

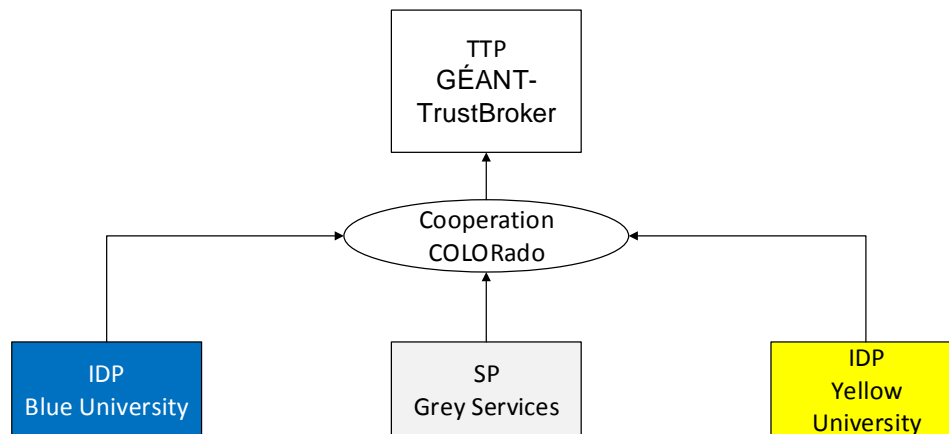


Figure 2-1: Demonstrator setup

The SPs and IDPs use different versions of the original Shibboleth software and, for IDPs, of the web server Tomcat. As Shibboleth 3 is not officially released for SPs yet and the release date for IDPs was 22nd December 2014, only Shibboleth 2 is used.

2.1 GNTB management workflow

Before an arbitrary organization, e.g., the IDP of Yellow University or the SP of Grey Services, can make use of GNTB, they need to register their metadata. As it is possible that metadata may change, e.g., when PKI certificates are renewed or DNS names change, each provider needs to be able to update its metadata and remove old versions.

This management workflow is shown using the IDP of Yellow University as an example. In order for its employees and students to be able to use their IDP within the collaboration COLORado, the IDP of Yellow

University has to be registered at the GNTB core service. To add the IDP, one of its administrators must register at the GNTB core service web-frontend. After the registration, he needs to be validated using one of the options discussed in milestone document M 1.1.1, e.g., based on creating a DNS entry or responding to a confirmation email. After this validation, the administrator is able to start the workflow.

The first step in order to add the IDP of Yellow University to GNTB is to get a copy of the IDP's metadata. As the provider is not and has never been registered at GNTB before, the administrator has to create a new provider of type IDP within the GNTB's web-frontend. While creating the provider, the administrator has to enter the provider's entity id, its provider type, an optional organisation that the provider belongs to, and a description as shown in Figure 2-2 on the next page.

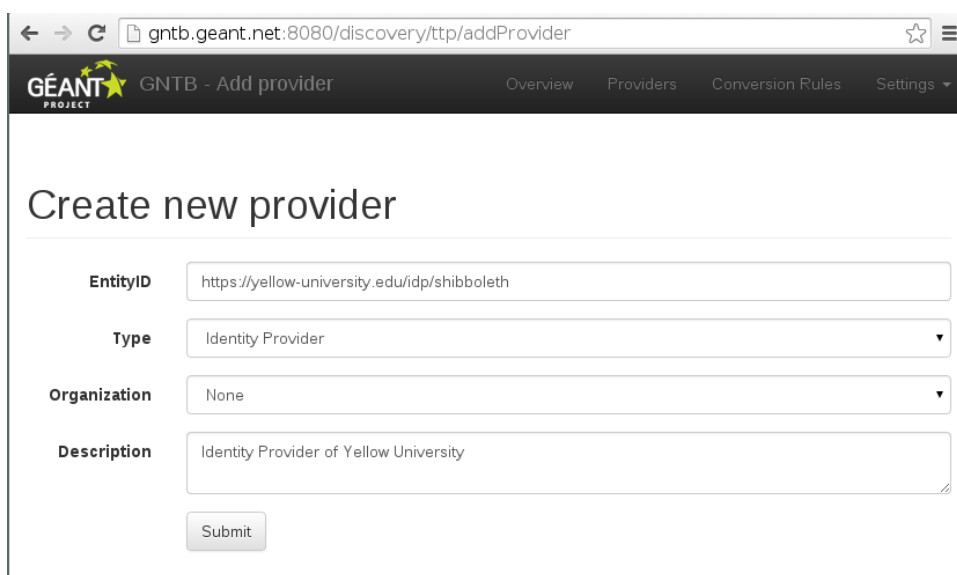


Figure 2-2: Screenshot of the GNTB web-interface for adding a new provider

To use IDP or SP metadata with GNTB, an extension needs to be added to the SAML metadata itself, before uploading it to TrustBroker's central repository. This extension specifies the location of the metadata exchange mechanism as defined in the provider's configuration (servlet `ttp_metasync` described in section 4.2.2.1); this SAML metadata extension is part of the DAME Internet-Draft and therefore supposed to be automatically added by DAME-compliant SAML implementations in the future. However, when using the prototype implementation of the extension for Shibboleth, it needs to be added manually using a text or XML editor.

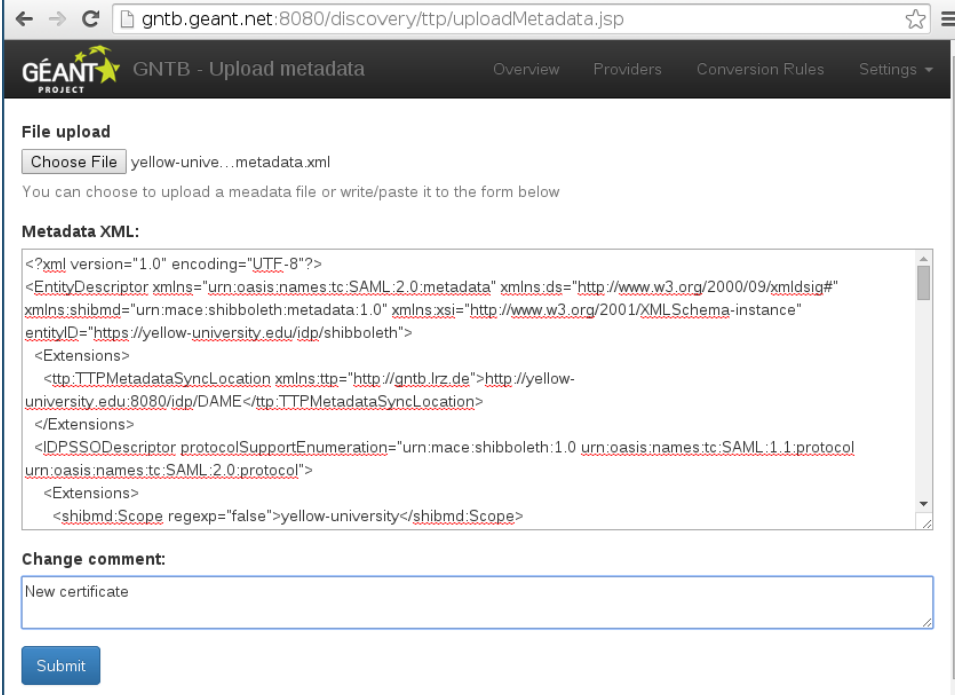
```
<EntityDescriptor xmlns="urn:oasis:names:tc:SAML:2.0:metadata"
  xmlns:ds=http://www.w3.org/2000/09/xmldsig#
  xmlns:shibmd="urn:mace:shibboleth:metadata:1.0"
  xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
  entityID="https://yellow-university.edu/idp/shibboleth">
  <Extensions>
    <ttp:TTPMetadataSyncLocation xmlns:ttp="http://gntb.lrz.de">
      http://yellow-university.edu:8080/idp/DAME
```

```
</http:TTPMetadataSyncLocation>
</Extensions>
<IDPSSODescriptor protocolSupportEnumeration="urn:mace:shibboleth:1.0
urn:oasis:names:tc:SAML:1.1:protocol urn:oasis:names:tc:SAML:2.0:protocol">
```

If the SAML metadata upload is complete, the administrator of Yellow University has to activate the provider's metadata. GNTB stores old versions of metadata to allow providers to roll-back unsuccessful changes quickly and does not activate new metadata automatically in order to allow administrators to prepare and upload metadata independent from activating it. The administrators of the IDP of Blue University and the SP of Grey Services have to add their SAML metadata in the same way. Afterwards GNTB is successfully set up to establish the connections between the three providers according to the DAME core workflow.

Continuing with the actions in the management workflow, after a few months the administrator of Yellow Universities IDP notices that the certificate used in its metadata is about to expire. He then generates new metadata for the IDP using a new certificate. To distribute the new metadata to SPs that want to connect with the IDP, the administrator then logs on to the GNTB web-interface and uploads and activates the new set of metadata as shown in Figure 2-3.

After a while the project COLORado may be finished and the IDP of Yellow University no longer wants to make use of GNTB as there are no further projects planned. The administrator logs onto the TrustBroker's web-interface and removes the IDP's metadata. At this point no new SPs can initiate a connection with the IDP via GNTB, but SPs connected previously can still be used unless their SAML metadata is manually removed from the IDP's local configuration.



gntb.geant.net:8080/discovery/http/uploadMetadata.jsp

GÉANT PROJECT GNTB - Upload metadata Overview Providers Conversion Rules Settings

File upload

Choose File yellow-unive...metadata.xml

You can choose to upload a meadata file or write/paste it to the form below

Metadata XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<EntityDescriptor xmlns="urn:oasis:names:tc:SAML:2.0:metadata" xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
xmlns:shibmd="urn:mace:shibboleth:metadata:1.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
entityID="https://yellow-university.edu/idp/shibboleth">
  <Extensions>
    <http.TTPMetadataSyncLocation xmlns:ttp="http://gntb.lrz.de">http://yellow-
    university.edu:8080/idp/DAME</http.TTPMetadataSyncLocation>
  </Extensions>
  <IDPSSODescriptor protocolSupportEnumeration="urn:mace:shibboleth:1.0 urn:oasis:names:tc:SAML:1.1:protocol
  urn:oasis:names:tc:SAML:2.0:protocol">
    <Extensions>
      <shibmd:Scope regexp="false">yellow-university</shibmd:Scope>
    </Extensions>
  </IDPSSODescriptor>
</EntityDescriptor>
```

Change comment:

New certificate

Submit

Figure 2-3: Screenshot of the web-interface for uploading metadata

2.2 GNTB core workflow (DAME)

The SAML metadata of a specific SP and IDP is exchanged in the GNTB core workflow, shown in Figure 2-4. The user triggers the SAML metadata exchange as specified in the DAME protocol specification. To ensure that only valid users of IDPs can trigger the SAML metadata exchange, the GNTB core service has the user authenticated by her IDP before actually starting the metadata exchange. When the metadata has been exchanged successfully and integrated locally on both the IDP and SP side, the user gets access to the service.

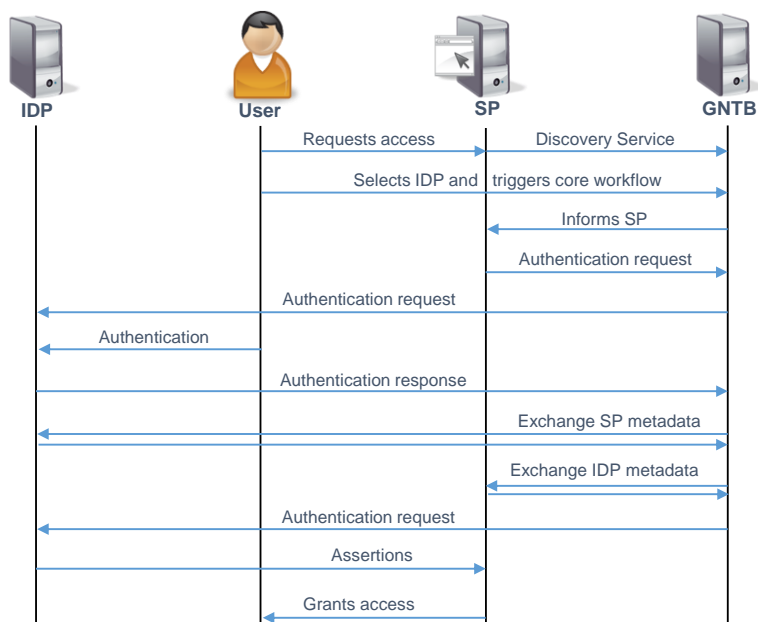


Figure 2-4: DAME core workflow

In our example, the COLORado project, a user Marina from Blue University wants to use the collaboration service provided by Grey Services. She navigates to the SP's login page and because SP Grey Services and IDP Blue University have not yet exchanged their SAML metadata, she cannot select her IDP from the embedded discovery service directly. But as the SP and her IDP are set up for using GNTB, the embedded discovery service of the SP can forward the discovery request to the GNTB discovery service. Figure 2-5 shows the embedded discovery service of Grey Services:

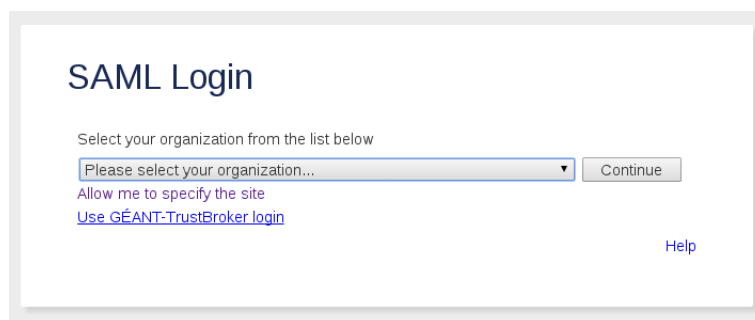


Figure 2-5: Screenshot of the SP's Embedded Discovery Service showing the option of using GNTB

Because both IDPs and the SP need to be registered at GNTB, GNTB knows about all registered entities and user Marina can select her IDP Blue University from GNTB's CDS. An example of how the discovery service at the GNTB core service looks like with a few example IDPs is shown in Figure 2-6.

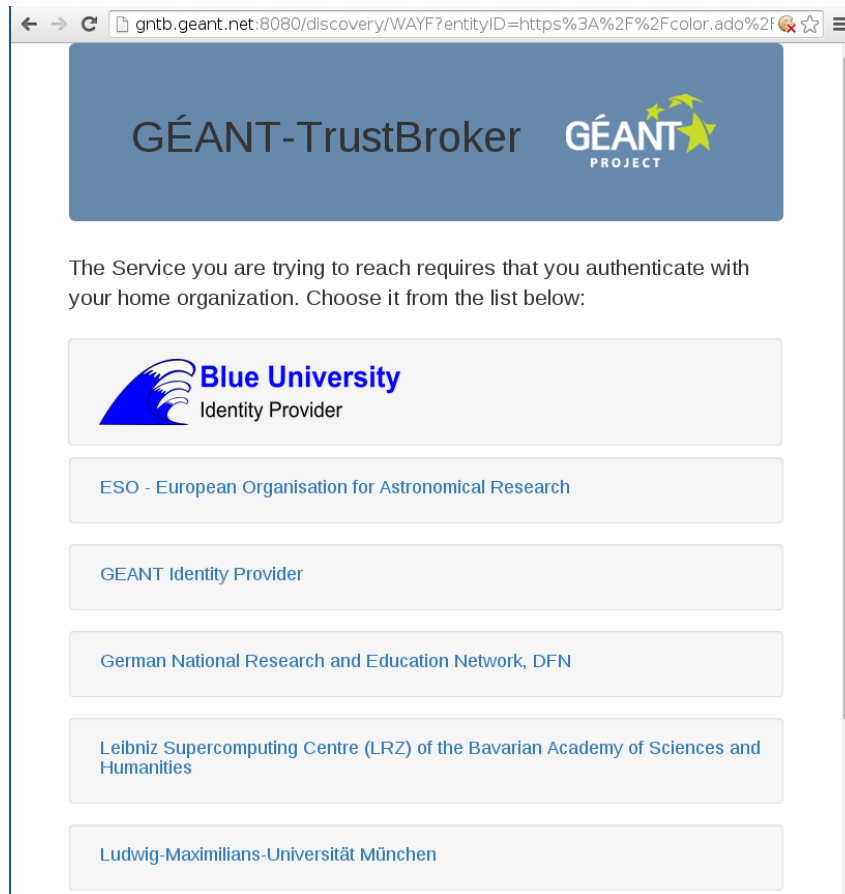


Figure 2-6: Screenshot of the IDP selection at the GNTB core service

The information about the selected IDP is then sent back to the SP of Grey Services, which now can generate a SAML authentication request for Marina's IDP. At this point the SP does not have any metadata information about the IDP and can only infer the location of its SAML endpoints using the entity id it knows from the GNTB discovery service. The newly generated authentication request is sent to the GNTB core service, which uses the signature of the authentication request to validate that the SP of Grey Services in fact created the request. This ensures the intent of the SP to exchange its SAML metadata with the user-selected IDP and prevents flooding attacks that try to get GNTB to exchange metadata between unsuspecting providers.

After receiving the authentication request from the SP, the GNTB core service checks the centrally managed whitelists and blacklists, which can be used by provider administrators to explicitly allow or disallow specific metadata exchanges. We assume in this example that the metadata exchange between SP of Grey Services and IDP of Blue University is allowed and therefore the authentication request of the SP is stored temporarily at GNTB. Afterwards GNTB generates a new SAML authentication request and sends it to the IDP of Blue University. Unless user Marina already has a valid session with her IDP, she needs to be authenticated. In the case of successful authentication, a response message containing a SAML authentication assertion is sent back to the GNTB core service. After that GNTB initiates the SAML metadata exchange.

The first step of the metadata exchange is to send the metadata of the SP of Grey Services to the IDP of Blue University. The metadata of the SP should contain the list of attributes it would like to acquire or requires for its operation using standard mechanisms that are already in use in many of today's federations. The IDP compares this list to the attributes it can provide. If there are attributes missing, the IDP will ask the GNTB core service whether it has attribute conversion rules that would allow the IDP to infer the missing attributes from the ones it has available. At this point the GNTB core service does not have any conversion rules that could be used yet, which is not a show stopper because the requested attribute is *optional* in the demonstrator scenario; however, missing *mandatory* attributes would result in the service not being usable immediately by the user, providing only the small benefit that IDP administrators can be notified by the GNTB extension automatically that mandatory attributes are missing. The creation and management of conversion rules as well as the automatic exchange of attribute conversion rules will be shown in dedicated sections below.

After the IDP of Blue University has integrated the metadata of SP Grey Services, the metadata of the IDP is sent to the SP. This completes the metadata exchange according to DAME and the GNTB core service can relay the previously stored SAML authentication request from the SP to the IDP. Because the SAML metadata has been exchanged, Blue University's IDP knows how to respond to the request directly to the SP of Grey Services, and the SP can confirm the authentication response's validity and grant Marina access to the service as shown in Figure 2-7. COLORado's collaboration service expects a custom user attribute called skypeID for a plugin that displays the Skype online status of all project participants. As the IDP of Blue University only has access to the attributes from the SCHAC schema, it cannot find and transmit the skypeID attribute out-of-the-box and Marina cannot use the plugin yet.

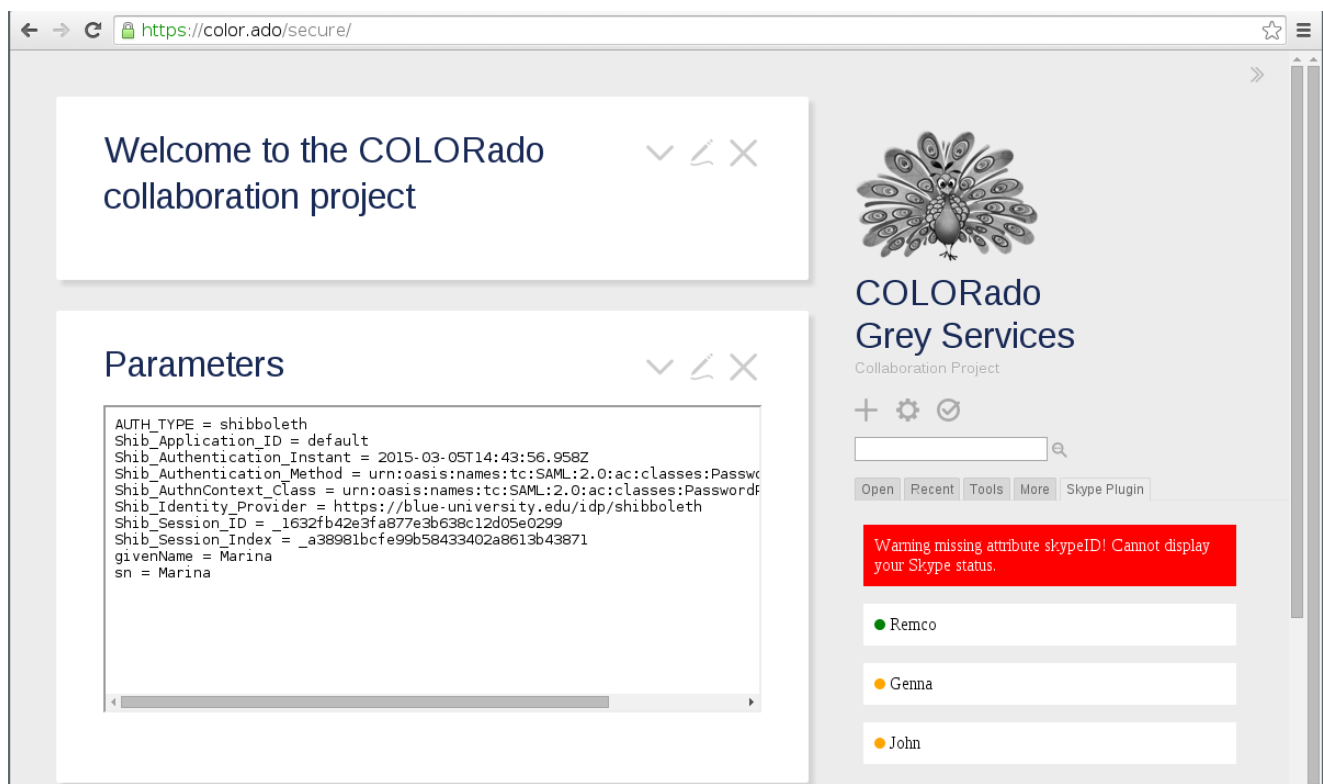


Figure 2-7: Screenshot of the collaboration service without the skypeID

2.3 GNTB conversion rule workflow

The conversion rule repository implemented by the GNTB core service allows the sharing and re-use of conversion rules by IDPs. If an IDP needs a conversion rule, it can search in the repository, download, and integrate it into its local configuration.

Conversion rules may be necessary when establishing links between previously unknown SPs and IDPs as they might use different attribute conventions. They might be as simple as converting date formats, combining “given name” and “surname” to form a new attribute “common name”, or be used in an application-specific way. The filtering of values through these conversion rules helps the SP to minimize the number of requested attributes and protects the users’ personal data because the filtering is happening on the IDP side.

For example, if the collaboration service provided by the SP of Grey Services requests an attribute `skypeID`, which is unknown to the IDPs, it could be tried to derive it from other available attributes. As Marina previously gained access to the SP of Grey Services using GNTB, she informs her IDP administrator, Azuro, that the skype plugin of the collaboration project’s website does not work properly because the optional `skypeID` attribute is missing.

Azuro then implements an attribute conversion rule that uses an attribute known to the IDP to form the `skypeID`. Luckily, the IDP stores the `schacUserPresenceID` attribute, which usually contains various messaging application identities, e.g., for XMPP, SIP, and also Skype. Azuro therefore can create a conversion rule that extracts the user’s `skypeID` from the `schacUserPresenceID` and releases it as a new attribute called `skypeID` to the SP Grey Services. The attribute conversion rule can be directly integrated in the local configuration or written as an XSLT file that can be applied to the IDP’s `attribute-resolver.xml` to insert the rule while automatically resolving local names of attributes. An example for such an attribute conversion rule is shown in appendix A.3. To locally test the application of the rule, the commandline utility `xsltproc` can be used as shown below. The output of the applied XSLT to the XML document is piped to the program `xmllint`, which will nicely format the xml, which is then displayed using `less`. If the result is satisfying, it can be copied to the `attribute-resolver.xml` file.

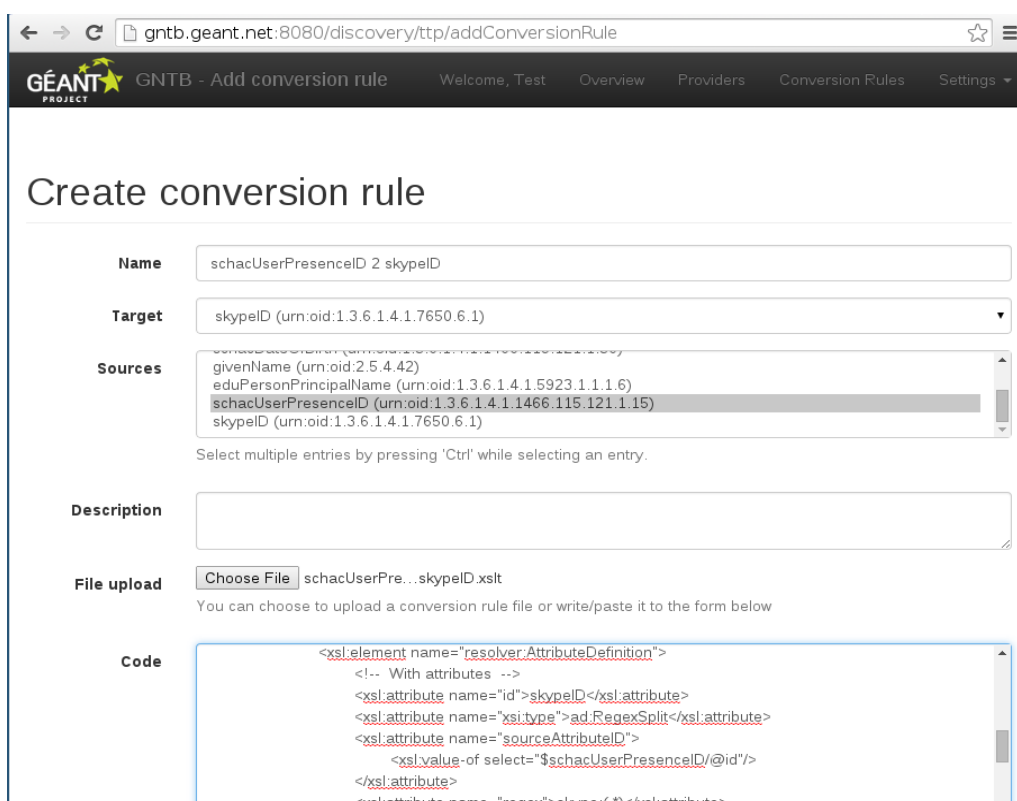
```
# xsltproc skypeID.xslt /opt/shibboleth-idp/conf/attribute-resolver.xml \
  | xmllint --format - | less
```

Besides the `attribute-resolver.xml` also the `attribute-filter.xml` needs to be modified in order to release the attribute to the SP. This can be tested using a similar method using the provided XSLT file `extendAttributeFilters.xslt`. This XSLT file needs two parameters, the attribute id of the new attribute and the entity id of the SP the attribute should be released to:

```
# xsltproc --stringparam attributeID skypeID \
  --stringparam spEntityID https://color.ado/shibboleth \
  /root/ttp-tools/extendAttributeFilters.xslt \
  /opt/shibboleth-idp/conf/attribute-filter.xml \
  | xmllint --format - | less
```

After testing the rule using his IDP, Azuro chooses to publish the rule to the central attribute conversion rule repository at the GNTB. To publish a conversion rule, the administrator has to log into the GNTB's web-interface and add a new conversion rule. This is displayed in Figure 2-8. He needs to specify the required source attributes' OIDs and the destination attribute's OID that will be created by the rule. This allows other IDPs to automatically re-use the rule if necessary. As Azuro tested and implemented the rule locally, Marina is directly able to access the collaboration service after a new login and the Skype plugin works as expected.

Conversion rules might sometimes contain errors; instead of removing them completely from the repository, another provider's administrator might want to update and correct them. In this example the regex used by Azuro "skype: (.*)" would generate an invalid skype id if the substring "skype:" is part of another attribute. Additionally, the regex allows an empty skype id, which should not be transmitted. The administrator of the SP of Grey Services corrects the regex to "^skype: (.+) \$" and uploads his improved version of the conversion rule. To prevent both rules from being used simultaneously, a GNTB administrator can remove the old version of the rule after reviewing the change (alternative methods, such as having IDP administrators give a rating of re-used conversion rules, are part of the GNTB design but have not yet been implemented).



The screenshot shows the 'GÉANT PROJECT GNTB - Add conversion rule' web interface. The page title is 'Create conversion rule'. The interface includes the following fields and content:

- Name:** schacUserPresenceID 2 skypeID
- Target:** skypeID (urn:oid:1.3.6.1.4.1.7650.6.1)
- Sources:** A list of source attributes with their OIDs:
 - givenName (urn:oid:2.5.4.42)
 - eduPersonPrincipalName (urn:oid:1.3.6.1.4.1.5923.1.1.1.6)
 - schacUserPresenceID (urn:oid:1.3.6.1.4.1.1466.115.121.1.15)** (highlighted)
 - skypeID (urn:oid:1.3.6.1.4.1.7650.6.1)
 Below the list is the instruction: 'Select multiple entries by pressing 'Ctrl' while selecting an entry.'
- Description:** An empty text area.
- File upload:** A 'Choose File' button and the filename 'schacUserPre...skypeID.xslt'. Below it is the text: 'You can choose to upload a conversion rule file or write/paste it to the form below'.
- Code:** A text area containing XSLT code:


```
<xsl:element name="resolver:AttributeDefinition">
  <!-- With attributes -->
  <xsl:attribute name="id">skypeID</xsl:attribute>
  <xsl:attribute name="xsi:type">ad:RegexSplit</xsl:attribute>
  <xsl:attribute name="sourceAttributeID">
    <xsl:value-of select="$schacUserPresenceID/@id"/>
  </xsl:attribute>
  <xsl:attribute name="regex">skype: / *</xsl:attribute>
</xsl:element>
```

Figure 2-8: Screenshot of the web-interface for conversion rules

2.4 GNTB complete core workflow

At this point the connection between the IDP of Blue University and the SP of Grey Services is set up. Another user, Sunny from Yellow University, now also requests access to the COLORado's collaboration tool at Grey Services. Her IDP is also registered at GNTB. As Marina did before, Sunny now navigates to the project's website of Grey Services and tries to login. Yellow University is also not directly available at the Embedded Discovery Service and Sunny uses the redirect to GNTB option. There she selects her IDP Yellow University.

This starts the DAME core workflow as described previously. The GNTB core service initiates the metadata download of Grey Service's SP to the IDP of Yellow University. As the Yellow University IDP uses the same SCHAC attribute schema as the Blue University IDP, it also cannot directly supply the requested attribute skypeID. It asks the GNTB core service whether there are any attribute conversion rules available that use the SCHAC attributes to generate the required skypeID attribute. Because Azuro from Blue University uploaded the rule he wrote, the GNTB core service offers this rule to the Yellow University IDP. The IDP then downloads the conversion rule and applies it to its configuration automatically. Afterwards the metadata of the Yellow University IDP is sent to the SP of Grey Services. Sunny does not notice the metadata or conversion rule exchange visibly, as the login process only takes slightly longer than normal and she is able to use the collaboration service with the Skype plugin without any manual configuration.

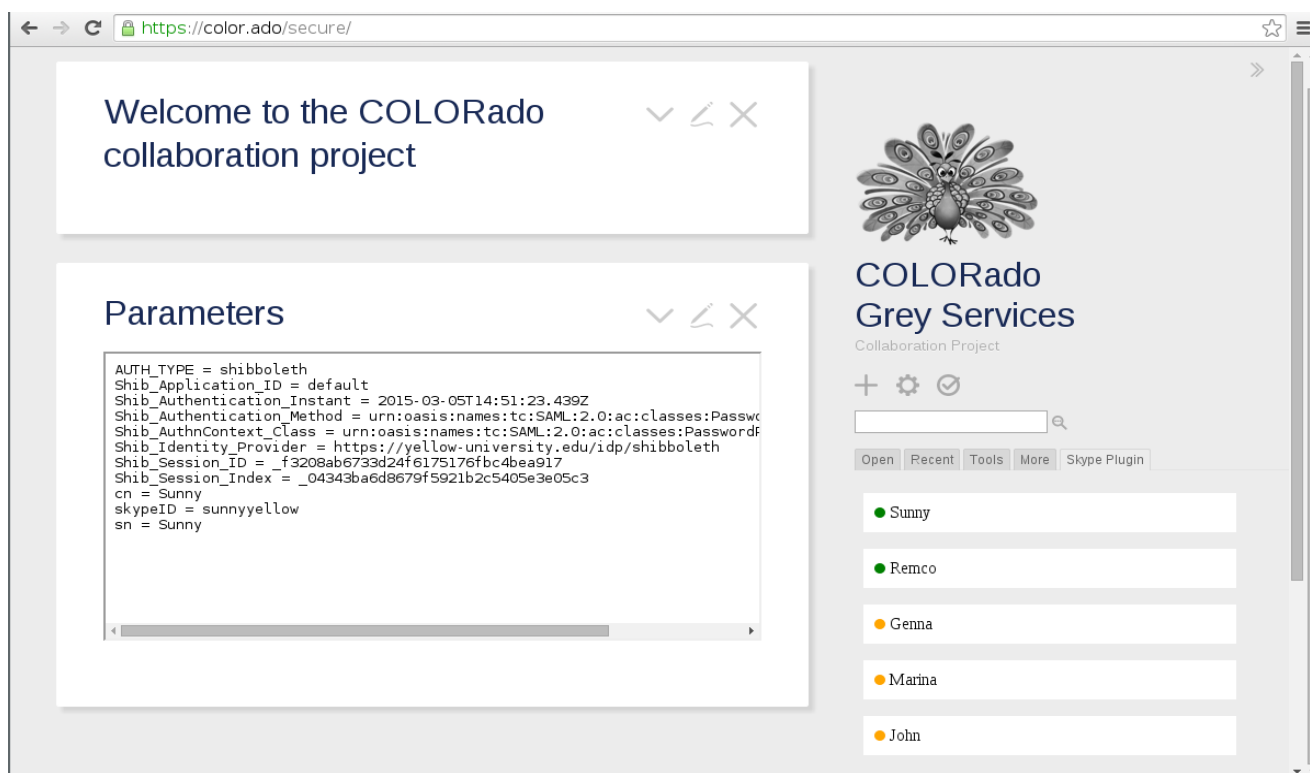


Figure 2-9: Screenshot of the collaboration service after automatic conversion of the skypeID

3 **GÉANT-TrustBroker implementation overview**

Before going into details about the installation and configuration of the implemented components, this section gives a short overview. The GNTB implementation consists of three parts:

1. GNTB core service, including a DAME trusted third party (TTP), implemented as an extension to the Shibboleth Centralized Discovery Service
2. GNTB extension to the Shibboleth IDP software, including DAME IDP functionality
3. DAME extension to the Shibboleth SP software

The GNTB core service is the central technical component. It is used to automatically exchange technical information, i.e., SAML metadata, between IDPs and SPs on-demand, and to re-use previously shared user attribute conversion rules. The GNTB core service proof-of-concept implementation is based on the Shibboleth Centralized Discovery Service. It is already explained in chapter 2 of milestone document M 4.1.1 and therefore omitted in this overview, but its installation and configuration are detailed in Section 4.

DAME extensions of the Shibboleth IDP and SP software are needed to communicate with the GNTB core service. The extensions implemented for GNTB furthermore automate previously manual steps, e.g., the integration of the exchanged SAML metadata as well as downloaded attribute conversion rules into the running Shibboleth instances. The extensions fit seamlessly with the Shibboleth software for IDPs respectively SPs. Details about the implementation of these extensions are explained in the next sections.

3.1 GNTB extension for Shibboleth IDP

The extension for the Shibboleth IDP basically adds two new features, which have, like the IDP itself, been implemented in Java. These features are 1) a metadata provider that is able to provision the other components of the IDP with the metadata that was transmitted by the DAME TTP, and 2) two handlers that can be used by the GNTB core service to synchronize metadata and conversion rules. Both components are implemented in their respective Java classes `TTPMetadataProvider` and `TTPMetadataSyncServlet`. The integration with some important existing classes is shown in Figure 3-1.

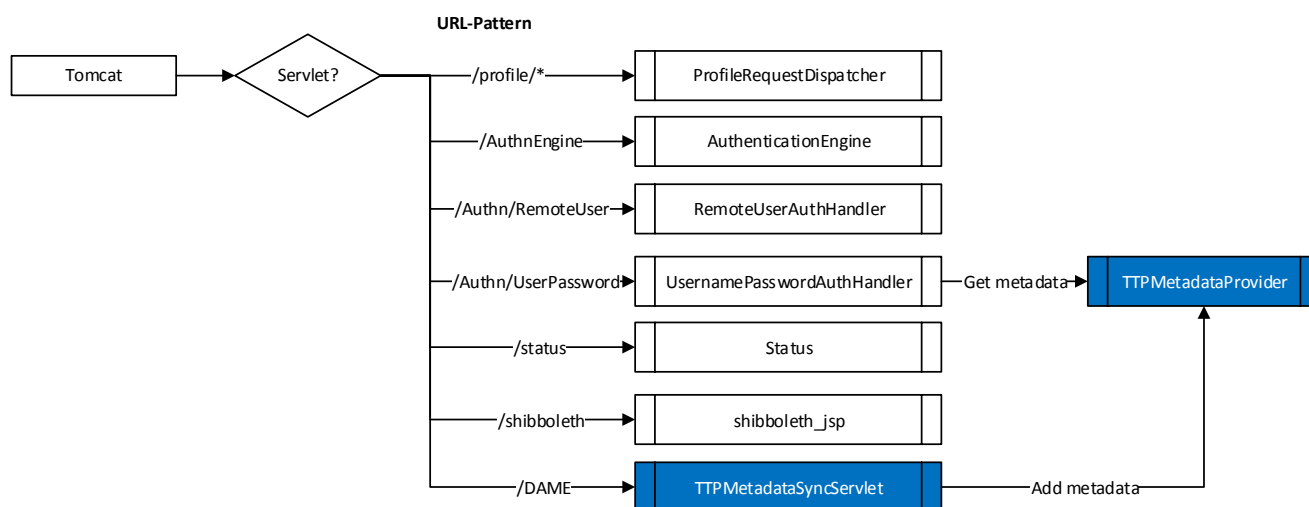


Figure 3-1: Overview of the implementation of the IDP extension

In order for the IDP to use the new metadata provider, it has to be defined in a schema definition. This file describes that this element extends the generic `MetadataProviderType` and defines that the only attribute is named `metadataStorageDirectory`, which is of type *string* and specifies where the metadata files should be saved.

```

<?xml version="1.0" encoding="UTF-8"?>
<schema
  targetNamespace="urn:mace:gntb.lrz.de:ttpeextension:ttpmetadataprovider"
  xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:md="urn:mace:shibboleth:2.0:metadata"
  elementFormDefault="qualified">

  <import namespace="urn:mace:shibboleth:2.0:metadata"
    schemaLocation="classpath:/schema/shibboleth-2.0-metadata.xsd" />

  <complexType name="TTPMetadataProvider">

```

```
<complexContent>
  <extension base="md:MetadataProviderType">
    <attribute name="metadataStorageDirectory" type="string" />
  </extension>
</complexContent>
</complexType>
</schema>
```

The metadata synchronization servlet saves exchanged metadata inside the metadata storage directory. To prevent conflicting and invalid filenames, the metadata files are saved using the SHA-1 hash of the respective entity id as file names, which complies with the SAML specification. When searching the metadata of a specific provider, the `TTPMetadataProvider` hashes the entity id of the requested provider and tries to locate the file. If it exists, it is loaded and passed to the requesting process. If it does not exist, the metadata provider reports that it cannot find any metadata for the searched provider.

The conversion rule synchronization part is slightly more complex as it first has to determine which attributes already exist at the local IDP. Those are then compared to the attributes requested by the SP's metadata. If attributes are missing, a request is sent to the GNTB core service, asking for conversion rules that use available attributes to generate the missing ones.

If one rule or multiple rules are found, the original versions of `attribute-resolver.xml` and `attribute-filter.xml`, which specify which attributes the IDP can generate and releases to which SP, are backed up. Then the Extensible Stylesheet Language Transformation (XSLT) of the conversion rule is applied to the `attribute-resolver.xml` and a second XSLT for adding the release of the attribute to the requesting SP is applied to the `attribute-filter.xml`. After applying the changes, the attribute resolver and attribute filtering engine need to be reloaded to activate the changes.

3.2 DAME extension for Shibboleth SP

The Shibboleth SP and the extension are implemented in C/C++. The extension basically has the same functionality as the IDP extension with the exception that the SP does not need conversion rule synchronization and application, as this is only done on the IDP side. But as SAML sessions, according to the SAML 2.0 Web Browser SSO Profile, are initiated by the SP, a special `SessionInitiator` element is needed.

The new session initiator is used to create an authentication request without having the metadata of the target IDP. The request is then sent to the DAME TTP, which verifies it and starts the metadata exchange as described in the core workflow. The authentication response is sent to the regular SP handler for authentication responses, e.g., the `AssertionConsumerService`.

In contrast to the IDP, the Shibboleth SP implementation is split into two parts, one that runs within the Apache webserver's process and a separate Shibboleth daemon process, `shibd`. Those two processes communicate with each other using different types of inter-process communication. This is shown in Figure 3-2.

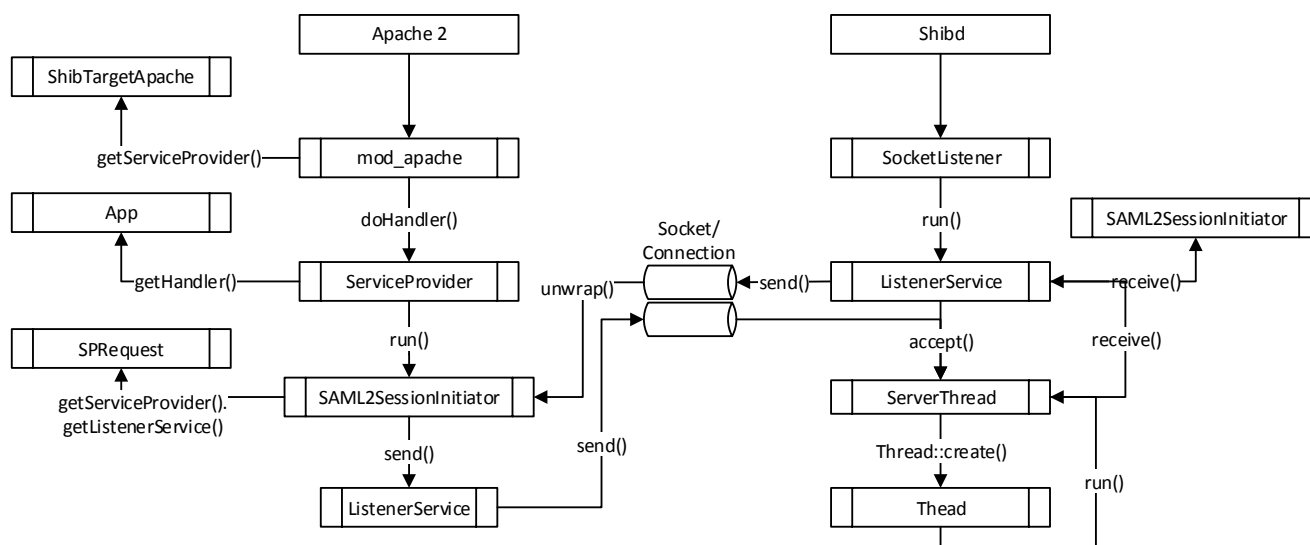


Figure 3-2: Overview of the inter-process communication used in the Shibboleth SP implementation

3.3 Getting the source code

All software components of GÉANT-TrustBroker can be downloaded from the subversion (SVN) repository <http://svn.geant.net/GEANT/TrustBroker>. It contains the commented source code, archived releases, and detailed installation instructions. The repository is structured as follows (all folders with the exception of the releases folder contain their own SVN directories branches, tags, and trunk):

- releases: contains packaged release versions of the extensions that should be used when installing an extension
- shib-eds-modification: contains the source code of the modified Shibboleth Embedded Discovery Service
- shib-idp-dame-extension: contains the source code of the IDP extension
- shib-sp-dame-extension: contains the source code of the SP extension
- shib-ttp-dame-cds-extension: contains the source code of the Shibboleth Centralized Discovery Service extension
- shib-ttp-db-dame-module: contains the source code for a database module that is used by the IDP and the GNTB extension

4 **GÉANT-TrustBroker installation and configuration**

The extensions to Shibboleth were designed according to the Shibboleth guidelines for plugins and extensions as shown in [2] and [3]. This allows for independent development of Shibboleth and our extensions, ensuring that administrators familiar with the plugin installation process can install the extension in their existing setup easily. In the following sections, the installation and configuration of the three parts of the GNTB implementation are explained.

4.1 **GNTB core service**

The GNTB extension of the Shibboleth Centralized Discovery Service (CDS) is implemented in Java and it includes the functions required for the metadata exchange in the role of a TTP in the DAME protocol specification, the GNTB-specific conversion rule repository, and the GNTB-specific management web frontend, as described in milestone document M 4.1.1.

The extension was developed based on the current Shibboleth CDS version 1.2.1 on a Debian Linux host running an Apache Tomcat 7 web application server. Because this component requires a database, it adds a MySQL database connector as additional dependency to the CDS. The required .jar file is distributed with the release version.

4.1.1 **Installation**

Before installing the TTP extension, a regular CDS needs to be installed. The necessary steps for installing and configuring the CDS are described in the official Shibboleth Wiki [4].

Additionally, a MySQL database has to be installed to store information about providers, attribute conversion rules, users, and the relations. The code to create the database and the corresponding user can be found in appendix A.1.

With the database in place, the next step is to extend the installed CDS. The extension is provided as a .zip file, which needs to be extracted. To have a shorter and version independent path, the extracted directory is renamed to `ttp-extension`.

```
~# unzip shib-ttp-dame-cds-extension-*-src.zip
~# mv shib-ttp*/ ttp-extension
```

To add the extension to the CDS, some files need to be copied to the CDS source directory, which contains the files used to initially install the Shibboleth CDS, e.g., `shibboleth-ds-1.2.1`.

The files needed for installing the TTP extension are located in the `lib` and `src/main/webapp/ttp` directories.

```
~# cd shibboleth-ds-1.2.1
~/shibboleth-ds-1.2.1# cp ~/ttp-extension/lib/* lib/
~/shibboleth-ds-1.2.1# mkdir src/main/webapp/ttp
~/shibboleth-ds-1.2.1# cp -r ~/ttp-extension/src/main/webapp/ttp/* \
    src/main/webapp/ttp
~/shibboleth-ds-1.2.1# cp -r ~/ttp-extension/src/main/webapp/WEB-INF/* \
    src/main/webapp/WEB-INF
```

4.1.2 Configuration

The configuration of the GNTB specific CDS extension consists of two parts:

- `web.xml`
- `wayfconfig.xml`

The configuration file `web.xml` is the deployment descriptor as typically used in Java web environments. It must be stored within the `/WEB-INF` directory in the web application's root, whereas `wayfconfig.xml` provides the metadata provider, i.e., IDPs and SPs that are known to the CDS.

4.1.2.1 Configuration of `web.xml`

The already existing `web.xml` configuration file of the CDS has to be extended by the TTP-specific configuration and servlets. Most of the configuration, specified in appendix A.2, can be copied without changes, but at least the `entityID`, database user (`dbUser`), and especially the password (`dbPassword`) have to be set up accordingly to one's own environment.

The context parameter `metadata` specifies the location to be used to store GNTB's SAML metadata, which will be automatically generated on the first start of the server. The location of the certificate to be integrated into the

metadata can be specified by the `cert` parameter. A certificate can be generated using `openssl` or `shib-keygen`, a utility provided by Shibboleth SP software:

```
~# shib-keygen -o /opt/shibboleth-ds/cert -h ttp.example.com
```

The values for `ttpMetadataStorageDirectory` and `ttpConversionRuleStorageDirectory` specify the directories that are used to store the SAML metadata of the IDPs and SP registered at GNTB as well as the shared attribute conversion rules. These directories must exist and be readable and writable by the Tomcat user.

After setting up these options, the CDS installation script needs to be run again without regenerating and overwriting the existing configuration.

```
~/shibboleth-ds-1.2.1# env JAVA_HOME=/usr/lib/jvm/default-java\  
./install.sh
```

4.1.2.2 Configuration of `wayfconfig.xml`

To finalize the initial configuration, two handlers `TTPServerMetadataSyncHandler` and `TTPServerConversionRuleSyncHandler` have to be set up by adding the following two lines after the `DiscoveryServiceHandler` element in the `/opt/shibboleth-ds/conf/wayfconfig.xml` file.

```
<TTPServerMetadataSyncHandler location=".+/TTP" default="true"/>  
<TTPServerConversionRuleSyncHandler location=".+/TTPCRS" default="true" />
```

Because the GNTB core service uses the CDS-provided functionality for IDP discovery, also the metadata of the connected SPs and IDPs has to be added to this configuration file.

```
<MetadataProvider displayName="Example IDP" identifier="eIDP"  
  url="file:///opt/shibboleth-ds/metadata/idp-metadata.xml"/>  
<MetadataProvider displayName="Example SP" identifier="eSP"  
  url="file:///opt/shibboleth-ds/metadata/sp-metadata.xml"/>
```

This will be removed in future versions of the GNTB core service implementation as it limits the dynamic metadata exchange. There are three possible ways how to solve this:

- The CDS could be extended to allow for generic `MetadataProviders`, but this could be rather difficult to implement.
- The TTP extension generates a metadata file whenever a provider changes its metadata. The CDS could then use a single static file based metadata provider, though this might limit scalability.
- Implementation of the GNTB core service as a standalone discovery service and integration of the discovery part closer to the metadata exchange. So far, this seems to be the best choice.

After finishing the configuration, the Tomcat server must be restarted.

```
~# service tomcat7 restart
```

4.1.2.3 Further steps

After the first successful start of the Tomcat server including the installed TTP extension, the SAML metadata file has been generated at the location specified in the `web.xml` configuration (context parameter `metadata`). This metadata file has to be distributed to all IDPs that want to use GÉANT-TrustBroker, as they need to be able to respond to authentication requests made by the DAME TTP of GNTB.

If any errors occur during the start of the Tomcat server, a log file is usually generated at `/opt/shibboleth-ds/logs/discoveryService.log` (The Tomcat process must be able to write to this directory). To increase logging verbosity, the file `/opt/shibboleth-ds/config/logging.xml` can be edited. To increase only the TTP extension's log verbosity, add the following new element to it:

```
<logger name="de.lrz">
  <level value="DEBUG"/>
</logger>
```

4.2 GNTB extension of the Shibboleth IDP

The installation and configuration of the GNTB or DAME protocol extension of the Shibboleth IDP software is similar to the procedure described in the previous section. Currently the GNTB extension has been tested successfully with versions 2.4.0 and 2.4.2 of the Shibboleth IDP software, with other versions expected to work as well.

4.2.1 Installation

In order to install the extension, a working Shibboleth IDP installation is required. The process of installing and configuring a Shibboleth IDP is described in the Shibboleth Wiki [5].

The next step is to download and extract the GNTB extension. The resulting directory is renamed to remove the version number and shorten it for further reference.

```
~# unzip shib-idp-dame-extension-*-src.zip
~# mv shib-idp-dame-extension-* ttp-extension
```

In the next step the library files of the extension must be copied to the `lib` subdirectory of the IDP's source directory.

```
~# cd /usr/local/src/shibboleth-identityprovider-2.4.0
/usr/local/src/shibboleth-identityprovider-2.4.0# cp ~/ttp-extension \
    /lib/* lib/
```

4.2.2 Configuration

The configuration of the IDP's GNTB extension consists of two parts:

- `web.xml`
- `relying-party.xml`

Besides the deployment descriptor, the `relying-party.xml` specifies, e.g., the metadata provider of the IDP, which is normally the metadata feed of the federations the IDP is a member of.

4.2.2.1 Configuration of `web.xml`

After copying the libraries, the new GNTB extension specific servlets need to be added to the `web.xml` file, located at `src/main/webapp/WEB-INF` directory. The configuration of those servlets must be adjusted to the respective environment as described below.

The value element of the `initiatorACL` parameter specifies a list of space-separated IP addresses, which are allowed to initiate a metadata exchange. Please note that no DNS entries or hostnames can be used at the moment because no resolver module is currently available; this will be added to the code later, but also come with additional dependencies in the build process. Therefore, the parameter value has to be adjusted depending on the IP address or addresses of the GNTB core service. The second parameter `metadataStorageDirectory` specifies where the exchanged metadata should be stored. It is important that this directory exists and is readable and writable by the Tomcat process.

```
<web-app>
...
<!-- TTP Metadata sync handler -->
<servlet>
    <servlet-name>ttp_metasync</servlet-name>
    <servlet-class>de.lrz.gntb.idpextension.TTPMetadataSyncServlet
</servlet-class>
    <init-param>
        <param-name>initiatorACL</param-name>
        <param-value>10.0.0.2</param-value>
    </init-param>
    <init-param>
        <param-name>metadataStorageDirectory</param-name>
        <param-value>/opt/shibboleth-idp/metadata/ttp</param-value>
```

```

        </init-param>
    </servlet>
    <servlet-mapping>
        <servlet-name>ttp_metasync</servlet-name>
        <url-pattern>/DAME</url-pattern>
    </servlet-mapping>
    ...
</web-app>

```

Now the installation script can be run again in order to activate the changes. When asked, the installation script must not use a different location or overwrite existing configuration.

```

/usr/local/src/shibboleth-identityprovider-2.4.0#
env JAVA_HOME=/usr/lib/jvm/default-java ./install.sh

```

4.2.2.2 Configuration of *relying-party.xml*

After the installation is complete, some changes have to be made to the configuration of the IDP. The *relying-party.xml* configuration file needs to be extended with a new XML namespace and a new *MetadataProvider* definition. The following listing shows an abbreviated version of this file, displaying only the necessary changes to the defaults.

```

<rp:RelyingPartyGroup ...
    xmlns:ttp="urn:mace:gntb.lrz.de:ttpextension:ttpmetadataprovider"
    xsi:schemaLocation="...
        urn:mace:gntb.lrz.de:ttpextension:ttpmetadataprovider
        classpath:/schema/ttp-metadata-provider.xsd">
    ...
    <metadata:MetadataProvider id="ShibbolethMetadata"
        xsi:type="metadata:ChainingMetadataProvider">
        ...
        <metadata:MetadataProvider id="TTP" xsi:type="ttp:TTPMetadataProvider"
            location="/opt/shibboleth-idp/metadata/ttp">
        </metadata:MetadataProvider>
    </metadata:MetadataProvider>
    ...
</rp:RelyingPartyGroup>

```

After saving these changes, the Tomcat server has to be restarted to activate the extension and the new configuration.

```

~# service tomcat6 restart

```

4.3 DAME extension of the Shibboleth SP

The extension of the Shibboleth SP software was implemented in C and C++ and requires the manual compilation of the extension library. The development and compilation has been tested on a Debian Linux system using an Apache webserver. The extension has been tested and used with the Shibboleth SP software versions 2.5.3 and 2.4.3; it does not add any external dependencies to the Shibboleth SP. Also an Embedded Discovery Service has to be used with the SP to be able to first check whether the user's IDP is already available locally or whether the user should be forwarded to the DAME TTP.

4.3.1 Installation

To use the extension, it has to be compiled from source. The build process is based on `autoconf` and `g++`. Building the extension also requires some dependencies. The configure script will check for the existence of all required libraries and notify the user of those that are missing (usually none if the original Shibboleth SP successfully compiled). The following commands show which packages have to be installed on a Debian system in order to be able to build the extension. Notice that some packages require newer versions than are available from the current Debian stable repository. In those cases more recent versions exist in the Debian backports repository.

```
# apt-get install dh-autoreconf g++
# apt-get install libssl-dev libboost-dev liblog4shib-dev libxerces-c-dev
# apt-get -t wheezy-backports install libxmltooling-dev libshibsp-dev
# apt-get install xmltooling-schemas
# apt-get -t wheezy-backports install libsaml2-dev opensaml2-schemas \
    opensaml2-tools
# apt-get install apache2-threaded-dev
```

When the dependencies are installed, the build process can be started. The extension requires a Shibboleth SP to be already installed. Instructions for the installation of a Shibboleth SP can be found on the Shibboleth Wiki [6]. On some Linux distributions, e.g., Debian, there is a Shibboleth SP package available from the distribution's software repository.

The first step to install the SP extension is to extract the source files. This example extracts the files to the version independent directory name `/usr/local/src/shib-sp-dame-extension`.

```
~# mkdir /usr/local/src/shib-sp-dame-extension
~# tar -xzf shib-sp-dame-extension-*.tar.gz \
    -C /usr/local/src/shib-sp-dame-extension --strip-components=1

/usr/local/src/shib-sp-dame-extension# autoreconf -if
/usr/local/src/shib-sp-dame-extension# ./configure
```

```
/usr/local/src/shib-sp-dame-extension# make
/usr/local/src/shib-sp-dame-extension# make install
```

If the build was successful, the last command ‘make install’ will copy the generated libraries into the /usr/local/lib/shibboleth directory.

4.3.2 Configuration

The next step is to configure the Shibboleth SP to load the libraries of the DAME extension. This is done by editing the shibboleth2.xml, which on Debian will be located at /etc/shibboleth/shibboleth2.xml. At the beginning of this file, two new elements <InProcess> and <OutOfProcess> must be defined. Those elements specify which libraries the Apache and shibd processes should load.

Within the <Sessions> element an additional <SessionInitiator> of type “TTPSessionInitiator” needs to be added below the already existing regular and discovery service SessionInitiator. The new SessionInitiator will handle the requests from the TTP. Depending on the version of the Shibboleth Service Provider, the surrounding configuration might be different because new and shorter elements were added to the <Sessions> element in version 2.4. Also the order in which the elements are placed in this file is important; if it is wrong, the Shibboleth SP will not start and log an error indicating to the offending element.

In the SP’s configuration file also a new handler for initiating the metadata exchange must be added. This <Handler> element also specifies an attribute initiatorACL, which contains space-separated IP addresses of hosts that are allowed to initiate the metadata exchange.

The last configuration item is the new MetadataProvider element, which specifies a directory that should be used to store the exchanged metadata. This directory has to exist and be readable and writable by the shibd and apache2 processes.

```
<SPConfig ...>
  <OutOfProcess>
    <Extensions>
      <Library path="/usr/local/lib/shibboleth/ttp_sync.so"
        fatal="true" />
    </Extensions>
  </OutOfProcess>
  <InProcess>
    <Extensions>
      <Library path="/usr/local/lib/shibboleth/ttp_sync-lite.so"
        fatal="true" />
    </Extensions>
  </InProcess>
  ...
</Sessions ...>
```

```

...
<SSO discoveryProtocol="SAMLDS"
    discoveryURL="http://sp.example.com/shibboleth-ds/index.html">
    SAML2 SAML1
</SSO>
<!-- SAML and local-only logout. -->
<Logout>SAML2 Local</Logout>
<SessionInitiator type="TTPSessionInitiator" Location="/TTP"
    id="TTP"
ttpInitURL="http://tp.example.com:8080/discovery/TTP?action=authenticate"/>
...
<!-- TTP Metadata synchronisation. -->
<Handler type="TTPMetadataSyncHandler" Location="/MetaSync"

metadataStorageDirectory="/usr/local/etc/shibboleth/metadata/ttp"
    initiatorACL="10.0.0.2"/>
...
</Sessions>
<Errors supportContact="root@localhost"
    helpLocation="/about.html"
    styleSheet="/shibboleth-sp/main.css"/>
<MetadataProvider type="TTPMetadataProvider"
    metadataStorageDirectory="/usr/local/etc/shibboleth/metadata/ttp"/>
...
</SPConfig>

```

With the configuration done the final step is to restart both the apache2 webserver and the shibd process.

```
~# service apache2 restart && service shibd restart
```

4.4 Modification of the Shibboleth Embedded Discovery Service

A local or Embedded Discovery Service (EDS) is necessary for a SP to determine whether the metadata of a user's IDP is available or whether the user will need to choose his IDP from the CDS provided by the GNTB core service. For the demonstrator, a modification of the already existing Shibboleth EDS was implemented that allows the user to forward the Discovery Service request to GNTB's CDS if she cannot find the IDP in the list of IDPs available to the SP. The EDS is written in JavaScript and Hypertext Markup Language (HTML) and can be integrated into any login page very easily. The modification does not add any new libraries or dependencies as it only adds an HTML link and a few new configuration entries.

4.4.1 Installation

The installation of the Shibboleth EDS has not been changed. It is distributed as a compressed tarball that needs to be extracted.

```
~# tar -xzf shibboleth-embedded-ds*.tar.gz
```

The distribution contains a Makefile, which has a target “install” that will copy the files to their appropriate place. On Debian the files will be copied to `/etc/shibboleth-ds` directory.

```
~/shibboleth-embedded-ds-1.0.3+GNTB# make install
```

After the files have been copied, the Apache server has to be configured to serve the files. A default configuration file is also distributed and can be linked to the `sites-enabled` directory of apache.

```
~# ln -s /etc/shibboleth-ds/shibboleth-ds.conf /etc/apache2/sites-enabled
```

4.4.2 Configuration

The configuration of the EDS is done by editing the `idpselect_config.js` file. The modification adds three new variables that need to be adjusted for each installation. The first, “`enableTTPForward`”, needs to be set to true to enable the modification, if it is false the EDS behaves as it would without any modification. The second “`forwardTTPHost`” is the Uniform Resource Locator (URL) of the central Discovery Service run by the GNTB core service. This URL needs to be published by the GNTB operator. The third and last variable “`ttpReturnPath`” specifies where the GNTB core service should send the user’s IDP selection to. This value has to match the location of the `TTPSessionInitiator` element specified in the SP extension configuration.

```
this.enableTTPForward = true;
this.forwardTTPHost = 'http://tp.example.com:8080/discovery/WAYF';
this.ttpReturnPath = 'https://sp.example.com/Shibboleth.sso/TTP';
```

5 GÉANT-TrustBroker project assessment

In this section the Open Call project GÉANT-TrustBroker is self-assessed by first comparing it to the original Open Call text and the wider Open Call scope. Next, the concept, design, implementation, and the submitted IETF Internet-Draft are compared with the submitted project proposal. Finally, the impact of the project is discussed.

5.1 Comparison with the objectives of the Open Call text

The original requirements of the call text and the wider Open Call scope are explained. Furthermore, it is described how the project met these requirements.

The first objective of the Open Call text for project #13 was to seek a solution that puts the users in a central role for establishing the trust relationships between their IDPs and the SPs they want to use. GÉANT-TrustBroker fulfils this requirement by specifying a user-triggered workflow for SAML metadata exchange, which is the technical basis for trust setup between IDP and SP. The workflow allows for full automation, which has been implemented as part of the GÉANT-TrustBroker demonstrator and submitted to the IETF for standardization with the DAME Internet-Draft. As the metadata is exchanged between IDP and SP only on-demand, i.e., when a user wants to access a service for the first time, the resulting process has a much higher scalability than previous approaches, such as metadata-aggregating inter-federations like eduGAIN. The related real-world problem of non-interoperable user attribute schemas when IDPs and SPs are from different federations has been successfully addressed by allowing the sharing and re-use of user attribute conversion rules; this not only solves the problem of having to agree on an inter-federation-wide user attribute schema, but also reduces the IDP-side implementation workload because once a conversion rule has been implemented by one IDP, it can be used by all other IDPs automatically. Alternatively federation operators can provide conversion rules for conversions between federation specific and common international attributes.

The second objective of the Open Call text was the analysis of ways to integrate Account Chooser functionality into the workflow. The project proposal stated that “we expect technical difficulties when trying to integrate the predominant campus IDP technology with Account Chooser”. As no working account chooser service with an interface for SAML was available during the project’s run-time, integrating IDP discovery services as they are used in today’s R&E federations was favoured. Requirements for an account choosing technology were considered in the design and mock-ups have been made, but the implementation relies on the Shibboleth Discovery Service. From a user experience and usability perspective, this should be considered not a drawback but a feature, because the GÉANT-TrustBroker core workflow does not have any visible differences to the federation workflows the users are already accustomed to. Also, once a suitable SAML Account Chooser becomes available, it should be easy to integrate it into the GNTB core service web-frontend.

Efforts on GÉANT-TrustBroker were coordinated with the two other FAM/AAI open call projects, WoT4LoA and HEXAA. To a certain degree, GNTB serves as a basis for WoT4LoA as it interconnects IDPs and SPs that did not yet cooperate with each other, and the WoT4LoA mechanisms can then be applied to perform quality

management and improving accountability in the typical GNTB scenarios, such as small research projects that do not found their own community federation and therefore do not have organizational measures such as written contracts in place. Regarding HEXAA, especially the dynamic conversion of user attributes as enabled by the GNTB rule repository is relevant; attributes provided by the HEXAA attribute authority can be converted on-the-fly into the data format required by the connected SPs. Plans have been made to offer the attribute conversion functionality of GNTB as a stand-alone service for use cases like HEXAA in GN4.

5.2 Comparison with the original project proposal

The project proposal (description of work) has split the GÉANT-TrustBroker solution approach into four work packages with 17 tasks and 9 milestone/deliverable documents. It also defined three key performance indicators (KPIs) to evaluate the results.

All milestone and deliverable documents were provided in time, often a few weeks ahead of schedule, i.e., KPI 1 is clearly fulfilled. More scientific papers than originally intended were published and GÉANT-TrustBroker was presented for dissemination purposes at more conferences and community workshops than planned, i.e., WP3 was more than achieved.

All goals defined in the project proposal have been achieved; the most important results are:

- GÉANT-TrustBroker along with its management workflows has been fully designed and documented based on an extensive requirements analysis, i.e., WP1 and WP2 have been completed successfully.
- An Internet-Draft about the GÉANT-TrustBroker core protocol, DAME (dynamic automated metadata exchange), has been submitted to IETF for standardization; several updates based on received feedback, also from the GÉANT JRA3 & SA5 and REFEDS communities, were made; in sum, KPI 2 is fulfilled.
- A prototype / proof-of-concept has been implemented based on Shibboleth. It includes the DAME workflow, the user attribute conversion rules, and web-frontends for both user and IDP/SP administrator interaction, i.e., WP4 has been completed successfully. The source code has been published via GÉANT's Subversion server, which completes the fulfilment of KPI 3.
- A demonstrator has been set up and presented at the 2015 GÉANT Symposium. It will serve as the basis for continued development, and also shows the potential synergies with the two partner open call AAI projects, WoT4LoA and HEXAA.

It must be noted that, in accordance with original planning and as expected and communicated from the beginning on, the Internet-Draft standardization process is not yet finished but must be continued in GN4; also the prototypical implementation of the Shibboleth IDP and SP extensions have full functionality and run stable in the demonstrator environment, but are not yet of production-grade code quality, which is another issue that needs to be worked on in GN4. Based on feedback received from GN3plus JRA3, the IETF working groups, and the REFEDS community, additional requirements for certain use cases and new feature requests have been identified that were not part of the original project plan. While some of them have already been addressed

on a conceptual level, as documented in the project deliverables and milestone documents, others had to be postponed to GN4 in favour of delivering a fully functional prototype implementation and demonstrator.

5.3 Discussion of project impact

Three impact objectives were described in the original Open Call text and refined in the project proposal. This section discusses how these objectives have been met.

The first expected impact was to make the current identity federation model more scalable, particularly in large federations. GÉANT-TrustBroker makes the federated access to services much more scalable and efficient by setting the technical trust basis up in an on-demand and fully automated manner. While traditional inter-federations aggregate a lot of SAML metadata and distribute it to all involved entities, which becomes a processing bottleneck due to its sheer size, GÉANT-TrustBroker ensures that each entity only fetches those entities' metadata where actually active users are involved. The full automation minimizes the workload for the involved IDP and SP administrators, and users can immediately access new services.

The second expected impact was to enable support for currently hard to address e-Science use cases. As GÉANT-TrustBroker does not require cooperating entities to be members of the same federation (e.g., national federation, community federation, or eduGAIN), but instead is especially designed for initial situations in which IDPs and SPs have not yet exchanged each other's SAML metadata, GNTB provides a suitable AAI solution for those e-Science use cases. For example, the GÉANT-TrustBroker demonstrator shows a scenario in which universities from different countries cooperate with an industry partner, which, without GNTB, would not work at all without founding a new community federation first.

The third expected impact was to broaden research horizons in the GÉANT project and seek cooperation with non-GÉANT partners. GÉANT-TrustBroker has been presented to several non-GÉANT partner communities, including IETF, REFEDS, FIM4R, as well as broader audiences at the various conferences, workshops, and scientific symposia that were attended. Intense contact has been established with SAML developer communities, especially for Shibboleth and SimpleSAMLphp. Within GN3plus, ideas and results were communicated to JRA3 and SA5, where they have been integrated into the plans for GN4. In sum, it seems fair to say that the world-wide visibility of the GÉANT activities on Federated Access Management has increased also due to this open call project.

Additionally, GÉANT-TrustBroker is simple to adopt due to its extension nature, i.e., existing IDPs and SPs only have to install the GNTB/DAME extensions and register their organization at the GNTB core service in order to benefit from it. From the user perspective, only familiar federation workflows are used, which means that the user experience and usability stay the same and no additional user training is necessary for IDPs using GÉANT-TrustBroker, but they benefit from immediate access to services that could not be used with their IDP accounts otherwise.

Finally, GÉANT-TrustBroker complements the work done in GN3plus JRA3 and SA5. Therefore, the adoption of the project's results will simplify several use cases identified in JRA3 during GN3plus. As a consequence,

work on GÉANT-TrustBroker will be continued in GN4 as new Task 3 in JRA3 with the primary goal of preparing for pilot operations early in GN4 phase 2.

6 Summary and outlook

This deliverable described the usage as well as the installation and configuration of the GÉANT-TrustBroker proof-of-concept implementation, which consists of the GNTB core service and the Shibboleth extensions for both IDP and SP. Using the GÉANT-TrustBroker demonstrator as an example, it allows others to re-create the necessary development/testing environment and validate the project's results. It therefore accompanies the source code available via GÉANT's TrustBroker Subversion repository.

In sum, this implementation and the other deliverable and milestone documents together show the real-world feasibility and benefits of the GÉANT-TrustBroker approach and that the original project goals have all been fulfilled.

In GN4 phase 1, work on GÉANT-TrustBroker is going to continue in a new task in JRA3. Besides preparing for pilot operations in GN4 phase 2, conceptual work will include the integration of feedback received into the DAME Internet-Draft, the addition of manual approval steps for those IDPs and SPs that want more direct manual control of which entities' SAML metadata is added to their site, the support for entity categories, integration with existing metadata registries such as REEP, and the implementation of GNTB service monitoring and reports. Also, the long-term interplay of eduGAIN and GÉANT-TrustBroker needs to be defined in order to provide the best of both worlds in the future GÉANT service portfolio.

Appendix A Installation and configuration code

In this appendix, the following installation and configuration code is shown:

- Create database of GNTB core service
- `web.xml` for TTP-specific CDS extension
- Example conversion rule for `skypeID`

A.1 Create database for GNTB core service

```
CREATE DATABASE ttp;
CREATE USER ttp IDENTIFIED BY '!!!CHANGE_ME!!!';
GRANT USAGE ON *.* to ttp@localhost IDENTIFIED BY '!!!CHANGE_ME!!!';
GRANT ALL PRIVILEGES ON ttp.* to ttp@localhost;

USE ttp;
CREATE TABLE ruleStatus (
    id INT NOT NULL PRIMARY KEY,
    name VARCHAR(128),
    description VARCHAR(256)
);
CREATE TABLE attributes (
    id INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(128),
    nameFormat VARCHAR(128),
    friendlyName VARCHAR(128)
);
CREATE TABLE ruleDependencies (
    ruleID INT NOT NULL,
    attributeID INT NOT NULL,
    PRIMARY KEY (ruleID, attributeID)
);
```

```
CREATE TABLE convRules (  
    id INT NOT NULL AUTO_INCREMENT PRIMARY KEY,  
    name VARCHAR(64) UNIQUE,  
    description VARCHAR(1024),  
    target INT,  
    owner INT,  
    location VARCHAR(256) UNIQUE,  
    parent INT,  
    status INT,  
    created DATETIME  
);
```

```
ALTER TABLE convRules  
    ADD CONSTRAINT convRules_targetRef  
        FOREIGN KEY (target)  
        REFERENCES attributes(id),  
    ADD CONSTRAINT convRules_parentRef  
        FOREIGN KEY (parent)  
        REFERENCES convRules(id),  
    ADD CONSTRAINT convRules_statusRef  
        FOREIGN KEY (status)  
        REFERENCES ruleStatus(id);  
ALTER TABLE ruleDependencies  
    ADD CONSTRAINT ruleDependencies_ruleRef  
        FOREIGN KEY (ruleID)  
        REFERENCES convRules(id)  
        ON DELETE CASCADE,  
    ADD CONSTRAINT ruleDependencies_attributeRef  
        FOREIGN KEY (attributeID)  
        REFERENCES attributes(id);
```

```
CREATE TABLE users (  
    id INT NOT NULL AUTO_INCREMENT PRIMARY KEY,  
    loginname VARCHAR(64) UNIQUE,  
    username VARCHAR(128) UNIQUE,  
    password VARCHAR(128),  
    salt VARCHAR(128),  
    givenName VARCHAR(256),  
    surname VARCHAR(256),  
    email VARCHAR(64),  
    validated TINYINT,  
    created DATETIME  
);
```

```
CREATE TABLE metadata (  
    id INT NOT NULL AUTO_INCREMENT PRIMARY KEY,  
    entityID VARCHAR(128) NOT NULL,  
    location VARCHAR(256) UNIQUE,  
    comment VARCHAR(1024),  
    owner INT REFERENCES users(id) ON DELETE SET NULL,  
    parent INT REFERENCES metadata(id),  
    created DATETIME  
);  
  
CREATE TABLE organizations (  
    id INT NOT NULL AUTO_INCREMENT PRIMARY KEY,  
    name VARCHAR(128) NOT NULL UNIQUE,  
    displayName VARCHAR(128) NOT NULL UNIQUE,  
    description VARCHAR(1024),  
    url VARCHAR(256)  
);  
  
CREATE TABLE providers (  
    id INT NOT NULL AUTO_INCREMENT PRIMARY KEY,  
    type TINYINT NOT NULL,  
    entityID VARCHAR(256) NOT NULL UNIQUE,  
    description VARCHAR(1024),  
    organization INT REFERENCES organizations(id),  
    metadata INT REFERENCES metadata(id),  
    created DATETIME  
);  
  
CREATE TABLE providerUserRelationship (  
    userID INT NOT NULL REFERENCES users(id),  
    providerID INT NOT NULL REFERENCES providers(id),  
    role TINYINT NOT NULL,  
    PRIMARY KEY(userID, providerID)  
);  
  
CREATE TABLE spIdpRelationship (  
    spID INT NOT NULL REFERENCES providers(id),  
    idpID INT NOT NULL REFERENCES providers(id),  
    created DATETIME,  
    PRIMARY KEY(spID, idpID)  
);  
  
CREATE TABLE providerBlacklist (  
    provider INT NOT NULL REFERENCES providers(id) ON DELETE CASCADE,
```

```

        blockedProvider INT NOT NULL REFERENCES providers(id) ON DELETE CASCADE,
        PRIMARY KEY(provider, blockedProvider)
    );

CREATE TABLE providerWhitelist (
    provider INT NOT NULL REFERENCES providers(id) ON DELETE CASCADE,
    allowedProvider INT NOT NULL REFERENCES providers(id) ON DELETE CASCADE,
    PRIMARY KEY(provider, allowedProvider)
);

CREATE TABLE attributeReleasePolicy (
    sourceProvider INT NOT NULL REFERENCES providers(id) ON DELETE CASCADE,
    targetProvider INT NOT NULL REFERENCES providers(id) ON DELETE CASCADE,
    attribute INT NOT NULL REFERENCES attributes(id) ON DELETE SET NULL,
    PRIMARY KEY(sourceProvider, targetProvider, attribute)
);

CREATE TABLE apiAccessTokens (
    id INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
    owner INT REFERENCES users(id) ON DELETE CASCADE,
    name VARCHAR(32) NOT NULL,
    created DATETIME,
    lastUsed DATETIME,
    token VARCHAR(16)
);

```

A.2 web.xml for TTP-specific CDS extension

```

<?xml version="1.0" encoding="UTF-8"?>
<web-app>
    <context-param>
        <param-name>entityID</param-name>
        <param-value>http://ttp.example.com:8080/discovery/TTP</param-value>
    </context-param>
    <context-param>
        <param-name>metadata</param-name>
        <param-value>/opt/shibboleth-ds/metadata/ttp-metadata.xml</param-
value>
    </context-param>
    <context-param>
        <param-name>cert</param-name>
        <param-value>/opt/shibboleth-ds/cert.crt</param-value>
    </context-param>

```



```

</context-param>
<context-param>
<context-param>
    <param-name>ttpMetadataStorageDirectory</param-name>
    <param-value>/opt/shibboleth-ds/metadata/ttp</param-value>
</context-param>
<context-param>
    <param-name>ttpConversionRuleStorageDirectory</param-name>
    <param-value>/opt/shibboleth-ds/conversionRules</param-value>
</context-param>
<context-param>
    <param-name>dbUser</param-name>
    <param-value>ttp</param-value>
</context-param>
<context-param>
    <param-name>dbPassword</param-name>
    <param-value>!!!CHANGE_ME!!!</param-value>
</context-param>
<context-param>
    <param-name>dbURL</param-name>
    <param-value>jdbc:mysql://127.0.0.1:3306/ttp</param-value>
</context-param>
<context-param>
    <param-name>jdbcDriver</param-name>
    <param-value>org.gjt.mm.mysql.Driver</param-value>
</context-param>
<servlet>
    <servlet-name>ttp_login</servlet-name>
<servlet-class>de.lrz.gntb.dsextension.webinterface.TTPLogin</servlet-class>
</servlet>
<servlet-mapping>
    <servlet-name>ttp_login</servlet-name>
    <url-pattern>/ttp/login</url-pattern>
</servlet-mapping>
<servlet>
    <servlet-name>ttp_userManagement</servlet-name>
<servlet-
class>de.lrz.gntb.dsextension.webinterface.TTPUserManagement</servlet-class>
</servlet>
<servlet-mapping>
    <servlet-name>ttp_userManagement</servlet-name>
    <url-pattern>/ttp/register</url-pattern>
</servlet-mapping>
<servlet-mapping>

```

```

        <servlet-name>ttp_userManagement</servlet-name>
        <url-pattern>/ttp/updateUserInfo</url-pattern>
    </servlet-mapping>
    <servlet>
        <servlet-name>ttp_providerManagement</servlet-name>
        <servlet-class>
de.lrz.gntb.dsextension.webinterface.TTPProviderManagement</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>ttp_providerManagement</servlet-name>
        <url-pattern>/ttp/updateProvider</url-pattern>
    </servlet-mapping>
    <servlet-mapping>
        <servlet-name>ttp_providerManagement</servlet-name>
        <url-pattern>/ttp/addProvider</url-pattern>
    </servlet-mapping>
    <servlet-mapping>
        <servlet-name>ttp_providerManagement</servlet-name>
        <url-pattern>/ttp/deleteProvider</url-pattern>
    </servlet-mapping>
    <servlet>
        <servlet-name>ttp_metadataManagement</servlet-name>
<servlet-class>de.lrz.gntb.dsextension.webinterface.TPMMetadataManagement
</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>ttp_metadataManagement</servlet-name>
        <url-pattern>/ttp/uploadMetadata</url-pattern>
    </servlet-mapping>
    <servlet-mapping>
        <servlet-name>ttp_metadataManagement</servlet-name>
        <url-pattern>/ttp/getMetadata</url-pattern>
    </servlet-mapping>
    <servlet>
        <servlet-name>ttp_providerConnectionManagement</servlet-name>
<servlet-class>
de.lrz.gntb.dsextension.webinterface.TTPProviderConnectionManagement
</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>ttp_providerConnectionManagement</servlet-name>
        <url-pattern>/ttp/providerConnectionManagement</url-pattern>
    </servlet-mapping>
    <servlet>

```

```

        <servlet-name>ttp_providerCRManagement</servlet-name>
<servlet-class>
de.lrz.gntb.dsextension.webinterface.TTPConversionRuleManagement
</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>ttp_providerCRManagement</servlet-name>
        <url-pattern>/ttp/conversionRules</url-pattern>
    </servlet-mapping>
    <servlet-mapping>
        <servlet-name>ttp_providerCRManagement</servlet-name>
        <url-pattern>/ttp/addConversionRule</url-pattern>
    </servlet-mapping>
    <servlet>
        <servlet-name>ttp_accessTokenManagement</servlet-name>
<servlet-class>
de.lrz.gntb.dsextension.webinterface.TTPAccessTokenManagement
</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>ttp_accessTokenManagement</servlet-name>
        <url-pattern>/ttp/generateAPIToken</url-pattern>
    </servlet-mapping>
    <servlet-mapping>
        <servlet-name>ttp_accessTokenManagement</servlet-name>
        <url-pattern>/ttp/revokeAPIToken</url-pattern>
    </servlet-mapping>

    <servlet>
        <servlet-name>ttp_ds</servlet-name>
<servlet-class>
de.lrz.gntb.dsextension.TTPServerServlet
</servlet-class>
    <init-param>
        <param-name>WAYFConfigFileLocation</param-name>
        <param-value>${DS_HOME}/conf/wayfconfig.xml</param-value>
    </init-param>
    <init-param>
        <param-name>WAYFLogConfig</param-name>
        <param-value>${DS_HOME}/conf/logging.xml</param-value>
    </init-param>
    <init-param>
        <param-name>WAYFLogConfigPollFrequency</param-name>
        <param-value>300000</param-value>

```

```

        </init-param>
        <load-on-startup>2</load-on-startup>
    </servlet>

    <servlet-mapping>
        <servlet-name>ttp_ds</servlet-name>
        <url-pattern>/TTP</url-pattern>
    </servlet-mapping>
    <servlet-mapping>
        <servlet-name>ttp_ds</servlet-name>
        <url-pattern>/TTPCRS</url-pattern>
    </servlet-mapping>
    ...
</web-app>

```

A.3 Example conversion rule for skypeID

```

<xsl:stylesheet xmlns:xsl=http://www.w3.org/1999/XSL/Transform
                xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
                xmlns:resolver="urn:mace:shibboleth:2.0:resolver"
                xmlns:ad="urn:mace:shibboleth:2.0:resolver:ad"
                xmlns:dc="urn:mace:shibboleth:2.0:resolver:dc" version="1.0">

    <!--
        Specify output method and elements that should have CDATA encapsulation
        of their text content
    -->
    <xsl:output method="xml" indent="yes"
                cdata-section-elements="ad:Script dc:FilterTemplate"/>
    <!-- Identity template, copies everything as is -->
    <xsl:template match="@*|node()" >
        <xsl:copy>
            <xsl:apply-templates select="@*|node()" />
        </xsl:copy>
    </xsl:template>
    <!--
        Match the root element of an attribute-resolver.xml
    -->
    <xsl:template match="resolver:AttributeResolver">
        <xsl:copy>
            <!-- Copy all elements -->
            <xsl:apply-templates select="@*|node()" />
            <!-- The attribute we are using as base -->

```

```

        <xsl:variable name="schacUserPresenceID"
select="//resolver:AttributeDefinition[resolver:AttributeEncoder/@friendlyName='sc
hacUserPresenceID']"/>
        <!--
        Test if all required AttributeDefinitions were found
        -->
        <xsl:if test="not(string($schacUserPresenceID))">
            <xsl:message terminate="yes">Error: Could not find all
dependencies!</xsl:message>
        </xsl:if>
        <!-- Add new AttributeDefinition -->
        <xsl:element name="resolver:AttributeDefinition">
            <!-- With attributes -->
            <xsl:attribute name="id">skypeID</xsl:attribute>
            <xsl:attribute name="xsi:type">ad:RegexSplit</xsl:attribute>
            <xsl:attribute name="sourceAttributeID">
                <xsl:value-of select="$schacUserPresenceID/@id"/>
            </xsl:attribute>
            <xsl:attribute name="regex">skype:(.*)</xsl:attribute>
            <!-- and elements -->
            <xsl:element name="resolver:Dependency">
                <xsl:attribute name="ref">
                    <xsl:value-of
select="$schacUserPresenceID/resolver:Dependency/@ref"/>
                </xsl:attribute>
            </xsl:element>
            <xsl:element name="resolver:AttributeEncoder">
                <xsl:attribute name="xsi:type">enc:SAML2String</xsl:attribute>
                <xsl:attribute
name="name">urn:oid:1.3.6.1.4.1.7650.6.1</xsl:attribute>
                <xsl:attribute name="friendlyName">skypeID</xsl:attribute>
            </xsl:element>
        </xsl:element>
    </xsl:copy>
</xsl:template>
</xsl:stylesheet>

```

References

- [1] Daniela Pöhn, Stefan Metzger, Wolfgang Hommel:
Integration of Dynamic Automated Metadata Exchange into the SAML 2.0 Web Browser SSO
Profile - draft-poehn-dame-02. I-D (Work in progress).
<https://datatracker.ietf.org/doc/draft-poehn-dame/> [Online: 03-02-2015]
- [2] Shibboleth: IdPDevCustomExtension.
<https://wiki.shibboleth.net/confluence/display/SHIB2/IdPDevCustomExtension>
- [3] Shibboleth: [extensions] Index of /cpp-sp-ext/trunk.
<http://svn.shibboleth.net/view/extensions/cpp-sp-ext/trunk/>
- [4] Shibboleth: DSInstall. <https://wiki.shibboleth.net/confluence/display/SHIB2/DSInstall>
- [5] Shibboleth: IdPInstall. <https://wiki.shibboleth.net/confluence/display/SHIB2/IdPInstall>
- [6] Shibboleth: NativeSPLinuxInstall.
<https://wiki.shibboleth.net/confluence/display/SHIB2/NativeSPLinuxInstall>
- [7] GÉANT: HEXAA – Higher Education eXternal Attribute Authority.
<http://www.geant.net/opencall/Authentication/Pages/HEXAA.aspx>

Glossary

AA	Attribute Authority
AAI	Authentication and Authorization Infrastructure
ACL	Access Control List
API	Application Programming Interface
CDS	Centralized Discovery Service
DAME	Dynamic Automated Metadata Exchange
DOS	Denial of Service
EDS	Embedded Discovery Service
FAM	Federated Access Management
FIM	Federated Identity Management
GNTB	GÉANT-TrustBroker
HEXAA	Higher Education eXternal Attribute Authority

HTML	Hypertext Markup Language
I-D	Internet-Draft
IDP	Identity Provider
IETF	Internet Engineering Task Force
IP	Internet Protocol
JRA	Joint Research Activity
NREN	National Research and Education Network
R&E	Research & Education
REFEDS	Research and Education Federations
Regex	Regular expression
SA	Service Activity
SAML	Secure Assertion Markup Language
SP	Service Provider
SQL	Structured Query Language
SSO	Single Sign On
SVN	Subversion
TTP	Trusted Third Party
URL	Uniform Resource Locator
WoT4LoA	Web of Trust based Level of Assurance enhancement
WP	Work Package
XML	Extensible Markup Language
XSL	Extensible Stylesheet Language
XSLT	XSL Transformation