# The INDIGO-Datacloud Authentication and Authorization Infrastructure

IOP Conf. Series: Journal of Physics: Conf. Series **898** (2017) 102016 doi:10.1088/1742-6596/898/10/102016

# The INDIGO-Datacloud Authentication and Authorization Infrastructure

**A Ceccanti[1], M Hardt[2], B Wegh[2], AP Millar[3], M Caberletti[1], E Vianello[1], S Licehammer[4]**

[1] INFN-CNAF, Viale Berti Pichat 6/2 40127 Bologna (Italy)
[2] KIT, Hermann-von-Helmholtz-Platz 1, 76344 Eggenstein-Leopoldshafen (Germany)
[3] DESY, Notkestraße 85, 22607 Hamburg (Germany)
[4] CESNET, z. s. p. o., Zikova 4, 160 00 Praha 6 (Czech Republic)

E-mail: andrea.ceccanti@cnaf.infn.it

**Abstract.** Contemporary distributed computing infrastructures (DCIs) are not easily and securely accessible by scientists. These computing environments are typically hard to integrate due to interoperability problems resulting from the use of different authentication mechanisms, identity negotiation protocols and access control policies. Such limitations have a big impact on the user experience making it hard for user communities to port and run their scientific applications on resources aggregated from multiple providers. The INDIGO-DataCloud project wants to provide the services and tools needed to enable a secure composition of resources from multiple providers in support of scientific applications. In order to do so, a common AAI architecture has to be defined that supports multiple authentication mechanisms, support delegated authorization across services and can be easily integrated in off-the-shelf software. In this contribution we introduce the INDIGO Authentication and Authorization Infrastructure, describing its main components and their status and how authentication, delegation and authorization flows are implemented across services.

## 1. Introduction

Contemporary distributed computing infrastructures (DCIs) are not easily and securely accessible by scientists. Computing environments are typically hard to integrate due to interoperability problems resulting from the use of different authentication mechanisms, identity negotiation protocols and access control policies. Such limitations have a big impact on the user experience making it hard for user communities to port and run their scientific applications on resources aggregated from multiple providers in different organisational and national domains. INDIGO-DataCloud will provide the services and tools needed to enable a secure composition of resources from multiple providers in support of scientific applications. In order to do so, an AAI architecture has to be defined that satisfies the following requirements:

- Is not bound to a single authentication mechanism, and can leverage federated authentication mechanisms;
- Provides a layer where identities coming from different sources can be managed in a uniform way;
- Defines how attributes linked to these identities are represented and understood by services;

- Defines how controlled delegation of privileges across a chain of services can be implemented;
- Defines how consistent authorization across heterogeneous services can be achieved and provides the tools to define, propagate, compose and enforce authorization policies;
- Is mainly targeted at HTTP services, but can accomodate also non-HTTP services, leveraging token translation.

In this contribution, we will present the work done in the first year of the INDIGO project to address the above challenges. In particular, we will introduce the INDIGO AAI architecture, its main components and their status and demonstrate how authentication, delegation and authorization flows are implemented across services.

## 2. INDIGO enabling use cases

The INDIGO AAI architecture primary objective is to fulfill the needs and requirements on authentication and authorization expressed by INDIGO user communities. The initial set of requirements of these user communities has been collected, organized and summarized [1, 2], and two generic use cases have emerged to drive the definition and development of the INDIGO platform.

The first generic user scenario is termed "Application portal as a service". In this scenario, computing applications are stored by the application developers in repositories as downloadable images (in the form of VMs or containers). Such images can be accessed by users via a portal, and require a back-end for execution; in the most common situation this is typically a batch queue.

The number of nodes available for computing should increase (scale out) and decrease (scale in), according to the workload. The system should also be able to do Cloud-bursting to external infrastructures when the workload demands it[1]. Furthermore, users should be able to access and analyse reference data, and also to provide their local data for the runs.

The second generic user scenario is described by scientific communities that have a coordinated set of data repositories and software services used to access, process and inspect the data. The processing is typically interactive, requiring access to a console deployed on the data premises [2].

## 3. INDIGO AAI requirements

To support the above user scenarios, the INDIGO AAI must satisfy the following requirements:

- Support for heterogeneous authentication mechanisms: The INDIGO AAI should not be bound to a single authentication technology, but should instead integrate and support federated authentication mechanisms like SAML [5] and OpenID Connect [6] and support X.509 [7] and username/password authentication;
- Identity harmonization and traceability: the INDIGO AAI should provide the ability to link multiple authentication credentials to a single INDIGO user profile, which provides a persistent and unique user identifier;
- Access to identity and authentication information: the INDIGO AAI must expose information regarding the user identity (e.g., presence or strength of the authentication mechanism used to prove the identity) and other attributes (e.g., group membership) to relying services so that authorization and accounting can be implemented taking into account this information;

[1] Cloud bursting is an application deployment model in which an application runs in a private cloud or data center and bursts into a public cloud when the demand for computing capacity spikes.

- Delegation: the INDIGO AAI must support constrained delegation, where a user can delegate part of his/her rights to an agent or service that acts on the user behalf. The service must be able to further delegate a subset of these rights to other services down the line, with the explicit or implicit approval of the user, to fulfill user requests; the INDIGO AAI must provide mechanisms and tools to safely define trusted delegation chains (i.e., which services can take part in a delegation chain, and which set of privileges can be delegated across the chain);
- Provisioning: the INDIGO AAI must provide the ability to provision, manage and deprovision identity information to relying services, to enable, for instance, local or service-specific account management;
- Virtual Organization/Collaboration management, registration and enrollment: the INDIGO AAI must provide the ability to define a VO/collaboration to group together users from distinct institutions sharing a common research goal, giving tools to manage the organization internal structure and registration flows;
- Integration and token translation: the INDIGO AAI must provide the ability to integrate with services that cannot be modified to directly support INDIGO AAI, for instance providing token translation functionality.

## 4. Authentication and Identity

The INDIGO AAI needs to accommodate heterogeneous authentication mechanisms and integrate with SAML identity federations like eduGAIN [8] and social identity providers (e.g., Google [10] or GitHub [11]).

In order to reduce the complexity of AAI integration at relying services and have the ability to decorate identities as provided by the upstream authentication mechanisms (SAML, OpenID Connect, X.509) with additional attributes, we developed a service whose responsibility is to authenticate users and agents (via the supported authentication mechanisms) and expose the authentication information to relying services through standard OpenID Connect interfaces. This service is the Login Service component of the INDIGO Identity and Access Management (IAM) service [12].

This centralization of authentication responsibility in a single service, depicted in Figure 1, is an emerging architectural pattern (the *IdP-SP-Proxy* pattern [9]), which provides several advantages:

- A single point of control for authentication for all services: this approach simplifies auditing and control on enabled authentication mechanisms. On the other hand, from an operational point of view, the service has to be engineered in a way that allows it to scale horizontally to handle requests from services and not represent a bottleneck or single point of failure.
- Simplified registration in identity federations like eduGAIN: with this approach only one service, the Login Service, needs to be registered in the federation, not each individual service provider. This greatly simplifies support for identity federation at relying services.
- The ability to decorate the identity information obtained from upstream authentication mechanisms with additional attributes, like group membership attributes, roles, and, more importantly, a unique and non-reassignable persistent identifier that can be used to track down user activity in a way that is orthogonal to the authentication mechanism used in a given session.
- Natural support for guest users, i.e. users that do not have external credentials (i.e., are not part of any identity federation or do not want to use a personal social account), via the creation of local IAM Login Service username/password credentials.

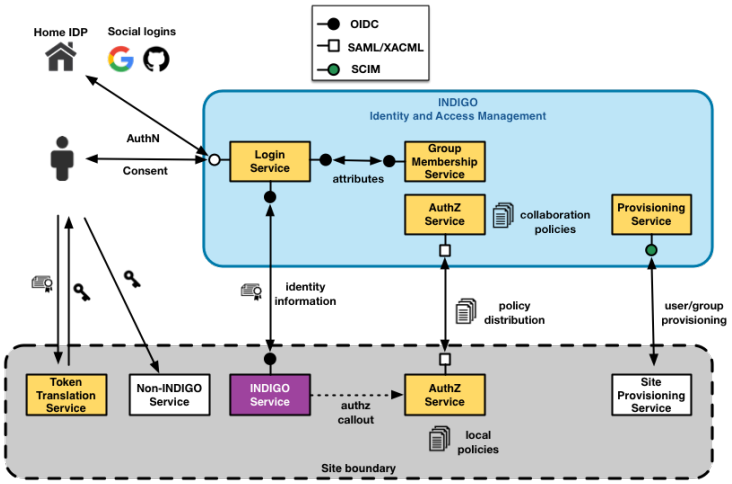


**Figure 1.** The INDIGO AAI overview. The INDIGO IAM Login Service is responsible for user and agent authentication, supporting several authentication mechanisms and exposing identity information through standard OpenID Connect interfaces. This approach simplifies registration in identity federations like eduGAIN and integration in relying services.

### *4.1. OpenID Connect as the INDIGO identity layer*

The INDIGO AAI identity layer is based on OpenID Connect. OpenID Connect is a Single Sign-On (SSO) protocol and identity layer recently standardized by the IETF, built on top of the OAuth 2.0 protocol framework [13], from which it inherits support for delegated authorization and offline access (that we discuss in more detail in section 6). Despite being a relatively new standard, OpenID connect is already widely adopted by many leading companies like Google, Microsoft, AOL, PayPal and LinkedIn.

OpenID Connect "allows clients of several types, including Web-based, mobile, and JavaScript clients, to request and receive information about authenticated sessions and end-users"[2] in an interoperable and REST-like manner. Moreover, it provides a flexible trust model that accommodates the dynamic nature of our target infrastructure, as it standardizes protocols for dynamic registration [18] and identity provider discovery [17].

The main benefits of choosing OpenID Connect as the identity layer are:

- simplified integration in relying services, especially when compared to the SAML-based alternative;
- native support for delegated authorization on HTTP services (via OAuth, on which OpenID Connect is based);
- native support for long running computations, which is a common requirement for batch scientific computations;
- ability to accommodate heterogeneous authentication mechanisms (the OpenID Connect specification does not constrain how a user/agent should be authenticated);
- strong adoption of the technology in the industry, and availability of open source client and server libraries;
- support for mobile clients.

[2] from http://openid.net/connect/faq/

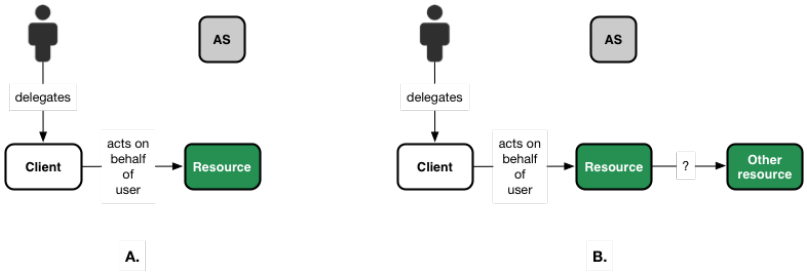


**Figure 2.** OAuth delegation vs OAuth chained delegation

## 5. Authorization

Authorization, in the INDIGO AAI, follows the OAuth 2.0 delegated authorization model [13]: only agents presenting a valid and trusted OAuth access token are granted access to INDIGO services. Access tokens are obtained by client applications from the INDIGO IAM service and provide access to identity information (e.g., group membership and other attributes) and other authorization information (e.g., OAuth scopes) (see Figure 1).

Fine-grained, powerful and distributed attribute-based authorization is implemented by integrating the Argus authorization service [4] with the INDIGO AAI identity layer. This provides policy distribution and centralized XACML policy management.

## 6. Delegation and offline access

OAuth 2.0 was designed to solve the problem of a delegated access to resources across services, mediated by an authorization server, as shown in Figure 2A.

In scientific computing there are scenarios where a service, in order to satisfy a client request, needs to access resources hosted by other downstream services on behalf of the user, as shown in Figure 2B. In these scenarios, such a service acts both as an OAuth client application and an OAuth resource server. In our token model, access tokens are bearer tokens, so the first service could simply use the access token received from the client to interact, on behalf of the user, with the downstream service. There are, however, situations in which just using the received access token against the downstream service is not possible, like for instance if the token audience was scoped to be valid only on the first resource server, or if the token does not grant scopes or privileges specific to the target service. Moreover, the resource server could need the ability to act on behalf of the user for an unbounded amount of time (e.g., to implement long-running computations), not limited by the validity of the received access token.

To support controlled chained delegation across services, in which a component can act both as a service and as a client for another downstream service, the INDIGO IAM implements the OAuth token exchange draft standard [15]. The token exchange is especially useful to implement controlled delegation of offline access rights across applications; i.e., the ability to execute tasks on behalf of a user while the user is not connected.

## 7. Provisioning

Provisioning is when a service learns about users before their first use of the service; deprovisioning is the reciprocal notification: a service learns about users who will never make subsequent use of the service. Provisioning allows for allocation of resources, which may be crucial for some services. Deprovisioning allows services to apply policies against established state; e.g., running virtual machines or stored data.

INDIGO IAM leverages version 2.0 of the standard System for Cross Domain Identity Management (SCIM) [16] to implement identity provisioning, deprovisioning and management.

The SCIM APIs provide a means to propagate identity and group information to relying services; for example, to implement dynamic account creation and other resource lifecycle management at various levels of the INDIGO infrastructure depending on events related to user identity status.

Current work involves adding support for the asynchronous notification of information related to user identity and group information to interested relying services. This mechanism is implemented leveraging the SCIM event

notification standard [22] by the *SCIM notification hub*, a standalone component that is linked to the IAM. The event notification hub takes care of registering interested listeners and ensuring the successful delivery of event notifications.

## 8. Identity harmonisation and account linking

Users often own multiple accounts; for example, one or more Google accounts, one or more institutional accounts (e.g., an INFN and a CERN one) and local UNIX accounts at one or more computing centers. Such accounts often support one authentication technology, such as OpenID Connect, SAML, username and password or SSH keys.

The goal of account linking is to give a user the ability to link multiple accounts (that possibly use different authentication technologies) to a single identity.

Identity harmonisation is the process of providing consistent authorization and auditing across services based on this linkage information, so that, for instance, users' activity could be mapped to the same UNIX local account when they accesses a site through different services possibly relying on different authentication mechanisms (e.g., SSH keys, X.509 certificates, OpenID Connect tokens or SAML assertions).

Account linking, in the INDIGO AAI, is implemented at the INDIGO IAM level, where a user can link several authentication credentials (OpenID Connect and SAML accounts, but also X.509 certificates and SSH keys) to a single user identity.

Access to this account linking information is then exposed to services via SCIM provisioning APIs. The Identity Harmonisation Service [21] leverages this information to implement a mapping to local UNIX accounts so that users get consistent access to their data stored locally in a way that is independent from the authentication mechanism used.

## 9. Token translation

Token translation is typically used to integrate services and tools that rely on different credentials and authentication/authorization mechanisms. In INDIGO, we propose a two-layered approach for token translation.

First, at the IAM level, different authentication credentials are harmonized to a single user account. The account information, and related privileges, is then exposed to relying services via OAuth/OpenID Connect tokens and endpoints. This approach allows to easily integrate a growing ecosystem of solutions and services, and provides other benefits (such as support for delegated authorization) to support our use cases.

In order to integrate services that do not directly support OpenID Connect, a second layer of token translation is introduced. Its purpose is to translate OpenID Connect tokens to other credentials.

For example, a service that requires an Amazon S3 [23] key, as access credential, can be supported by a specific service responsible for the translation of an OpenID connect authentication or access token to the S3 key.

The scope of the credential issued by this token translation step and the deployment of the service that will accept this credential determine the nature of the deployment of the Token Translation Service (TTS) that does the translation.

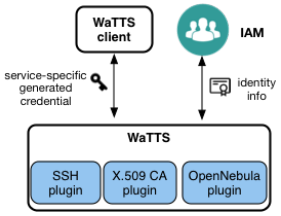


**Figure 3.** The INDIGO Token Translation Service (WaTTS) architecture

There are scenarios, e.g., when generating SSH keys on-demand, where the TTS will be deployed at a site level; i.e., close to the service secured by the issued credentials. The on-demand generation of X.509 certificates, on the other hand, could require the deployment of a central TTS, trusted by several sites that will accept the generated certificates as valid credentials for some services.

### *9.1. WaTTS: The INDIGO Token Translation Service*

The INDIGO Token Translation Service, named WaTTS [20], is used to translate OpenID Connect tokens into service specific credentials, that are then used when authenticating to services that have no direct support OpenID Connect.

WaTTS follows a client-server architecture (see Figure 3): the credential translation service is exposed through a REST API, which can be accessed directly by client applications, via a command line client or through a web application. This web application, which relies on the REST API, provides a convenient interface for users.

The server side part of WaTTS is composed of two main components: the WaTTS core and the plugin runner. The core handles OpenID Connect authentication and authorization and implements the REST API for the token translation service, while the plugin runner manages and executes the plugins that implement the credential translation.

At the time of writing, WaTTS provides support for the on-demand generation of S3 keys, SSH keys, X.509 credentials and OpenNebula credentials. Its pluggable architecture allows WaTTS to support other types of credentials by writing a plugin specific to that credential.

## 10. Integrated applications

The INDIGO AAI has already been successfully integrated in almost all INDIGO services and in many external software components that INDIGO relies on (e.g., OpenStack [24]). The key for this successful integration is the adoption of the OpenID Connect standard. While OpenID Connect is widely supported, where that support is lacking, integration is still easy because it relies on technologies (e.g., JSON) that are already in use. This ensures the availability of libraries and SDKs in several programming languages. In the next section we describe in more detail how the Openstack integration works.

### *10.1. OpenStack integration*

Starting from the native support for OpenID Connect implemented in OpenStack Keystone [25], a native OpenStack Mitaka deployment has been successfully integrated with IAM in order to support federated authentication and group-based authorization. The flow is as follows:

- a user that wants to access OpenStack IaaS service points his browser to the OpenStack dashboard URL. At this stage the user can choose to authenticate via INDIGO IAM credentials.

- The user is redirected to INDIGO IAM for authentication via a standard OpenID Connect authentication flow. Then, the user chooses how to authenticate (e.g., via home institution IdP or Google credentials) at the IAM. After successful authentication the user is redirected back to OpenStack;
- OpenStack Keystone is configured to grant access to specific tenants taking into account group membership information derived from the authentication token returned by the INDIGO IAM. For example, users in group CMS can access the CMS tenant.
- The same integration works also for the OpenStack command line client, through the OAuth password credential authentication flow [14] supported by the INDIGO IAM, that allows a user to directly authenticate with username and password.

## 11. Conclusions and future work

In this work we have introduced the main concepts behind the INDIGO Authentication and Authorization Infrastructure, the problems that it solves and its main components (the INDIGO Identity and Access Management (IAM) service and the INDIGO Token Translation Service (WaTTS)). The INDIGO AAI already represents a production-ready solution based on modern standards for securing access to infrastructure and services in support of scientific computing.

In the final months of the INDIGO project we will focus on supporting the integration of the AAI in all the target INDIGO use cases and on strengthening the code base and improving documentation.

## References

[1] INDIGO-DataCloud D2.1: Initial Requirements From Research Communities https://www.indigo-datacloud.eu/documents/initial-requirements-research-communities-d21
[2] INDIGO-Datacloud: foundations and architectural description of a Platform as a Service oriented to scientific computing https://arxiv.org/abs/1603.09536
[3] The VOMS website http://italiangrid.github.io/voms
[4] The Argus authorization service website http://argus-authz.github.io
[5] The Security Assertion Markup Language (SAML) Wikipedia page https://en.wikipedia.org/wiki/SAML_2.0
[6] The OpenID Connect website http://openid.net/connect/
[7] The X.509 Wikipedia page https://en.wikipedia.org/wiki/X.509
[8] eduGAIN interfederation http://www.geant.org/Services/Trust_identity_and_security/eduGAIN
[9] First draft of the AARC Blueprint architecture https://aarc-project.eu/wp-content/uploads/2016/08/MJRA1.4-First-Draft-of-the-Blueprint-Architecture.pdf
[10] The Google Identity Platform https://developers.google.com/identity/
[11] The Github OAuth API reference https://developer.github.com/v3/oauth/
[12] The INDIGO IAM Github repository https://github.com/indigo-iam/iam
[13] The OAUTH 2.0 authorization framework https://tools.ietf.org/html/rfc6749
[14] The OAUTH 2.0 resource owner credential authentication flow https://tools.ietf.org/html/rfc6749#section-4.3
[15] The OAuth 2.0 Token Exchange draft standard https://tools.ietf.org/html/draft-ietf-oauth-token-exchange-07
[16] The System for Cross Domain Identity Management websitehttp://www.simplecloud.info/
[17] OpenID Connect discovery specification http://openid.net/specs/openid-connect-discovery-1_0.html
[18] OpenID Connect dynamic registration http://openid.net/specs/openid-connect-registration-1_0.html
[19] The eduGAIN website https://technical.edugain.org
[20] The WaTTS Github repository https://github.com/indigo-dc/tts
[21] Identity Harmonisation Service https://github.com/indigo-dc/identity-harmonization
[22] SCIM Event Notification https://tools.ietf.org/html/draft-hunt-scim-notify-00
[23] Amazon S3 https://aws.amazon.com/s3
[24] OpenStack website https://www.openstack.org/
[25] OpenStack Keystone https://docs.openstack.org/developer/keystone/