UNIfied identiTY management

Krzysztof Benedyczak
ICM, Warsaw University

Outline

- The idea
- Local database
  - Groups, Entities, Identities and Attributes
  - UNITY Authorization
- Local authentication
  - Credentials
- External authentication
- Endpoints and protocols
- Cross-cutting features
- History & **Status**
- Alternative solutions

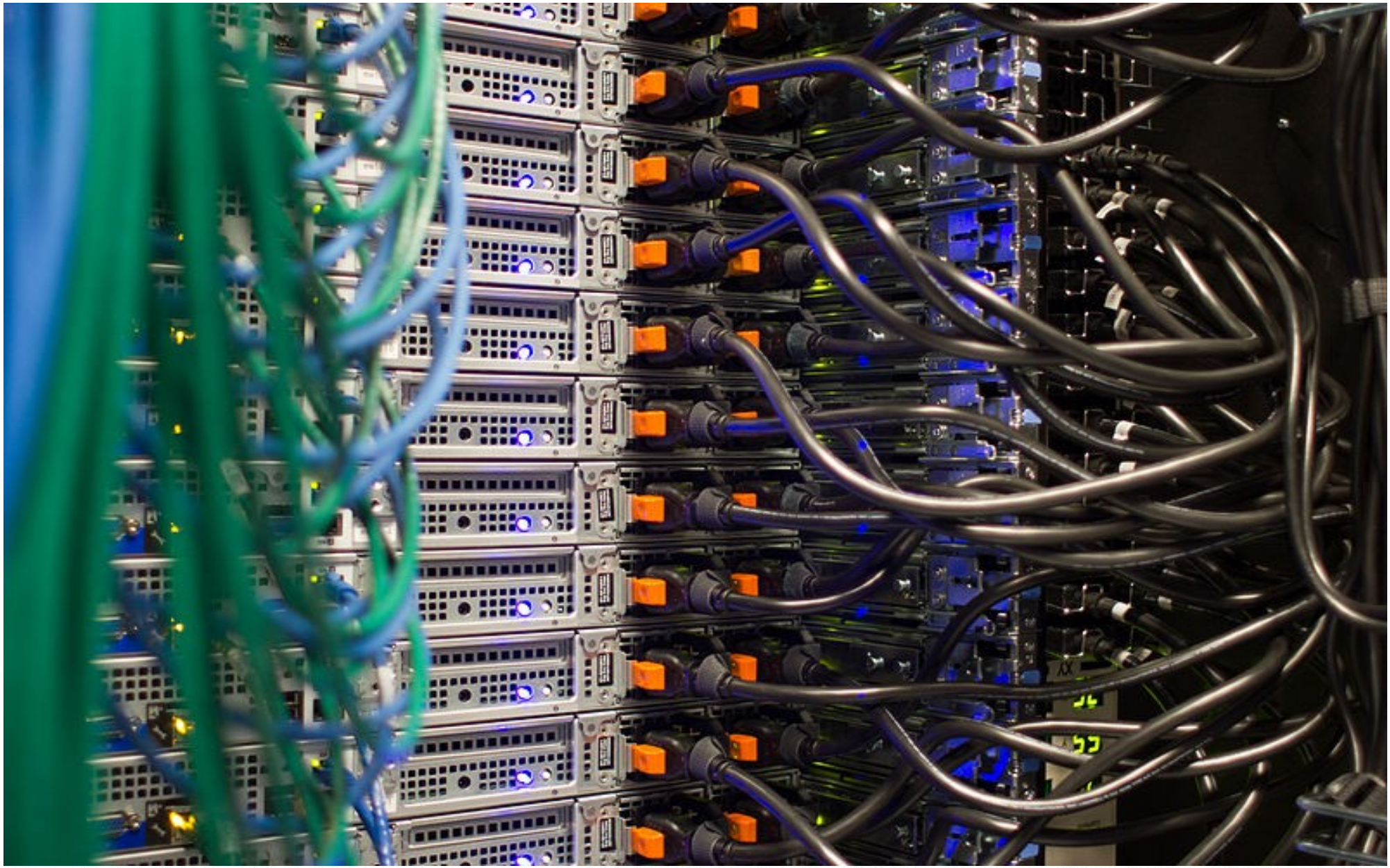
Notice

- The slides are an updated version of the materials shown during the Unity & UNICORE workshop which took place on 04.07.2013 in Forschungszentrum Jülich.
- In few cases the slides cover a planned functionality. See the Status slide or the latest release documentation to check the actually supported features.

The goal is to

...connect them...

...with this...



...taking care of:

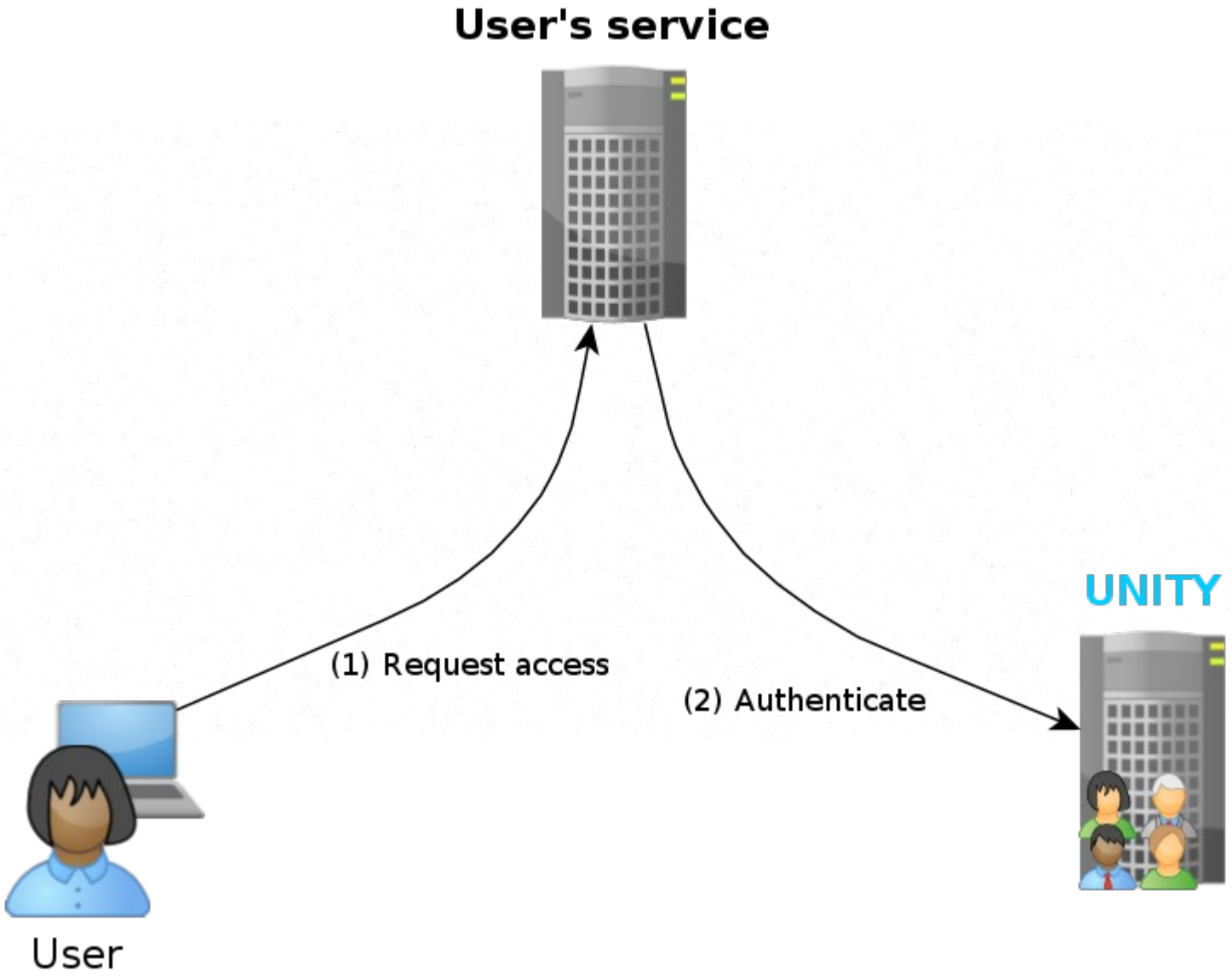
- Authentication
  - who is who?
  - covering wide range of mechanisms & security levels
- Federations
  - from where do they come from?
- Authorization levels
  - The relaying systems must be able to decide **easily** who has an access.
- Privacy
  - from none to paranoid
- *Not forgetting about the crowd's **size**, amount of cables and **quality**.*

The idea

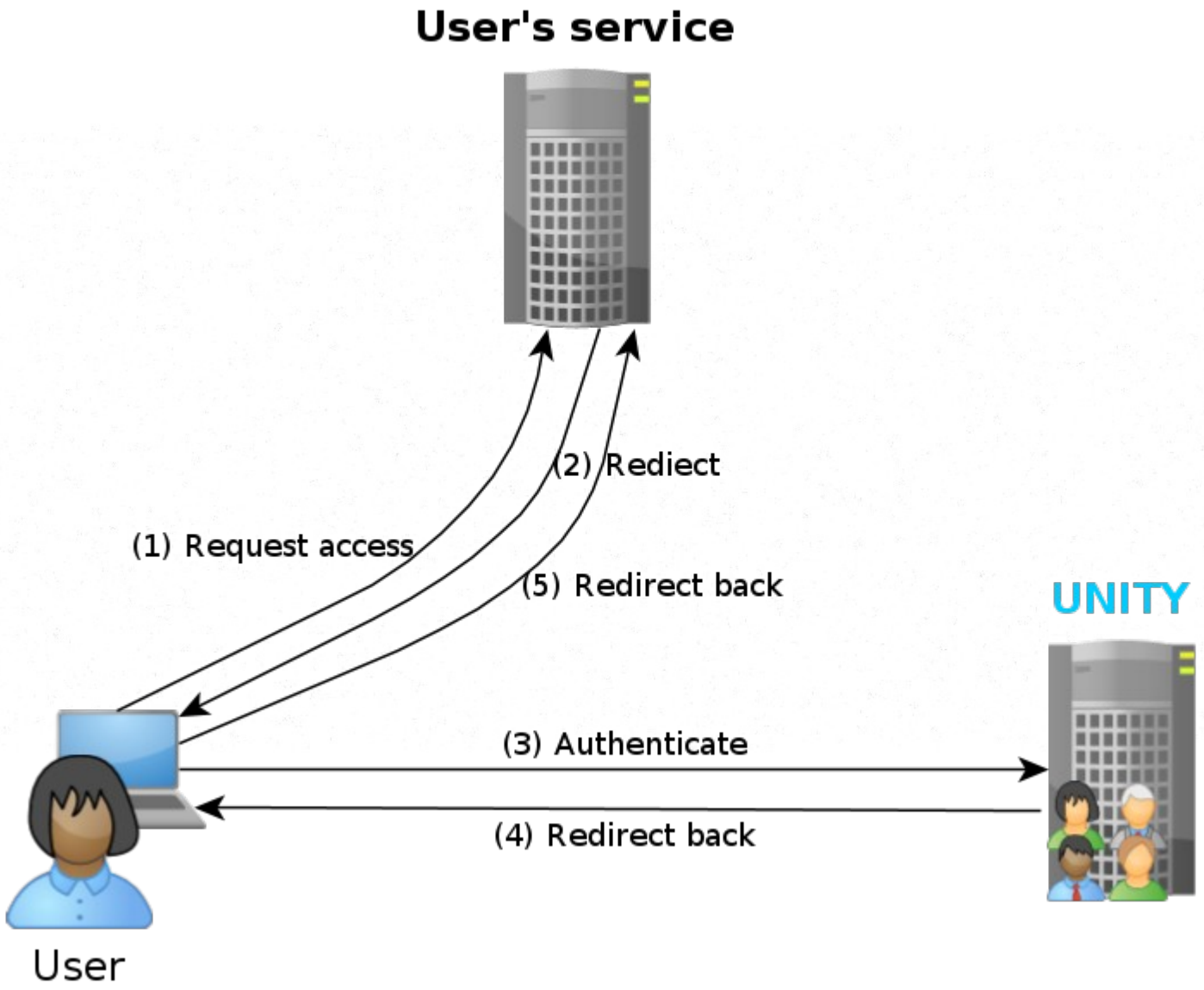
- Cloud approach: **Identity Management As a Service**
  - with attributes management and authentication included.
- Multiple authentication protocols supported
  - easy integration with various consumers/clients.
- Ability to outsource credentials (and attributes) management to a 3rd party service.
  - Again multiple upstream protocols supported.
  - UNITY becomes a bridge (protocol translation)...
  - ... and a hub (single service aggregating various IdM systems).

Core concepts of UNITY

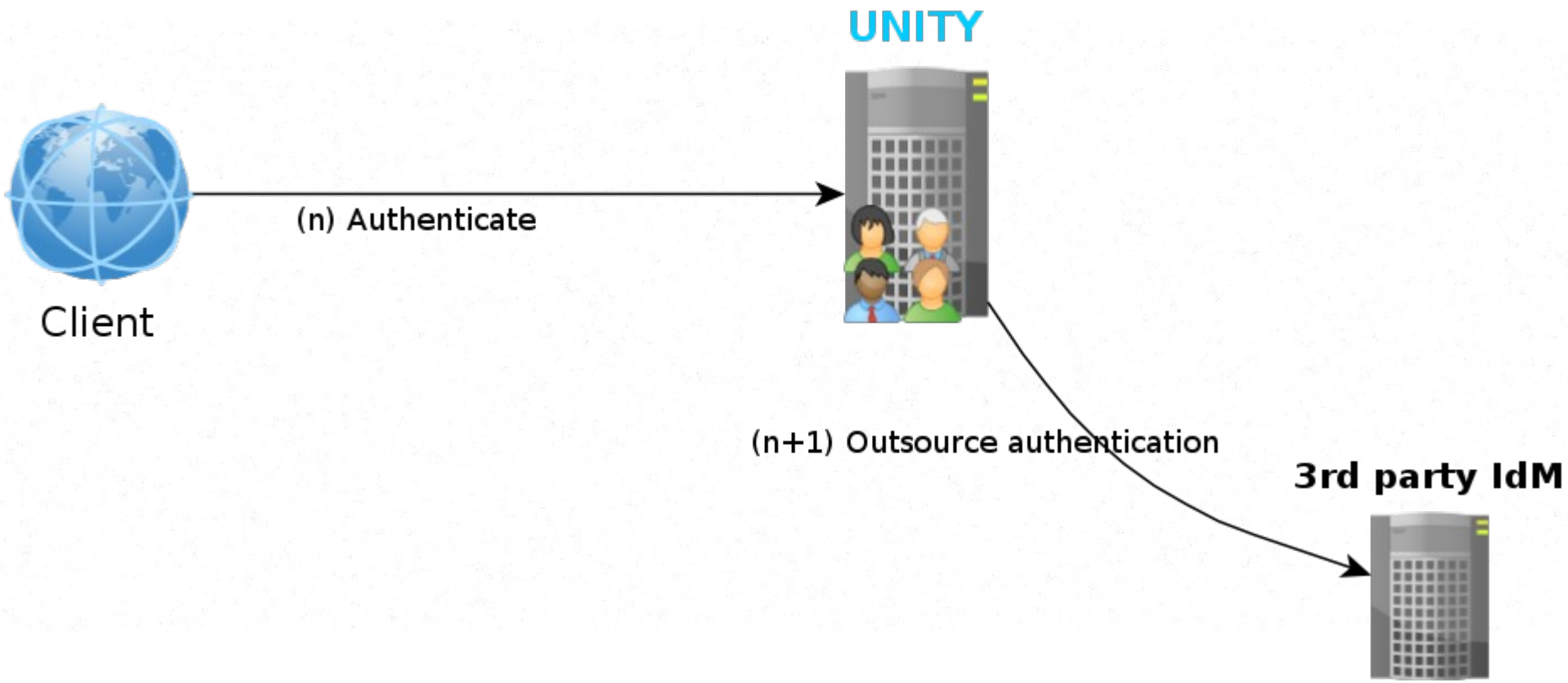
Basic use case



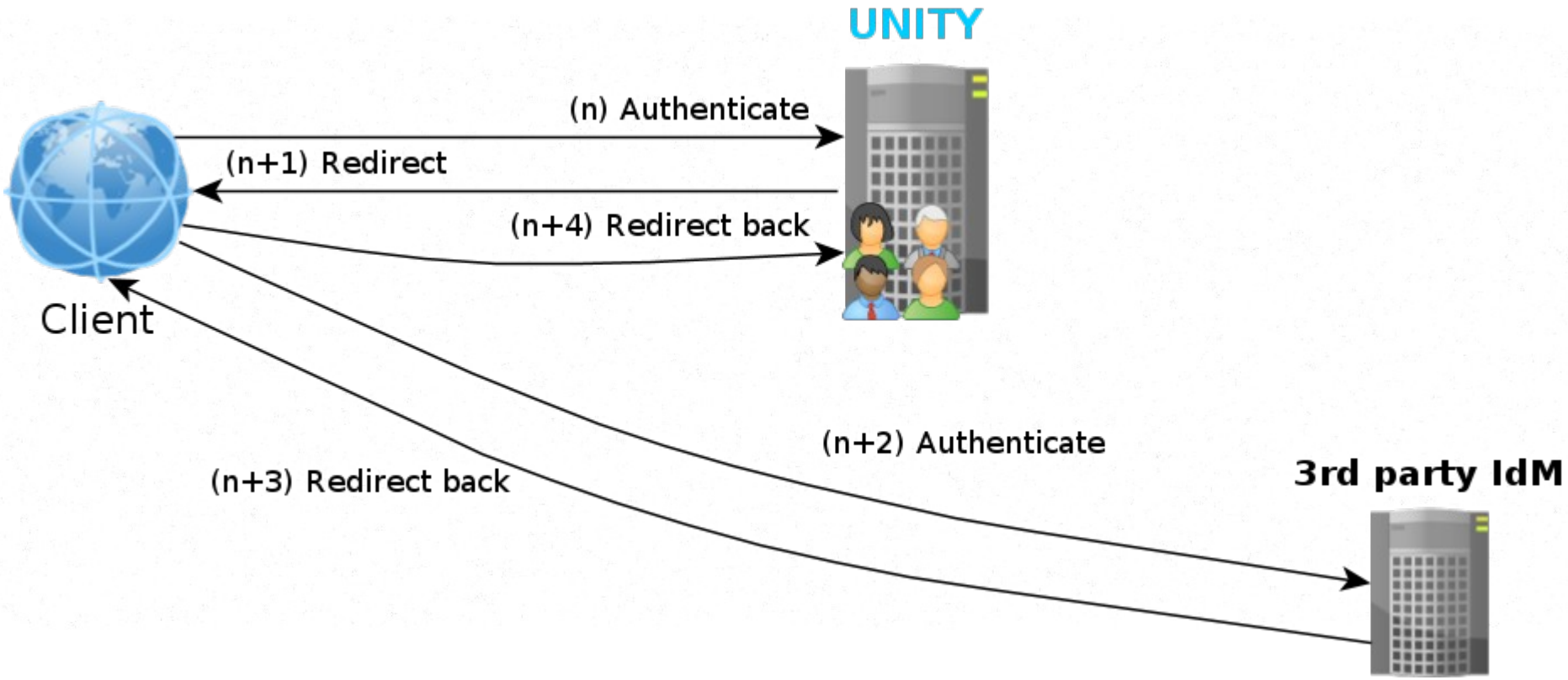
Redirected web authentication



UNITY can use 3rd party IdM



As well as redirect to it



True extensibility

- Flexible system requires large number of pluggable components:
  - credentials,
  - authentication protocols,
  - attribute types,
  - ... and many more

- Extensibility is the fundamental concept.
- From persistence to management interface.
- Start time pluggability (not runtime).
- Modular design.
  - For instance *all* UNICORE specific elements are in separate module, administration UI in another, etc.

Members

- Each distinguished member is an Entity.
  - Typically a user,
  - or some software agent/service, acting as UNITY client.
- Each entity can have multiple representations called Identities.
  - For supporting various authentication protocols&formats,
  - to have aliases.
- Each Identity has its type, which defines the syntax, comparison rules, etc.
  - Types are pluggable.

- Entities and Identities are separate artifacts in UNITY, in contrast to e.g. LDAP.

Members

- Each distinguished member is an Entity.
  - Typically a user,
  - or some software agent/service, acting as UNITY client.
- Each entity can have multiple representations called Identities.
  - For supporting various authentication protocols&formats,
  - to have aliases.
- Each Identity have its type, which defines the syntax, comparison rules, etc.
  - Types are pluggable.

- Entities and Identities are separate artifacts in UNITY, in contrast to e.g. LDAP.

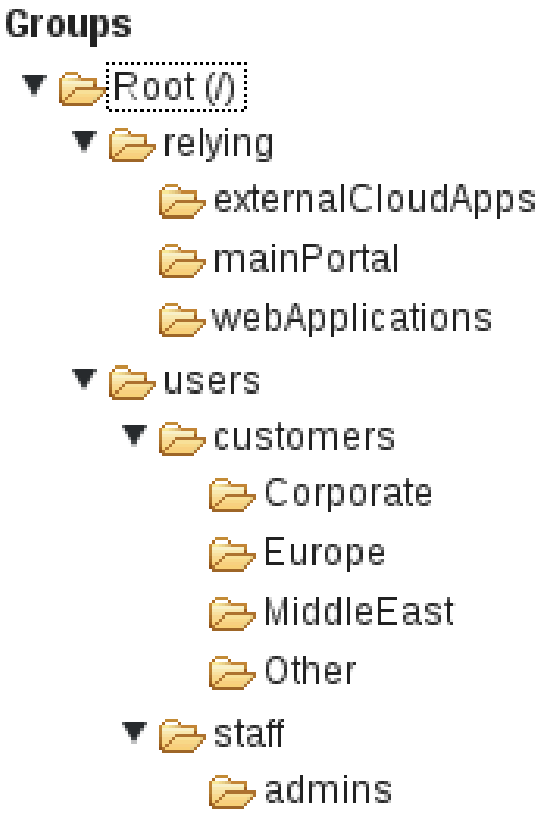
Examples of identity types

- X.500 name (distinguished name)
  - CN=Krzysztof Benedyczak,O=ICM,O=GRID,C=PL
- User name
  - kbenedczak
- OpenID identifier
  - http://kb.myopenid.example.com
- E-mail address
  - kbenedyczak@mail.example.com
- Persistent, anonymous identifier
  - 9ab8c946-aa89-23fd-2231-9221a7da2756
- Transient, anonymous identifier
  - 9ab8c946-bb89-23fd-2231-9221a7da2756

Groups

- Groups allow for logical categorization of Entities.
- Additionally groups are the fundamental concept for administration automation:
  - members can have attributes assigned automatically,
  - groups are used to control access and to configure advanced protocol-specific settings for multiple users at once.
- Groups are hierarchical:
  - every child group member must be a member of its parent group,
  - every user with some access privileges in parent group, has the same privileges in all child groups.

Group examples



Attributes

- Carries arbitrary information about entities.
- Attribute has a name and from 0 to many values.
- Values of an attribute are typed.
  - Value types are pluggable.
  - Value type defines the value's syntax, the 'external' representation, comparison rules and storage form.
  - Examples: string, enumeration, JPEG image, integer number.

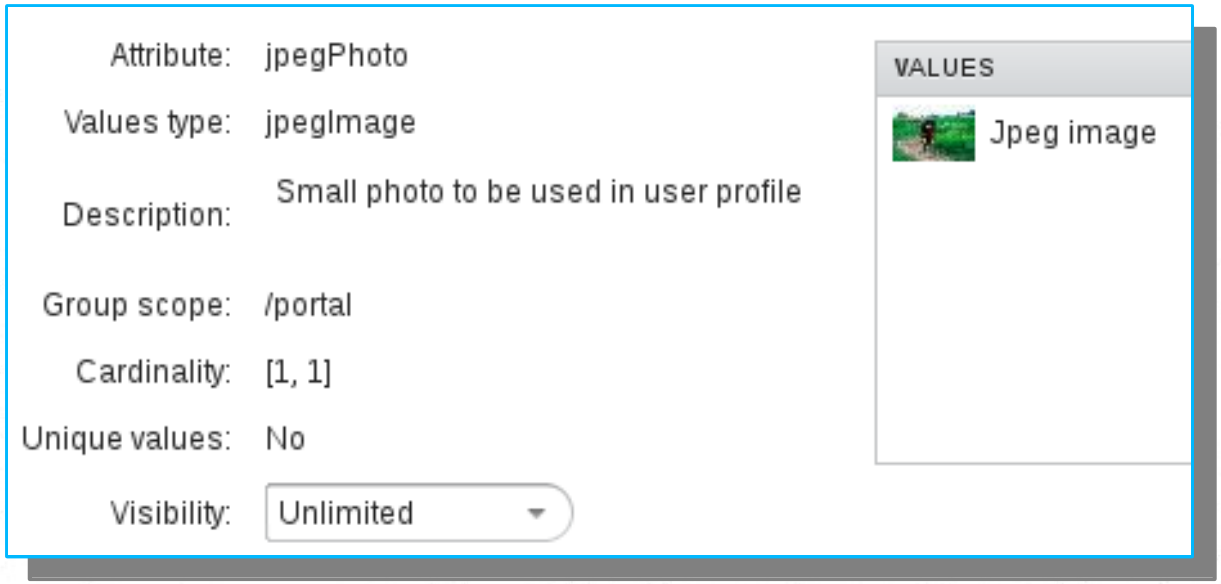
- Attributes are assigned in a group.
  - It is only possible to have an attribute in a context of some group.
- Group attribute statements can be used to automatically propagate attributes between groups.

Attribute types and classes

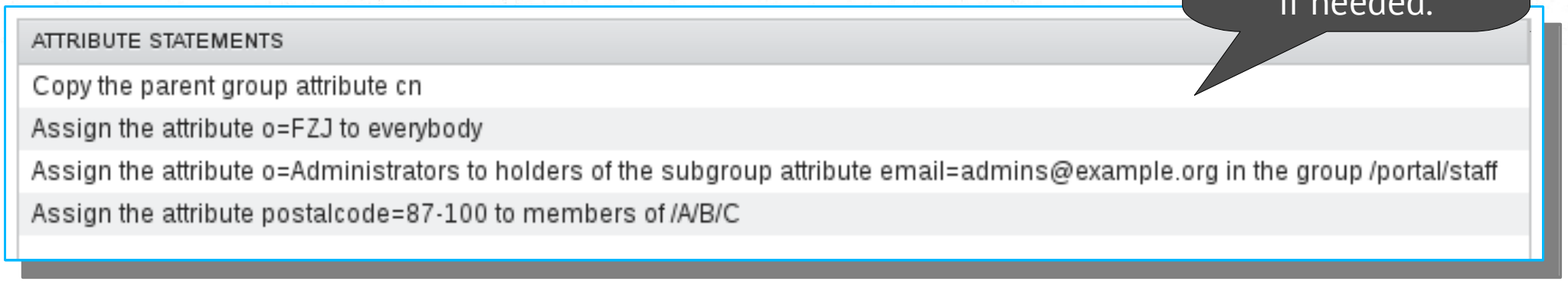
- Each attribute is defined in the system by **attribute type**, defining:
  - name,
  - values type,
  - values cardinality,
  - whether values must be unique,
  - whether the attribute can be exposed to clients,
  - whether the attribute in general can be self modifiable.
- Attribute types are grouped into **attribute classes**.
  - Classes are assigned to entities and define allowed and mandatory attributes.

Attribute examples

Regular attribute



Group */portal* attribute statements



More conditions might be plugged if needed.

Special attributes

- UNITY stores a lot of special/internal information about managed entities.
  - configured credentials, credential status, ...
  - Such information is also modeled with attributes.
- Some of those special attributes can not be updated with standard attribute management methods.
  - Those attributes are fully handled internally.
- There are also predefined attributes, with a special meaning for the system.
  - E.g. authorization role.
  - It is possible to update the such attributes, but attribute type can not be modified.

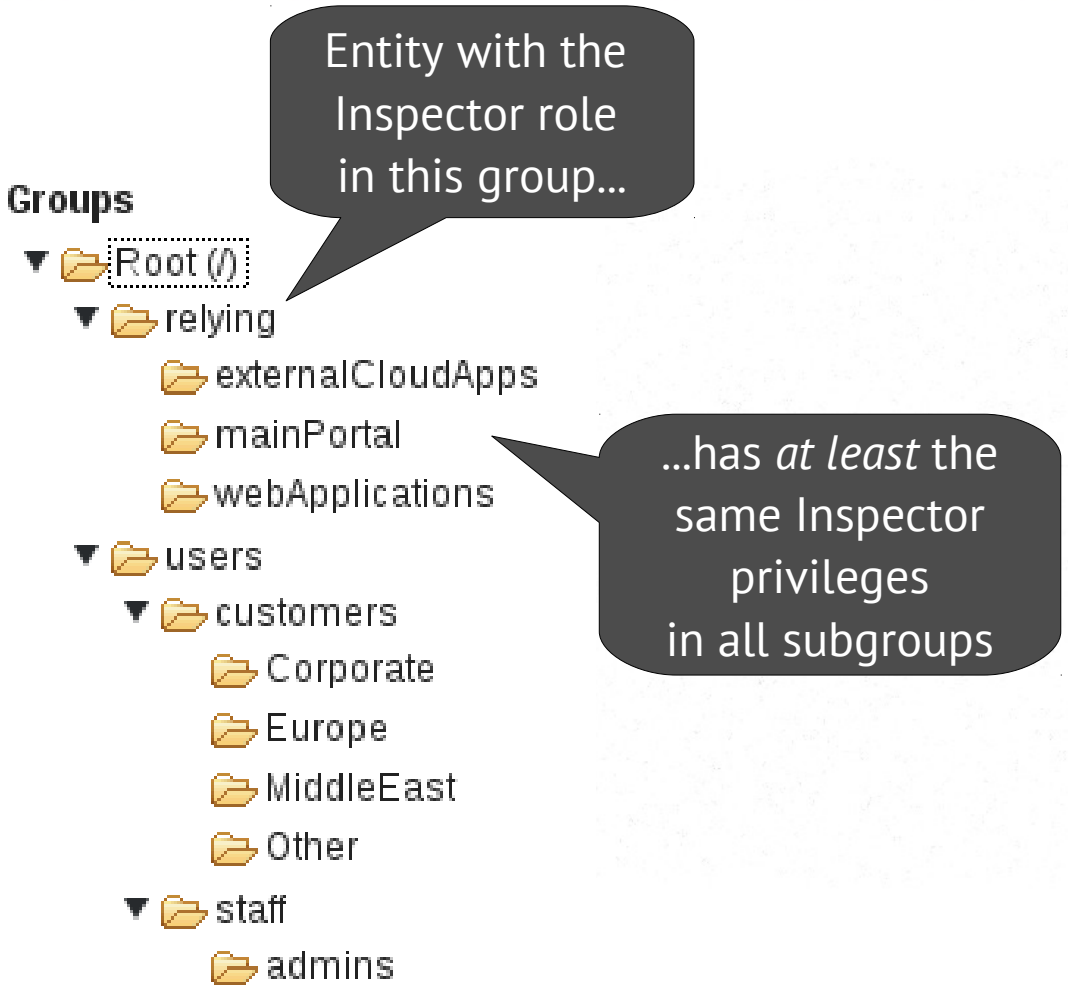
Local authorization

- Simple, yet flexible.
- Each engine operation requires a set of capabilities.
  - Actually a single one.
  - Capabilities are predefined and association with operations is fixed.
- Authorization role has a set of capabilities assigned.
  - Roles are predefined and fixed. No need to manage them.
  - In future, if needed (unlikely) it can be made editable.
- Administrator assigns roles to entities using the sys:AuthorizationRole attribute.
  - System-wide operations require a capability in the root group.
  - Group-wide operations require a capability in the affected group(s).
- It is guaranteed that all capabilities of the parent group are propagated downwards.

Local authorization example

- *System Manager*
  - System manager with all privileges.
- *Contents Manager*
  - Allows for performing all management operations related to groups, entities and attributes.
- *Inspector*
  - Allows for reading entities, groups and attributes. No modifications are possible.
- *Regular User*
  - Allows owners to read the basic system information and retrieval of information about themselves. Selected attributes might be marked as editable for such users.
- *Anonymous*
  - Allows for minimal access to the system.

Local authorization inheritance

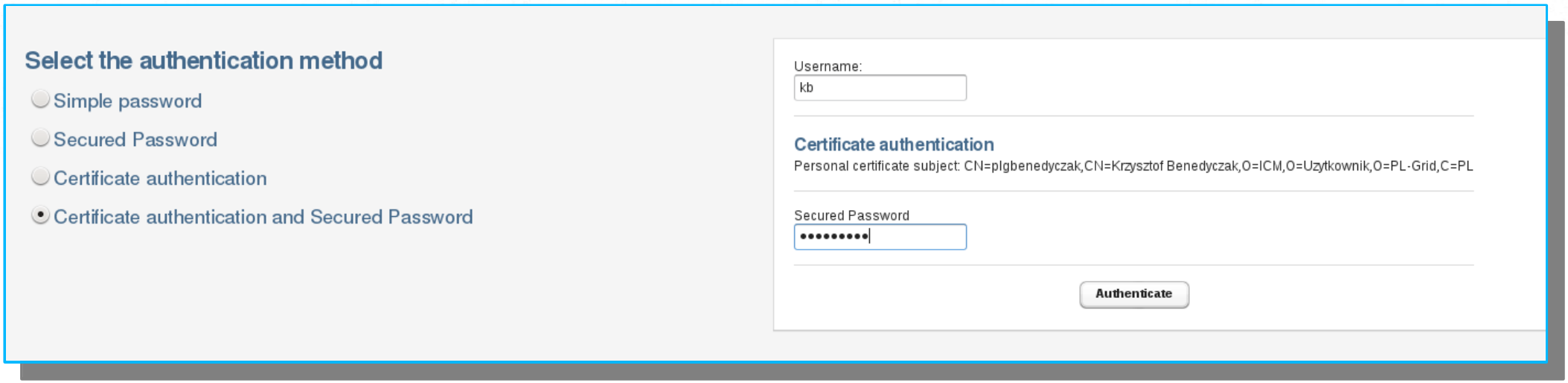


Local authorization roles

- *System Manager*
  - System manager with all privileges.
- *Contents Manager*
  - Allows for performing all management operations related to groups, entities and attributes.
- *Inspector*
  - Allows for reading entities, groups and attributes. No modifications are possible.
- *Regular User*
  - Allows owners to read the basic system information and retrieval of information about themselves. Selected attributes might be marked as editable for such users.
- *Anonymous*
  - Allows for minimal access to the system.

Local authentication

- Uses locally stored credentials.
- Highly configurable:
  - per endpoint,
  - different authentication options,
  - each authentication option may require more then one credential verification.



Local credentials

- Local credentials are fully pluggable.
  - Classic password
  - Certificate login
  - One time password
  - SMS code
  - challenge-response

- Each credential can have advanced configuration.
  - Credential + configuration = credential type.
- Entity has credential requirements assigned, which defines which credential types are allowed for the entity.

External authentication

- Instead of using local authentication UNITY can delegate the authentication to external service.
  - LDAP/ActiveDirectory
  - SAML IdP
  - OpenID provider
  - OAuth
  - others...

- External users must be registered in UNITY (associated with a local identity).
  - Automatic or manual with registration form
  - Support for exposing, re-encoding and importing of external attributes.

Endpoints and protocols

- Access to UNITY is available via different endpoints.
- Endpoint has its implementation (type):
  - Web Administration UI
  - Web User Home UI
  - SAML SSO POST (web) binding
  - SAML SSO SOAP (web service) binding
  - UNICORE extended SAML SSO POST (web) binding
  - UNICORE extended SAML SSO SOAP (web service) binding
  - OpenID provider
  - web service/RESTful administration
  - others

- Each endpoint can be (un)deployed at runtime multiple times with different configuration.

Cross-cutting features

- Built-in support for redundancy
  - redundant instance(s) are read-only
- Auditing
  - configurable
- Notifications
  - email

- Registration support
- Import/export of DB contents.

History

- UVOS - UNICORE VO System - was a predecessor of UNITY.
  - Created in Chemomentum project.
  - Still used in production.
  - SAML support.
  - Mature and very stable, however missing flexibility and extensibility. Some design solutions turned out to be problematic.
- *UNITY is not UVOS 2*, however it is based in UVOS experience.

Status & roadmap

- 1.0.0 released
  - basic local credentials: password, certificate; credential reset
  - Endpoints: SAML IdP (HTTP POST & SOAP), web admin UI, user's home UI, UNICORE SAML IdP.
  - single 3rd party IdP support: LDAP
- 1.1.0 about to be released
  - extensive SAML enhancements:
    - remote SAML IdPs integration
    - publishing of SAML metadata
    - interop tests and related fixes, support for HTTP Redirect in IdP endpoint, many smaller improvements
  - web admin UI can reload and inspect authenticators, endpoints and translation profiles at runtime.
  - many smaller improvements as preferences saving of the identities table settings.
- What's planned for 1.2.0?
  - OpenID integration
  - further enhancements of the admin UI and SAML subsystem

Cloud ready

- Highly useful for cloud applications.
- Frequent updates (every 4-6 weeks).
- Web-based management.

Alternative solutions

- ForgeRock's OpenAM
  - Several products combined together provide similar solution as UNITY.
  - Formerly Sun's OpenSSO, Identity Manager and Sun Directory Server EE.
- PingIdentity
  - PingOne
  - PingFederate
- Onelogin
- And several others, typically named as IAM, SSO IAM or Cloud IAM.

Acknowledgments

- http://commons.wikimedia.org/wiki/User:Victorgrigas
  - servers photo
- James Cridland
  - crowd photo