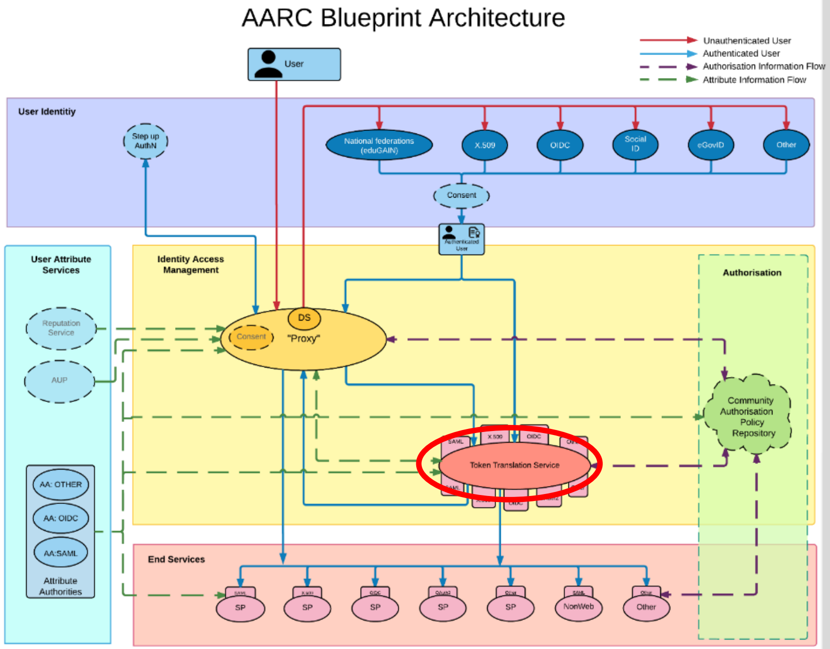
WaTTS

Architecture and Functionality

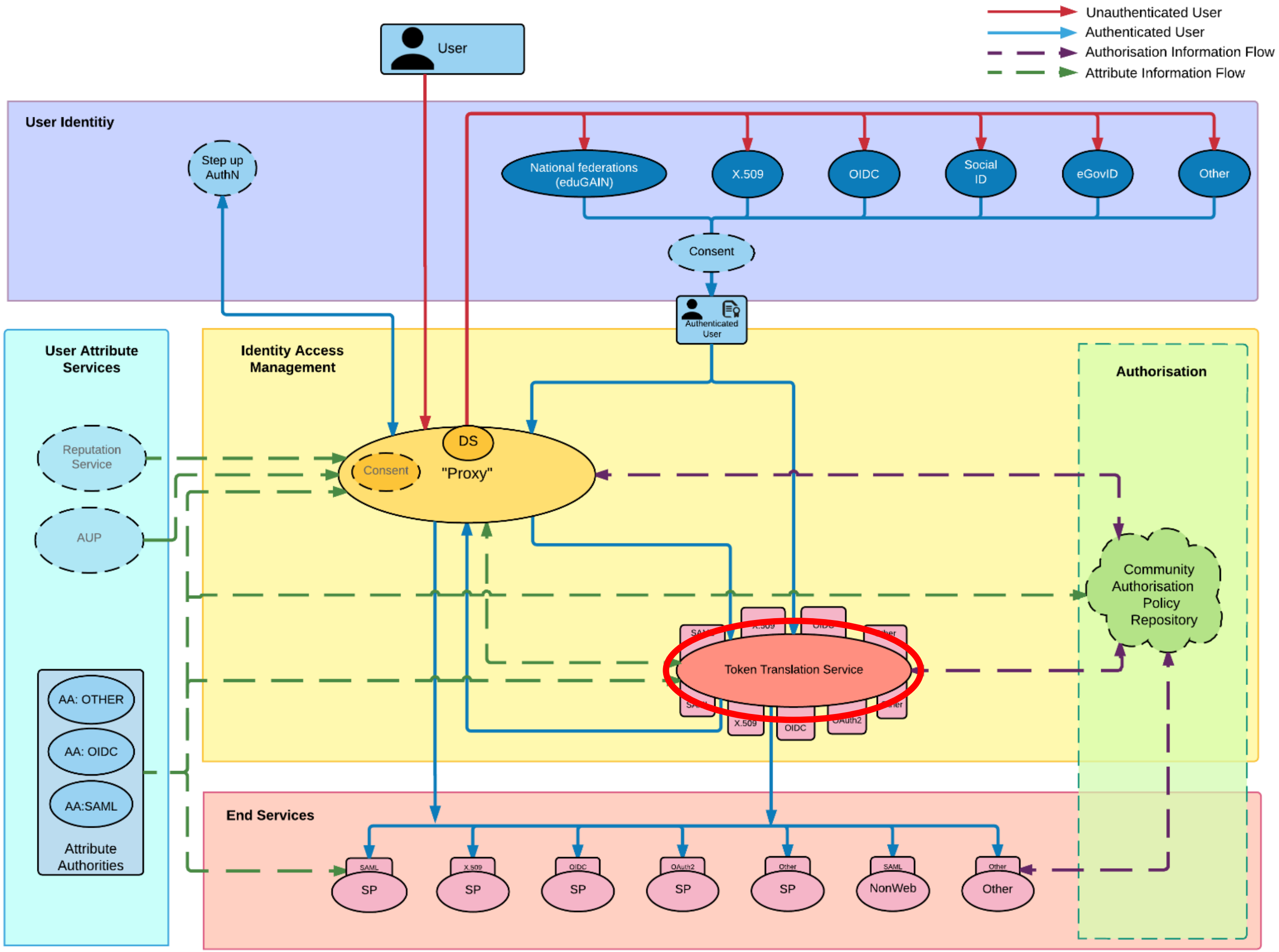
Uros Stevanovic, KIT

Token Translation Services

- Token Translation is used to “bridge” different technological and/or administrative domains
- Access to services that require different “tokens” (credentials) than currently in possession
- Example scenarios:
  - SAML <=> X509
  - SAML <=> OIDC
  - OIDC <=> X509
  - OIDC <=> SSH
- Deployment use cases:
  - Central
  - Service Instance Specific
- Token Translation Type
  - Direct (explicit)
  - Indirect (provisioned implicitly in the service)



AARC Blueprint Architecture



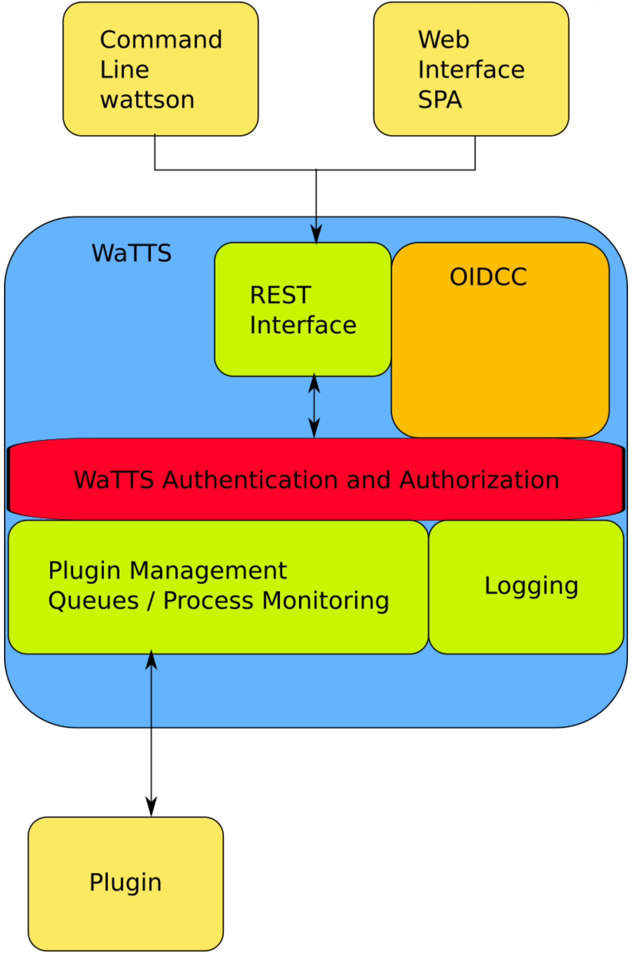
Karlsruhe Institute of Technology

WaTTS – KIT Token Translation Service

- OIDC TTS service
- KIT in-house development, Erlang
- Plugin based, easily extendable
- Enables functionality according to attributes and attributes’ LOA
- Already provides many services (plugins):
  - SSH
  - S3
  - X.509
  - OpenNebula credentials
- Many Python and Go plugins already available
- Open source, Apache 2.0

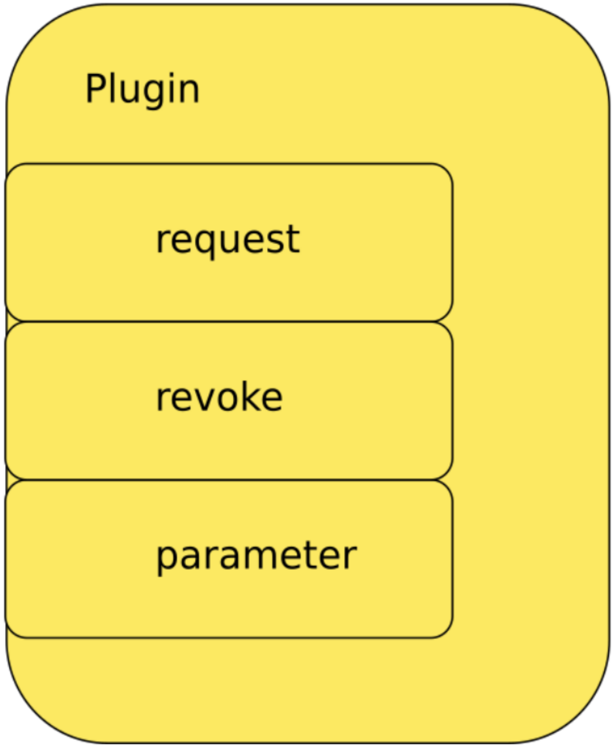
Architecture

- External Clients
  - Command line e.g. wattson or curl
  - Web Interface JS SPA
  - Using REST interface
- Authentication and Authorization layer
  - All data must pass this layer
  - Verification of data
- Plugin Management
  - Handle execution of plugins



Service scheme → Plugin

- Plugin
  - Highly customizable by the administrator
  - Any kind of executable (bash, python, binary)
- Must implement three operations
  - Operation ‘request’
  - Actual translation step
  - OpenID Connect → Service specific scheme
  - Returns a ‘state’, the only information stored at WaTTS
  - (Sometimes ‘abused’ for account-provisioning)
- Operation ‘revoke’
  - Removal or revocation of the credential
  - Gets passed the ‘state’ from the request operation
- Operation ‘parameter’
  - Configuration of the plugin at startup of WaTTS



WaTTS Configuration and Info

- Many OIDC OP are integrated
  - HDF Unity, B2ACCESS, EGI CheckIn, Google, IAMs (DEEP, XDC,..), eduTEAMS
- Provides easy access to plugins to all OP released information
  - iss+sub
  - Name, Email, username
  - Group info
  - SSH keys
  - Info plugin

WaTTS configuration

- watts.conf
  - Textual file
  - Usually in /etc/watts/watts.conf
- Example configuring OIDC OP

```
openid.iam.description = INDIGO Datacloud Identity and Access Management (IAM)
openid.iam.client_id = e70e7190-d64cad3771b9
openid.iam.client_secret = AN5pY379_Z_1jgiiAyKBEHe8zKP2KuiWxV34zWzw
openid.iam.config_endpoint = https://iam.indigo-datacloud.eu/.well-known/openid-configuration
openid.iam.request_scopes = openid, profile
```

- Example service configuration

```
service.info.description = Simple Info Service
service.info.credential_limit = 1
service.info.connection.type = local
service.info.cmd = /home/watts/info.py
service.info.parallel_runner = infinite
service.info.authz.allow.any.sub.any = true
```

Example use case – SSH

Researcher’s point of view:

> *“As a researcher participating in an international research collaboration, I need to have SSH access to a set of VMs, in order to run an interactive software simulation software”*

Technical coordinator’s point of view

> *“Our users should be able to get ssh access to authorized VMs. The should be able to log-in on the project portal and configure their SSH keys and automatically be granted access having a similar experience to popular commercial services like Github.”*

Example use case – SSH

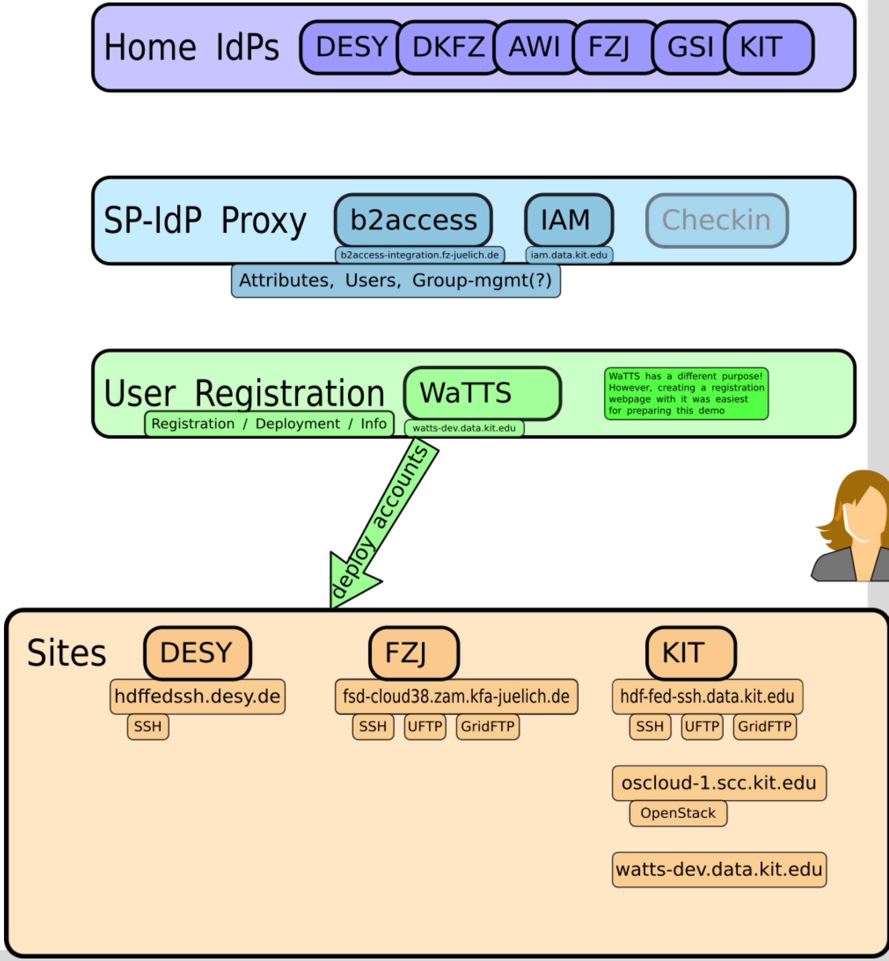
- The community has reserved resources with the HDF partners

- Users authenticate to their home organizations using HDF UNITY

- The WaTTS TTS service has been deployed at KIT, and is used to upload SSH keys

Goals:

- Enable user to SSH into any of his VMs
- Distinguish Authorised users using LoA



Home IdPs
DESY
DKFZ
AWI
FZJ
GSI
KIT

SP-IdP Proxy
b2access
b2access-integration.fz-juelich.de
IAM
iam.data.kit.edu
Checkin
Attributes, Users, Group-mgmt(?)

User Registration
Registration / Deployment / Info
WaTTS
watts-dev.data.kit.edu

WaTTS has a different purpose! However, creating a registration webpage with it was easiest for preparing this demo

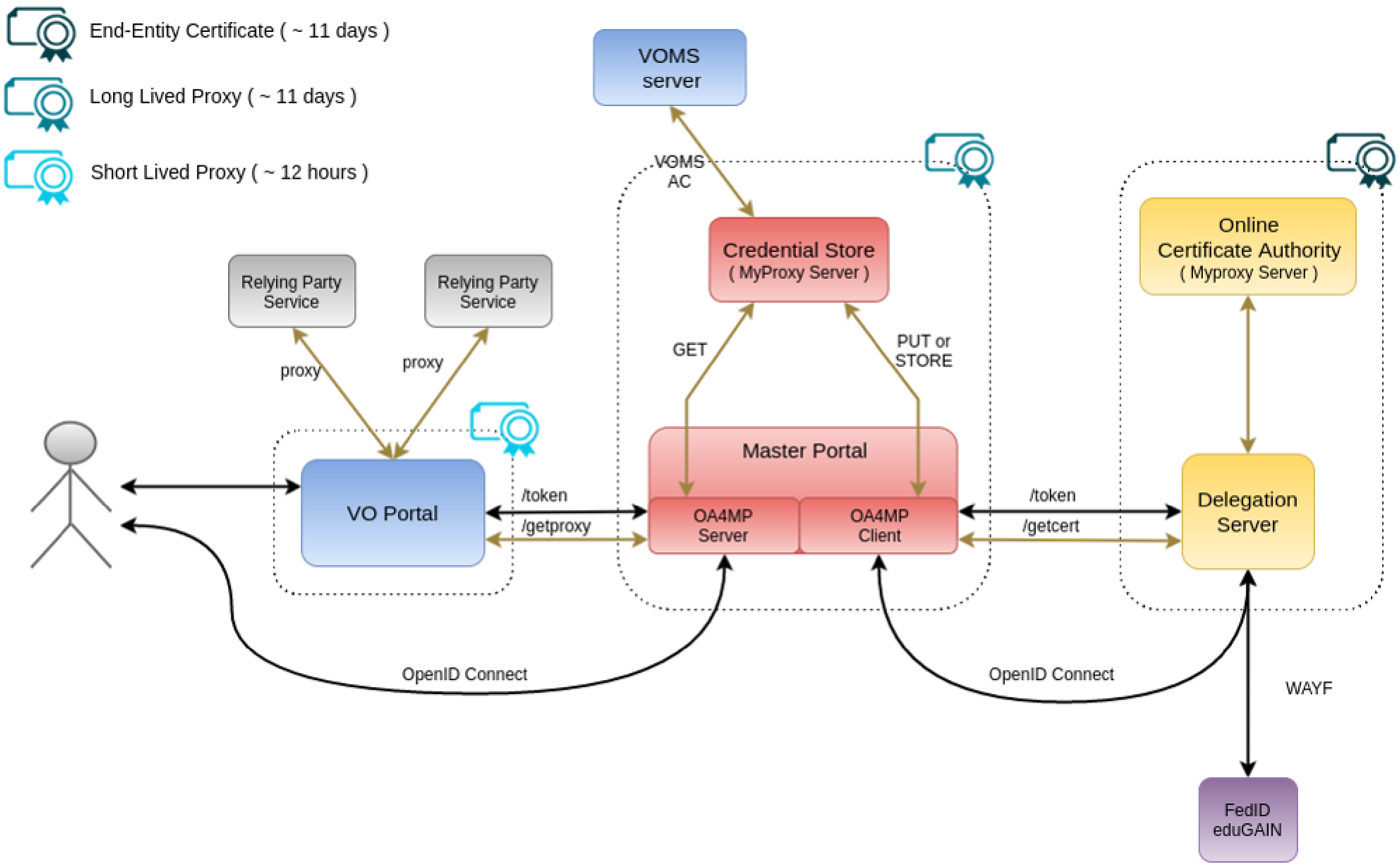
deploy accounts

Sites
DESY
hdffedssh.desy.de
SSH
FZJ
fsd-cloud38.zam.kfa-juelich.de
SSH
UFTP
GridFTP
KIT
hdf-fed-ssh.data.kit.edu
SSH
UFTP
GridFTP
oscloud-1.scc.kit.edu
OpenStack
watts-dev.data.kit.edu

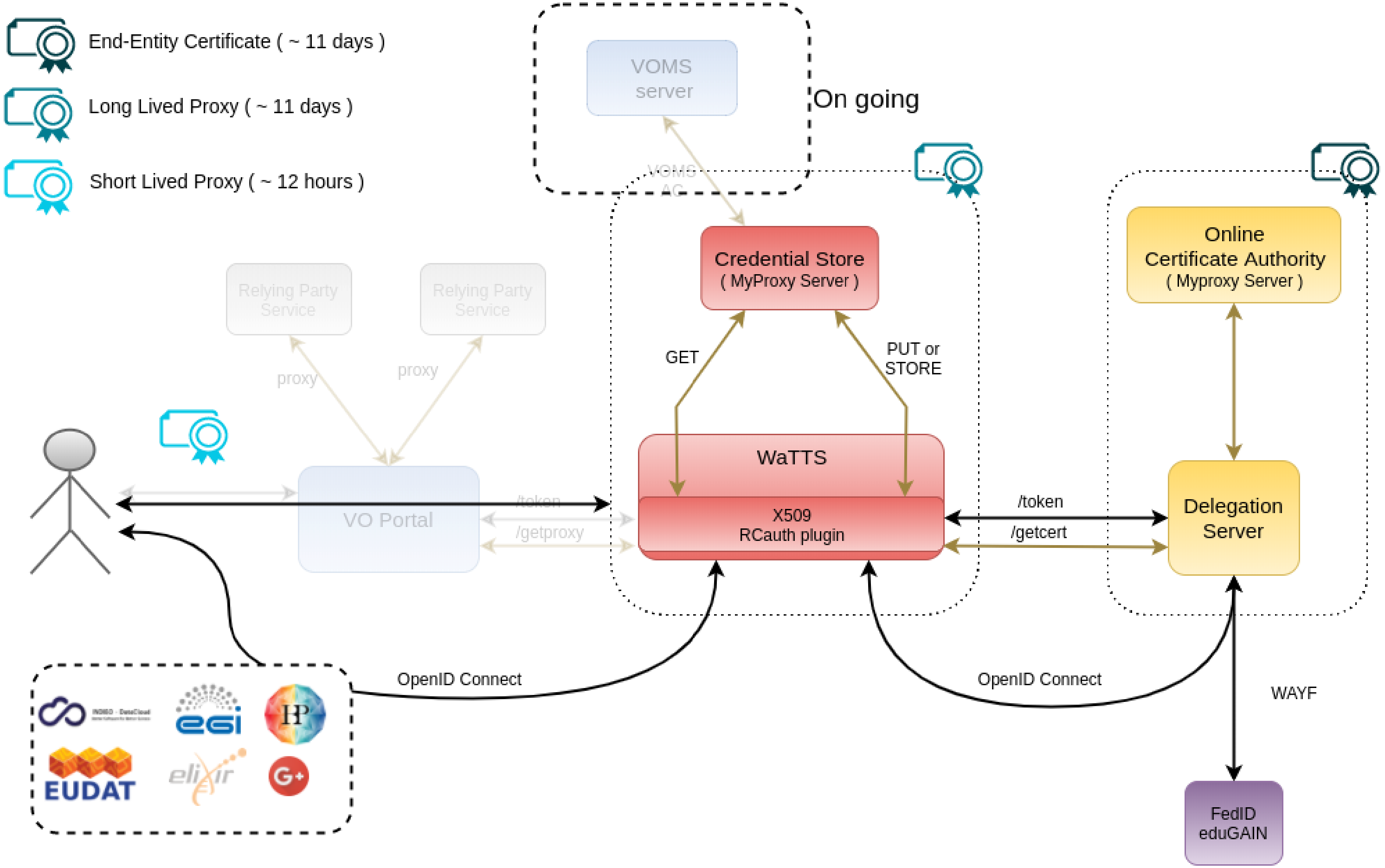
Example use case – x509 IOTA certs

- OIDC flow, with OIDC providers (**HDF UNITY**, EGI, IAM, …)
- “User uses its OIDC token to get a **IOTA** proxy certificate”
- New certificate is obtained from RCauth only when needed (or has expired)
- Simple interface (with web frontend) to receive (VOMS) proxy certificate
- CLI access to get a proxy certificate (with OIDC access token or SSH key)

NIKHEF Master Portal



WaTTS implementation to obtain Rcauth cert



End-Entity Certificate (~ 11 days)

Long Lived Proxy (~ 11 days)

Short Lived Proxy (~ 12 hours)

VOMS server

On going

VOMS AC

Credential Store (MyProxy Server)

Online Certificate Authority (Myproxy Server)

Relying Party Service

Relying Party Service

proxy

proxy

GET

PUT or STORE

WaTTS

X509 RCauth plugin

VO Portal

/token

/getproxy

/token

/getcert

Delegation Server

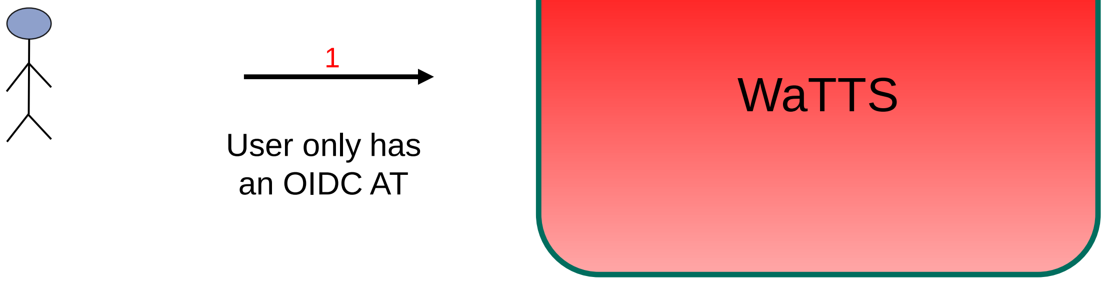
OpenID Connect

OpenID Connect

WAYF

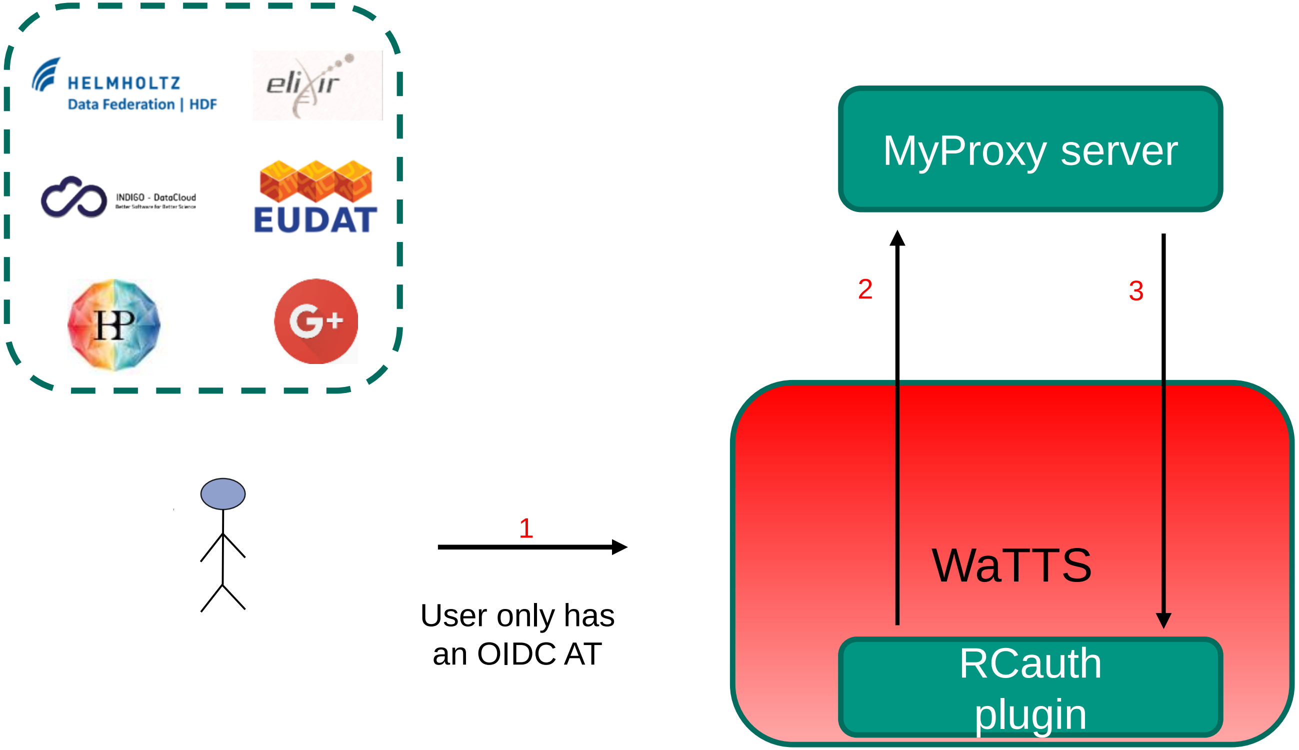
FedID eduGAIN

RCauth flow



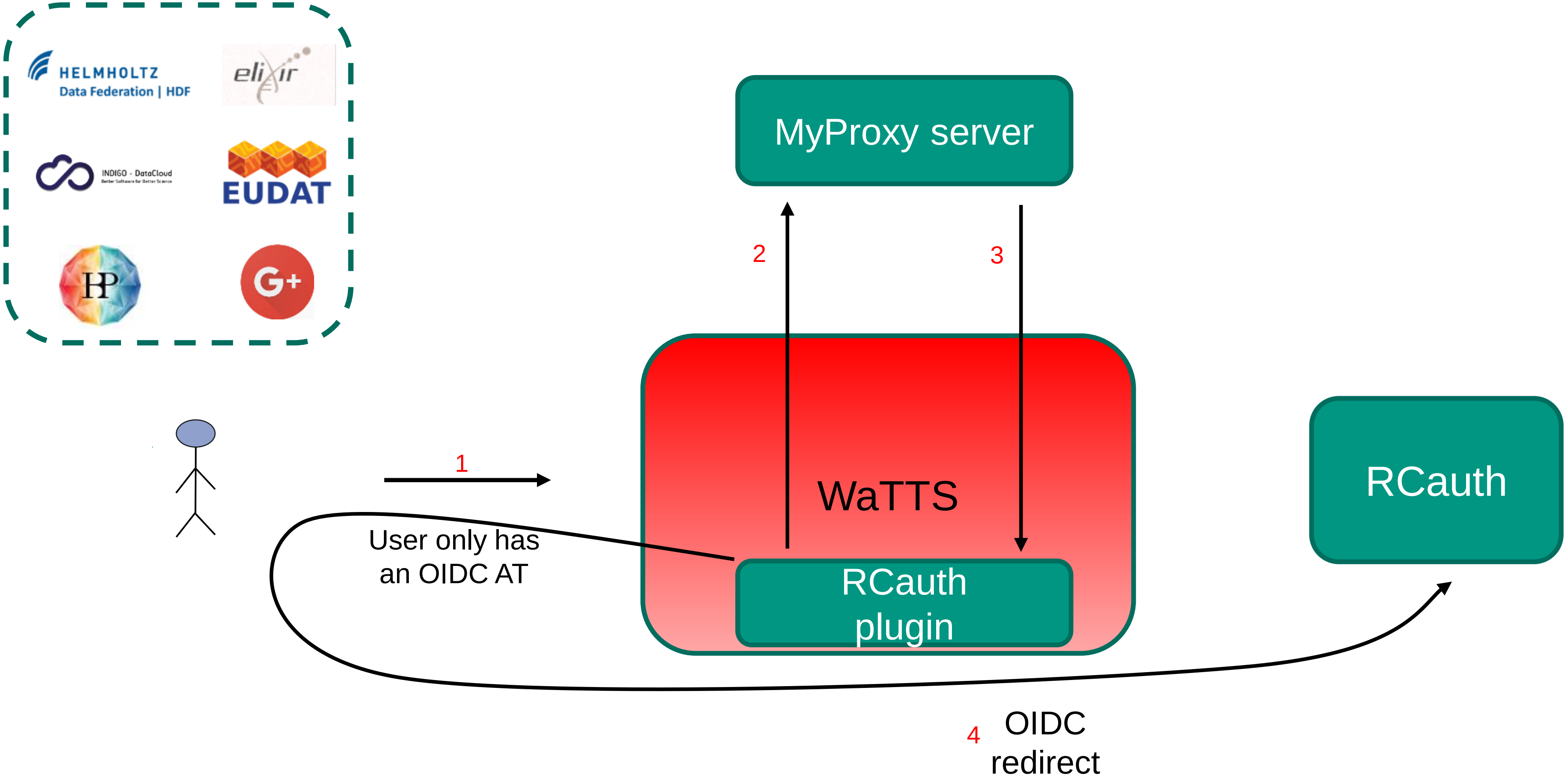
User only has an OIDC AT

RCauth flow



User only has an OIDC AT

RCauth flow



MyProxy server

2

3

1

WaTTS

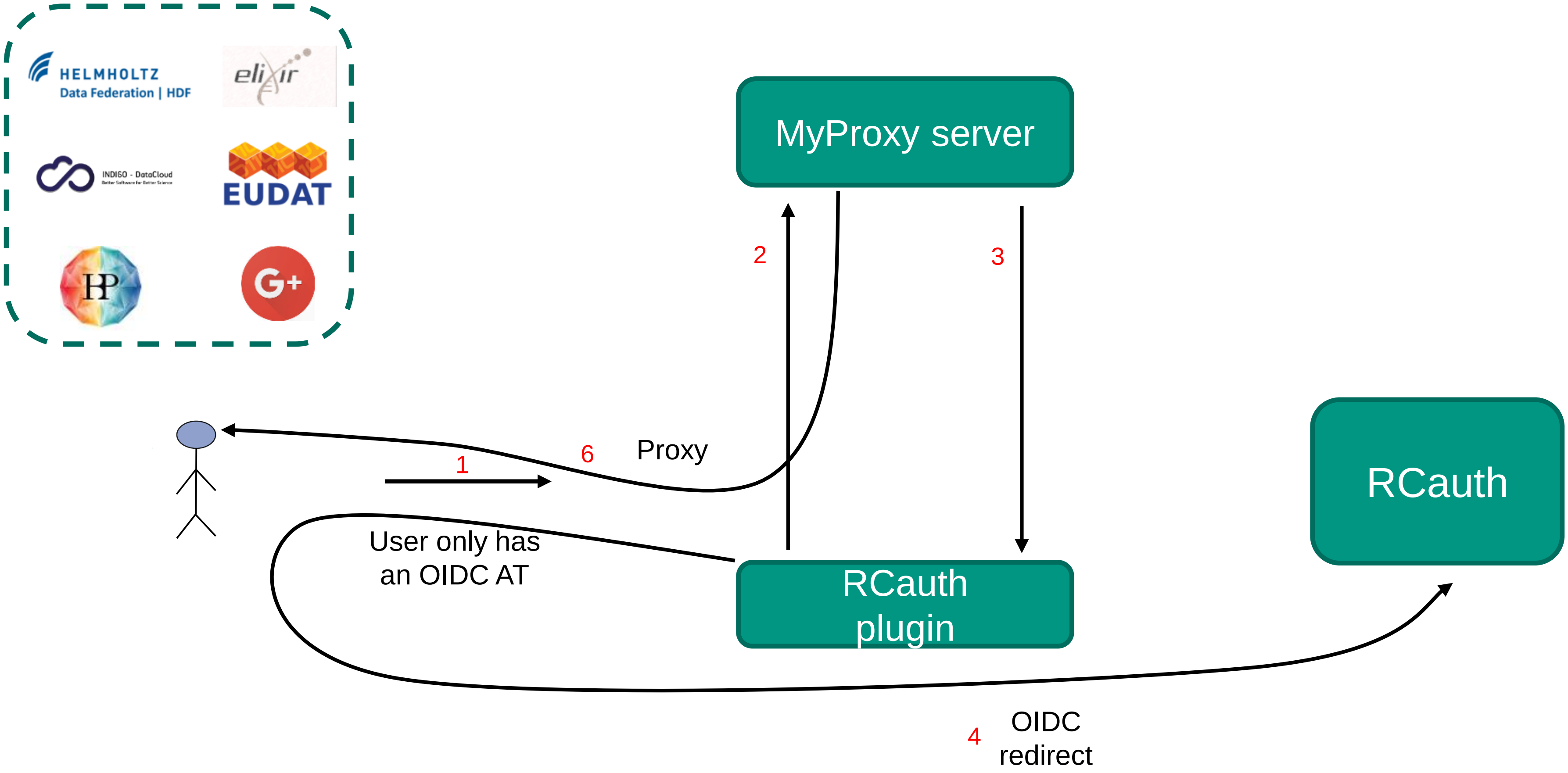
RCauth plugin

RCauth

User only has an OIDC AT

4 OIDC redirect

RCauth flow


HELMHOLTZ
Data Federation | HDF
elixir
INDIGO - DataCloud
EUDAT
HP
G+
MyProxy server
2
3
6
Proxy
1
User only has an OIDC AT
RCauth plugin
RCauth
4
OIDC redirect

WaTTS current deployment

- Life Science Pilot
  - RCauth, AARC Demo CA, EGI Demo CA
- WaTTS and WaTTS-dev
  - KIT
  - RCauth, AARC Demo CA, EGI Demo CA
  - SSH, SSH-CA
  - S3
- WaTTS HDF
  - RCauth
  - SSH

https://watts.helmholtz-data-federation.de

https://github.com/watts-kit/watts