didmos V2 – Technical information

DAASI International

File name: didmosV2-Projektbeschreibung-Englisch-0.14.odt

Created on: 08/17/2018

Last change: 06/07/2021

Version control:

Version	Date	Author	Executed changes
0.01	May 27, 2019	Peter Gietz	First incomplete draft
0.02	June 6, 2019	Jennifer Vosseler	First review minor corrections and comments
0.03	June 14, 2019	Markus Widmer	Answers to comments
0.05	Aug 1, 2019	Markus Widmer	Added chapters on Pwd Synchroniser, ETL Flow and Provisioner
0.06	Aug 6, 2019	David Hübner	Additions to chapter on Authenticator
0.07	Aug 20, 2019	Peter Gietz	Redaction, additions and new 1.1 Aim of this text
0.08	Aug 21, 2019	Peter Gietz	Added 1.2 Used abbreviations
0.09	Oct 09, 2019	David Hübner	Additions for v2.1.0 release of didmos2-demo
0.10	Oct 14, 2019	Romy Hoffart	Correction of grammar, puncutation, spelling
0.11	May 26, 2020	David Hübner	Additions for v2.2.0 release of didmos2-demo
0.12	June 15, 2020	Peter Gietz	Redaction and additions
0.13	July 16, 2020	Jennifer Vosseler	Proofread according to marketing conditions
0.14	June 07, 2021	Jennifer Vosseler	Implementing new logo

Final acceptance:

Version	Date	Accepted by	Role / position
1.00			

Index

1 Introduction

1.1 Aim of this text

didmos V2 is a new open source Identity Management product based on state-of-the-art technologies. It is in some cases an evolution in other cases a total rewrite of version 1. This document describes the basic principles and functionalities of the current version. Since new features are planned and being developed the software is a moving target and thus new versions of this document will be published from time to time.

1.2 Used abbreviations

AD	Microsoft Active Directory
AM	Access Management
API	Application Programming Interface
AT	Access Token
CLI	Command Line Interface
CSS	Cascading Style Sheets
DLL	Dynamic Link Library
DC	Domain Controller
didmos	DAASI Identity Management with Open Source
DSML	Directory Service Markup Language
ETL	Extract, Transform and Load
GUI	Graphical User Interface
HTML	Hyper Text Markup Language
HTTP	Hyper Text Transport Protocol
IAM	Identity & Access Management
IdM	Identity Management
IdP	SAML Identity Provider
JSON	JavaScript Object Notation
LDAP	Lightweight Directory Access Protocol
LDIF	LDAP Date Interchange Format
LUI	LDAP User Interface
MVC	Model, View and Controller
OIDC	OpenID Connect
PDP	Policy Decision Point
RBAC	Role Based Access Control
REST	Representational State Transfer

RPM	RPM Package Manager
SAML	Security Assertion Markup Language
SCIM	System for Cross-domain Identity Management
SLO	Single Logout
SOAP	Netzwerk-Protokoll zum Datenaustausch
SPML	Service Provisioning Markup Language
SCIM	System for Cross-domain Identity Management
SSL	Secure Socket Layer
SSO	Single Sign-On
TLS	Transport Layer Security
XML	eXtensible Markup Language
XSLT	eXtensible Stylesheet Language Transformations

2 didmos General Information

2.1 Basic idea of didmos

didmos (DAASI Identity Management with Open Source) was created over the course of many years of DAASI International doing projects in the field of Identity & Access Management. The basic idea was and still is that the general requirements arising when introducing Identity & Access Management to an organisation are often similar, but that the detailed requirements differ quite substantially in each organisations. Thus didmos is a system consisting of building blocks that can be arranged and enhanced to fulfill all customers needs.

First questions that arise when determining customer requirements are:

- How should data exchange operate among the different systems (data models, interfaces, access information etc.)?
- How can internal permissions be represented in the best way?
- How can specific organisational processes (administration, self-service functions for users etc.) be modeled optimally?

But since these questions always relate to different requirements that are specific to the organisation, we think that proprietary standard software is often an unsatisfactory solution.

In contrast to a fixed standard solution, didmos can be shaped individually because it is made up of modules, which can be individually combined and which each have dedicated plugin interfaces for customer specific logic.

All modules are dovetailed and together they form a flexible, broad and profound Identity & Access Management system that can be adapted to individual requirements and desires. Additionally, didmos provides the ideal conditions to be integrated into your existing IT landscape through high compliance standards and a strong focus on expandability.

In summary, didmos is not a proprietary standard software, but an individual and sustainable all-in-one solution that adapts to all conditions and requirements with the help of its modular structure.

2.2 Basic didmos Architecture

In contrast to a rigid standard solution, didmos is flexible, as it consists of five adjustable modules. They implement all functionalities needed for a reliable, flexible and feature-rich Identity & Access Management system.

Two big design decisions were made for didmos:

1. We use an LDAP server as persistance layer, called metadirectory, instead of using a relational database. For one we think that the object oriented data model is more suited to store identity information. Secondly, it allows for a much greater flexibility in data schema, since any object can be enhanced by applying new auxiliary object classes and attributes to the main objects (structural object classes), like person, organisation, etc. but also by introducing completely new types of objects, like request for group membership and the like.
2. We use unidirectional connectors to reduce complexity following the “keep it simple” paradigm. This does reduce complexity without giving up the possibility to have one system be source and target at the same time. This is done by using one synchronisation and one provisioning connector for the same system.

Thus the following diagram (1) shows three basic layers (source databases, metadirectory and target applications) and our software modules in between or within these layers:

- **didmos ETL Flow** for synchronising data from source data bases into the metadirectory. This is a number of single functionalities or sub-modules that can flexibly be arranged in workflows. Sub modules exist, e.g. for:
  - connectors, reading from the different data bases and transforming the date into a common format, namely DSML (Directory Service Markup Language), an XML dialect for describing directory data, which is compatible to LDIF (LDAP Data Interchange Format)
  - identifier, that find the same identities occurring multiple times either in one or in multiple data sources; this module is highly configurable considering what to do when two entries are interpreted as the same person (e.g. which attributes to use with which weighting, thresholds for identity, non identity and doubtful cases, when to write an automatic email to an administrator for further research and decision)
  - attribute merger, that decides which attributes to take from which sources also with complex configuration options such as „use attribute x if in source 1, else attribute y in source 2“, or „collect attributes from all sources“
  - grouper: for automatically creating group and role memberships based on certain source attributes
  - transformer, a module that executes XSLT transformations
  - diffcreator comparing the final result of the workflow with the data in the metadirectory to create a diff file to input into the directory
  - ldaploader transforming the computed diff into LDAP operations
- didmos Core a REST web services based interface for reading and writing to a central metadirectory, which is the persistancy level of didmos. Besides the metadirectory it

consists of a number of so called Apps, which each have a REST based interface (see also below 2.3 and figure 2).

- The metadirectory is implemented with OpenLDAP with a configured Accesslog overlay, so that all changes to the metadirectory are logged in dedicated LDAP entries (one entry per change). With this information, any audit relevant reports can be created.
- Core App: this main REST interface for reading and writing identity and group related data is based on the SCIM standard, so that SCIM supporting frontends should work with Core. In addition to the schema and protocol specified in the RFCs, additional functionality had to be implemented in this interface to allow for all identity management relevant operations.
- Also included as App in didmos Core is the didmos Decision Point, an RBAC (Role Based Access Control) compliant central policy decision point, with which access to any kind of resource can be managed: applications, single menu items, files, etc. as well as any component within the didmos world, including the single identities or subtrees of identities, etc.
- Another App is in charge of workflow and task management. Basically this is an infrastructure to create applications (e.g. for account registration, role-inhabitance, group membership, etc.) and for granting or denying such applications, where a dedicated administrator can retrieve all applications for which the role inhabitant is in charge. This can be modeled to single group instances, which can have a separate role which allows it's management.
- The central configuration for all didmos modules is also a separate app, which stores the configuration in LDAP and provides it via dedicated REST web services endpoints.
- In customer projects it is possible to add customised Apps for special customer requirements, with their own REST API.

- **didmos LUI** for creating web based administration and self-service interfaces. This is a highly flexible and configurable set of functions which can easily be extended to fit any custom requirements. It allows for creating interfaces that have the same look and feel as other applications of the customer respecting its corporate identity. Basically LUI is an Angular based framework, that communicates with didmos Core via the SCIM REST API.
- **didmos Provisioner** for provisioning the data to any target applications. It reads every change in the metadirectory and provides respective change requests for target systems. The basic idea being that only the last bit, i.e. the changes described in the methods of the target specific protocol or interface, needs to be amended when adding a new target system. According to the version used (didmos V1 or didmos V2), both respective standard protocols, SPML (Service Provisioning Markup Language) and SCIM (System for Cross-domain Identity Management), are supported, either SPML with a SCIM endpoint (=didmos V1), or SCIM with an SPML endpoint(= didmos V2).
- **didmos Pwd Synchroniser**, a Windows DLL and Windows service that allows for reading the cleartext password of password changes made on a Windows Active Directory (AD) Domain Controler (DC) and then creating either a hash, several of which are supported, a reversibly encrypted password via X.509 based asymetrical encryption or sending the password to a REST service. A queuing service ensures that changes are temporarily stored encrypted on disk if the target is not available

- **didmos Authenticator** a SAML2 and OIDC compliant single sign-on solution that allows for authentication and access management. It can also act as a proxy solution for translation between different SSO protocols (e.g. authenticate with a SAML IDP at a OIDC based service) and connect to various external IDPs (e.g. Social IDPs).

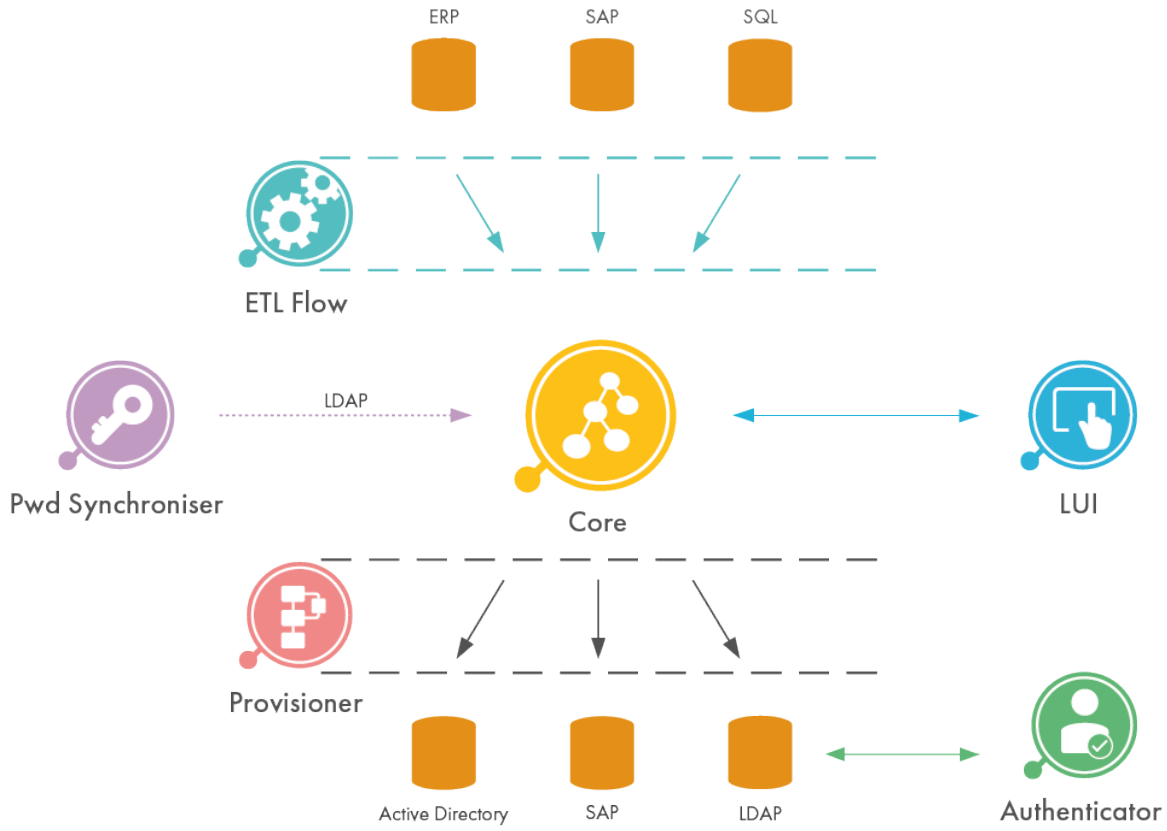

Figure 1: General Architecture of didmos

2.3 Architecture of didmos V2 Core

In general, the same architecture is used in didmos V1 and didmos V2. Thus modules of didmos V1 and didmos V2 can be used together.

The new Core module, didmos LUI, didmos Decision Point and didmos Provisioner already exist as a first iteration of version 2. Since the respective interface is our metadirectory with unchanged LDAP v3 protocol, didmos V1 ETL Flow and didmos Pwd Synchroniser can be used together with didmos V2 components. There is no need to redesign the Pwd Synchroniser component, it is thus integrated in didmos V2 rather unchanged. In the following diagram (2) the architecture of didmos Core and the interaction with other modules is shown.

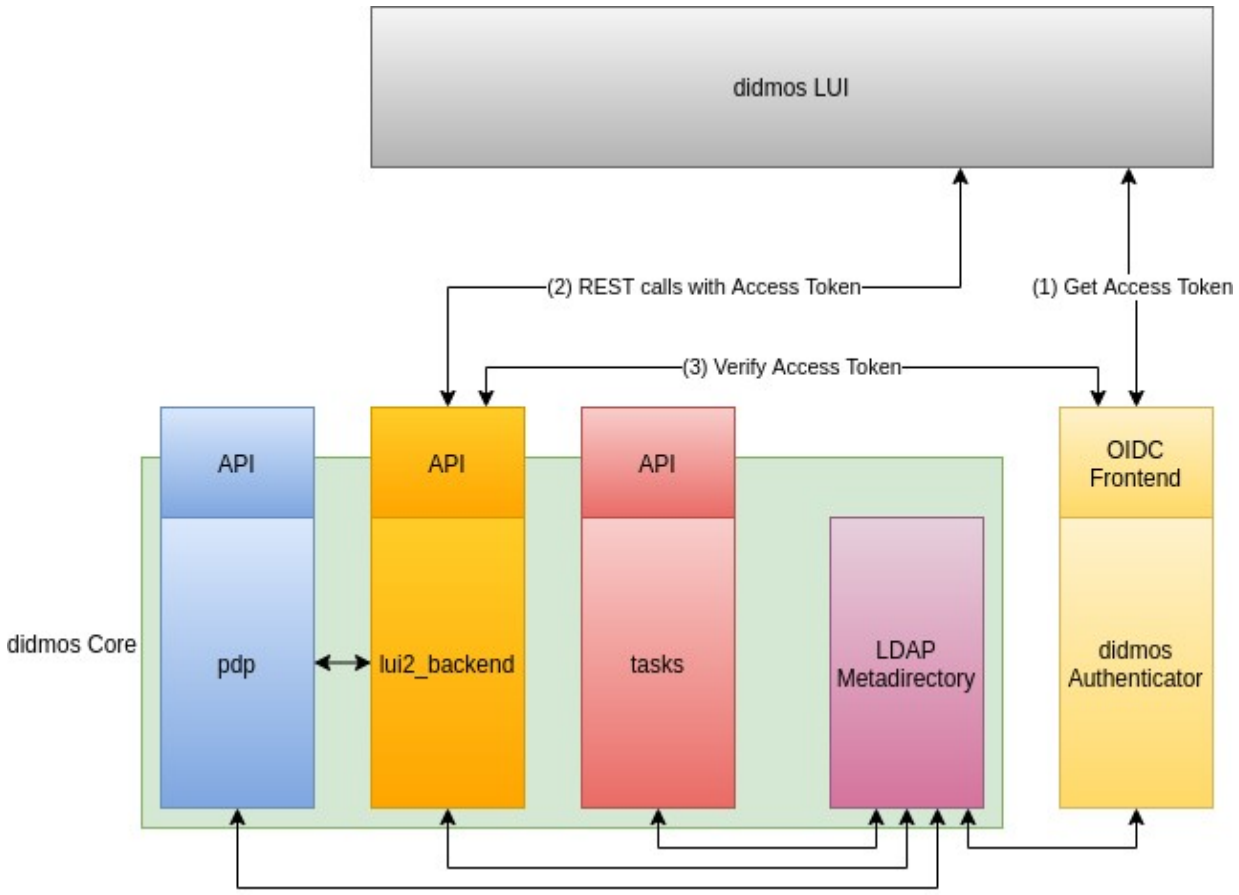

Figure 2: Architecture of didmos 2 Core

didmos V2 is compliant with the current software paradigm microservices architecture. It is strictly divided into frontend (didmos LUI) and backend (didmos Core) which are communicating through a well-defined REST interface.

The frontend is purely JavaScript-based, whereas the library and programming paradigm used is Angular.js. The layout of the frontend can be adapted as desired through HTML templates and CSS, so any corporate identity can be reflected. The positioning of site components (menus, display elements, etc.) can also be administered. Besides the basic functionalities customer-specific functionalities can be implemented, whereby the respective source code is kept strictly separate. During the execution of basic identity management functions (create, change or delete data objects; add members to a group or role, etc.), the frontend communicates with the backend using the standardised protocol SCIM, which was extended by some necessary functionalities (therefore “SCIM+”). The customer-specific functionalities are all able to communicate to the backend using customer-specific REST protocols.

Just like the other didmos components, the didmos Core backend has been programmed using Python, whereby the widespread open source framework Django, which follows the MVC principle, is used. Django is a web framework based on Python which enables the programmers to develop applications clean and fast. The framework follows the idea that you can focus on coding the so-called applications, while a variety of standard coding tasks are performed by the framework. This allows for an efficient and goal-oriented coding process. In the backend, the microservices

principle is used consequently as well. The features which are encapsulated in the applications can communicate with each other as well as access common libraries. The also encapsulated layer of the LDAP interface on the other hand takes care of the reading from and the saving into the LDAP server.

As can be seen, in 2, the new didmos Decision Point (called pdp in the diagram) is now integrated into the core module. This enables every component in didmos Core module to ask the Policy Decision Point (PDP) access questions, without using its REST API. Nevertheless any application can use this PDP for their own purposes, provided they implement the standards based interface. Another new module is the task module, which allows for any kind of work flow, e.g. an account or group membership request that can be made via a self-service instance of LUI and its granting via two occupants of a respective role via an administration instance of LUI.

The new didmos Authenticator module is heavily used by various other didmos modules, especially for all HTTP requests between frontend applications (didmos LUI) and the REST endpoints in didmos Core.

2.4 Differences between didmos V1 and didmos V2

didmos V1 has been developed successively, implementing features needed in the different projects DAASI International has been engaged with. The single modules have been developed by different developers and also in different programming languages (LUI in Perl, the PDP in PHP and the rest, ETL and Provisioner, in Python). In didmos V2, all components are based on Python, with the exception of JavaScript-based frontends and the Pwd Synchroniser, which is implemented in Windows-friendly C++ and C#.

Whereas in didmos V1 every module has its own configuration, in didmos V2 we are implementing a single configuration for all modules stored in the LDAP server. This “config server” will be a dedicated app in didmos Core and we’re constantly moving more a more configuration to that central place.

Whereas in didmos V1 we used the technologies popular at the time it was created, basically XML and SOAP, in didmos V2 we rather use JSON and HTTP/REST. Where ever we still see the XML technology as a better fitting option, we stick to XML for now (ETL Flow).

Whereas in didmos V1 we used Shibboleth as identity provider supporting SAML2, in didmos V2 we use our own SATOSA-based Authenticator that supports SAML2 and OIDC.

In didmos V1 we create software packages (RPM, Debian packages), whereas in didmos V2 we follow the policy “Docker first”. But for didmos V2, classical packages are also available. For now we have deployment instructions for Ubuntu 18. WE can also provide other packages for other llinux based operating systems.

3 didmos V2 Modules & Technology Details

3.1 Persistance layer OpenLDAP

As already described, we use an instance of an OpenLDAP server as the metadirectory.

3.1.1 Advantages of OpenLDAP

Especially in the last years of its development, scalability and performance were the main objectives for developers working on OpenLDAP. According to the main developer, current versions can “handle billions of objects and data bank volumes in the areas of terabytes by now. More than 100 000 queries per second with latency periods below one millisecond are possible. Even in such high performance situations, the software runs robustly. If there is any downtime at all, it is usually due to hardware failure.”

OpenLDAP has, among others, the following advantages:

- a highly performing system, which allows more than 500 000 read accesses per second even for vast amounts of data (> 1 billion entries), given the appropriate hardware and search filters with indexed attributes
- the option of a highly failsafe multimaster-cluster
- one server can use multiple database backends, which can either be located on local hard drives or on SAN file systems (distributed data storage in the network)
- fine-grained access controls, up to the access management of a certain attribute value
- entirely LDAPv3 compatible – in fact the reference implementation of the standard – and can therefore be addressed by any system supporting LDAPv3
- secure options for data transfer through encryption (SSL or TLS)
- configuration through LDAP data (subtree cn=config), so changes in configuration won’t require a server restart
- the software package includes not only the server, but also other tools required for the configuration and necessary libraries; it is mainly made up of the following components:
    - sladp – stand-alone LDAP daemon, the actual server
    - backends – through these, the actual access to the data is realised; there are multiple standard backends available as well as specialised backends, for example for monitoring
    - overlays – allow you to modify the behavior of the backend and thereby the behavior of the sldap, without changing the backend or the sldap themselves, examples include:
        - “refint” to secure the referential integrity, so to avoid indicators on not (or no longer) existing entries and to update group memberships
        - “unique” to make sure that within certain attributes, all over the database there are only definite/unambiguous values saved
        - “syncrepl” for the synchronisation and replication on various servers
        - „accesslog“ for storing every change made in the directory data. This is used by LUI for displaying all changes made to an entry and all changes in group and role memberships of this entry and it is used by Provisioner for provisioning all changes to the target systems.

        - „memberOf“ that automatically creates and removes attributes when attributes of other entries that refer to their DN are added and removed. Basic use case is to have the attribute memberOf with a particular group name added to every entry that is pointed to in a member attribute of a group object and deleted again if membership is removed.
    - libraries which provide the LDAP protocols
    - command line tools, like for example adding, modifying and deleting data, to create data backups and to recover those, etc.
    - tools, auxiliary means and examples
    - good UTF8 support

3.1.2 LDAP data model

The data model is not limited in regard to a specific application and can therefore be adapted precisely to the requirements of the structure described in the performance specification. The LDAP data model is made up of entries which can be arranged in a hierarchical tree. An entry is modeled as an object by determining the type of the entry through the object class. The object class specifies which attributes (equivalent to data fields like for example name, address, creation date etc.) can be saved in the entry. Compared to relational databases, this allows for greater flexibility since an additional object class can simply be added to an entry in order to save another attribute. Besides the access protocol, a number of object classes and attributes were internationally standardised as well. Where it is reasonable and they are available, standard LDAP attributes and object classes are used in the project. For all other required objects, new project-specific object classes and attributes were defined and used. With this very flexible data model, we can say that didmos can be adapted to any kind of data, which alows tio implement any customer requirements.

3.2 didmos V2 Core

The didmos V2 core component provides various functionality for managing objects such as users and groups including creation, deletion and modification of such objects. The module is entirely written in Python based on the widely used Django framework. It supplies a SCIM v2 endpoint for object management as well as endpoints for the decision point and the request engine, all accessible over REST.

The main principle for developing the core was flexibility and extensibility. Therefore the software is highly configurable to suit all needs. If there exists a use case in which configuration is not enough, it is possible to modify functionality of the core in a fast and clear way. This enables the developer to adjust the functionality of the software according to his needs as well as to rely on widely tested components to keep testing and decrease the risk of bugs to a minimum.

Besides the three apps described in 2.3 “Architecture of didmos V2 Core”, additional custom apps can be integrated to allow for any functionality required by the customer.

3.3 didmos V2 LUI

didmos LUI (**L**DAP **U**ser **I**nterface) is a web-based frontend, which enables a number of self-service and administrative functions. This generic web portal framework can create, modify and delete data in a LDAP server. A user only has access to those functions, for which his or her LDAP role object is authorised. The entire frontend can be configured individually and adapted to a corporate design.

In addition, a number of basic functions such as the following are available:

- self-registration with an email verification
- management of personal data
- automated creation and sending of emails
- search and browse function
- manipulation of group and role memberships
- password (re)setting function
  - email with a once valid link or token
  - secret attributes
- GDPR-compliant information on all saved data in the system about one specific individual
- make requests (e.g. for membership is groups)
- modify and permit or decline requests
- mass imports of data via CSV files
- web service interfaces to any other web frontends
- paged search results to handle large data sets

This means didmos is suited for the implementation of administration and self-service tools. Thanks to its flexible configuration options, LUI can be used for any other desired web frontend, for example within digital humanities applications with complex opportunities for visualisation.

Within several projects the following features are already in use productively:

- General
  - multiple languages
  - multi tenants
  - password policies (also different policies per tenant)
  - changable themes
- Pages for not logged-in users
  - login page
  - “forgot password” workflow (via email or SMS)

    - self registration
    - account requests
    - general web portal functionality (e.g. welcome page, help page, application overview)
- Self-service portal
    - my data (show me all data stored about me)
    - change password
    - delete my account / all my data
    - manage MFA tokens (using the software privacyIDEA)
    - make request deletion of all my personal data
    - activity log: show all changes to my entry
    - request to become member of an ID-Federation (for registering new service providers in a federation), supporting SAML 2.0 and OIDC/OAuth 2.0
    - request for admin privileges
- Administration portal
    - search user
    - list users
    - add new user entry
    - modify user entry
    - CSV import of users
    - list groups
    - add new group
    - modify group and group memberships
    - restrict group membership by role of user
    - browse LDAP tree
    - grant / deny data deletion requests made in the self service (Tasks)
    - create organisations
    - modify organisations
    - grant / deny account requests
- other smaller features
    - activate / deactivate accounts
    - activate / deactivate single pages and respective menu items via runtime configuration and Docker environment variables
    - configurable external links in menu

- Layout features
  - footer always at the bottom
  - background image on main page
  - sign-in and sign-up link in header when not logged in

Continuously more features are being developed, in addition to the list mentioned at the beginning of this sub chapter.

3.4 didmos V2 Authenticator

didmos Authenticator is the central authentication component in the didmos V2 software suite. It is based on the SATOSA proxy and supports the SAML and OpenID Connect protocols. With the component, it is possible to offer arbitrary authentication methods, including login with local user accounts and login using social IdPs, such as Facebook and Google.

Other components, for example didmos LUI, delegate login to the didmos V2 Authenticator. But usage is not limited to didmos applications. All services, which offer SAML or OpenID Connect support, can be connected to the didmos V2 Authenticator and included in the Single Sign-On (SSO) landscape.

On a technical level, SATOSA is composed of different modules. Backend modules represent authentication methods and connect to different authentication sources (e.g. local accounts or Facebook). The result of a backend module consists of an user identifier and possibly additional user attributes from the authentication source. Conversely, frontend modules are used to connect to various services (e.g. didmos LUI or external applications). They convey the information (which is based on whatever the backend module produced and potential modifications in micro services) back to the relying parties. Finally, micro services perform all kinds of tasks (like routing or attribute modifications) between frontend and backend modules. They can be further divided into request micro services (these run when routing from the frontend to the backend takes place, before any response from the backend is produced) and response micro services (these run on the way back from the backend to the frontend, after the backend has produced its result).

With this modular approach it is possible to adapt SATOSA to the specific needs of the didmos software suite and it is also a very flexible system to extend in order to fulfill new use cases.

It also supports multi-factor authentication by interfacing to privacyIDEA, an open source application for management of and login with various 2FA tokens.

Finally, a custom command line interface can be used to control OIDC clients registered in Authenticator (e.g. list all registered clients or register a new client). This CLI internally talks to the MongoDB storage layer.

The following SATOSA modules are currently used in didmos Authenticator:

Backend Modules	local	Login with local accounts in didmos V2
	facebook	Social Login with Facebook
	google	Social Login with Google
	linkedin	Social Login with LinkedIn

		github	Social Login with GitHub
		Saml2	Login with SAML2 IDP or federation
		Saml2UCS	Login with UCS SAML IDP
		externalldap	Login with external LDAP or AD
		LdapAttributeStore	Query additional user attributes at internal LDAP
Response Services	**Micro**	Privacyidea	MFA with privacyIDEA for internal users
		ldap (ldap_account_registration)	Shadowaccount registration for all but internal users
Request Services	**Micro**	DiscoveryRouter	Frontend service to choose login method
Frontend Modules		OIDC	OIDC Provider
		SAML2	SAML IDP

Most basic features can be configured via docker environment variables, but of course it is also possible to write more extensive custom configuration in SAML files, which are understood by SATOSA. The system is also capable to display all interfaces in mutliple languages.

3.5 didmos V2 Provisioner

The didmos Provisioner is used to update target systems instantly when changes are recognised in a source system. Currently this works for OpenLDAP in combination with the access log overlay as source. For target systems an ICF connector is used. Sending a change to a target system follows these steps (Figure 4):

- Recognise a change in the source system
- Transform the change into a JWT containing a SCIMv2 document as payload
- Write the JWT to a message queue (RabbitMQ)
- A worker reads the document from the message queue and applies it to the target
- The result is written back to the message queue
- A worker reads the response and decides what to do (e.g. write back an ID or status)

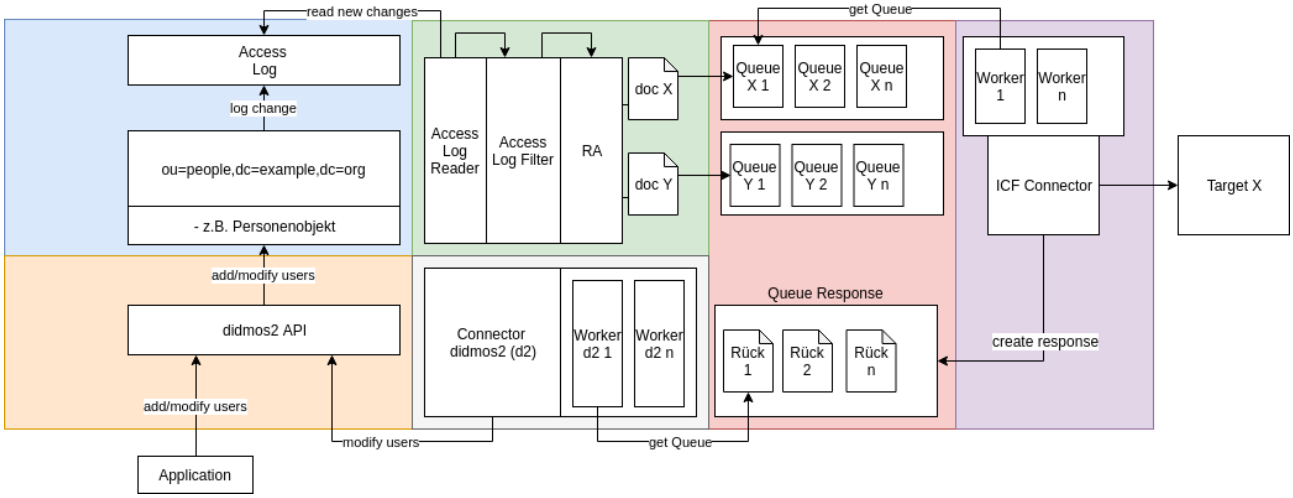

Figure 3: System layout of provisioner.

The Provisioner is meant to send changes in the source instantly to one or more target systems. By using a message queue (RabbitMQ) and multiple workers, the changes can be applied in parallel and it is assured that no changes can get lost due to network outages etc. The system is easily scalable to achieve fast provisioning.

3.6 didmos ETL Flow

The didmos ETL Flow (*Extract, Transform, Load, Workflow*) is a collection of various small modules that can be combined to collect data from various sources (databases, LDAP directories, files), combine them to normal condition and compare it to the actual state of the target system.

A typical workflow could look like this (Figure 4):

- read data from various sources (e.g. one SQL database and one CSV file)
- convert all retrieved data to DSMLv1 format
- prepare the data using XSLT
- link each entry in the sources to an entry in the target system by searching them in the target system
- merge duplicate entries within the same source (not pictured in Figure 4)
- merge duplicate entries from different sources
- read the actual state of the target system
- calculate the differences between the two data sets
- apply the changes to the target system

Two of the most important features are the capability to

- link entries from the sources to entries in the target system using complex mechanisms such as weighted comparison of different attributes or
- merge attributes from different sources based on priorities for each source/attribute combination.

These features allow to design complex rules about how to find and merge entries.

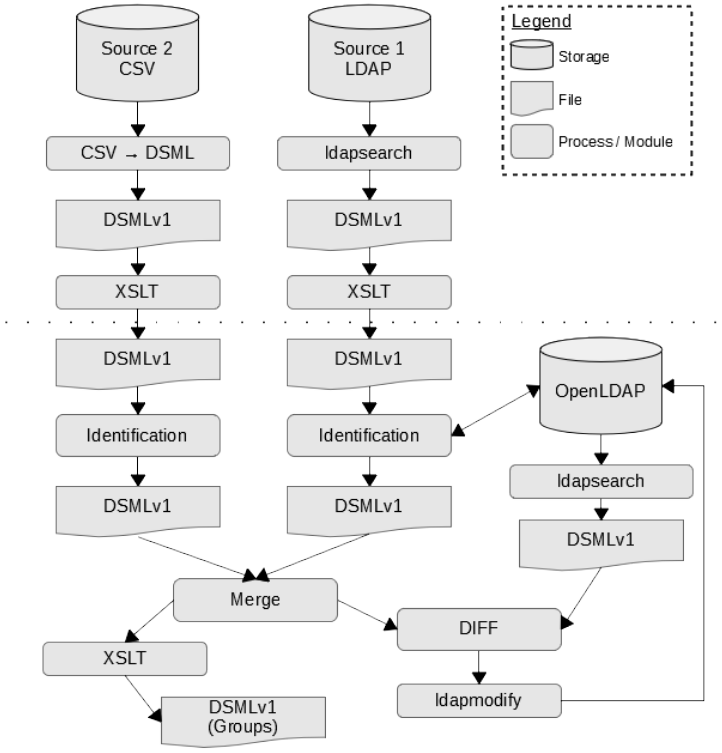

Figure 4: Example for ETL with two sources.

The ETL is not a tool to have instant changes written to a target system but to initially synchronise or periodically re-synchronise sources with a target.

3.7 didmos V2 Decision Point

The didmos Decision point is a Python implementation of role based access control (RBAC) with extensible functionality to fulfill special requirements of the didmos Core. RBAC implies that permissions to execute actions are not assigned directly to a user, but to a role. In a second step, the role is assigned to the user. Not only is it possible to set permissions for an object but also to restrict possible actions executable on such an object. It is for example possible to set the permission to modify an object but with modification not being allowed on the object, the modification request will still be denied.

Often a lot of repository objects have the same permissions and executable actions. To simplify this, it is possible to set permission templates for an object, where only the template has to be assigned to the object and permissions and restrictions are enforced over this template.

Another functionality implemented in the didmos Decision Point is role inheritance. This enables the administrator to assign a role to a user and implicitly assign the user to all parent roles.

3.8 didmos Pwd Synchroniser

The Pwd Synchroniser was developed to enable the provisioning of passwords from Active Directory into other directories. Passwords are stored in a proprietary hash format, so they cannot be easily read and transmitted from a Domain Controller into another directory. Often this blocks the implementation of other directories for a parallel operation or during a migration phase. The solution, developed by DAASI International, makes it possible to apply the passwords changed at a Windows PC into external directories, e.g. OpenLDAP. The users are assigned by a loose interconnection to a distinct ID, e.g. the e-mail addresses of the users. That's how the software can also easily be integrated into an already existing infrastructure. The synchronised passwords are then available for an authentication of the users at an external directory service.

Feature overview:

- easy installation as a Windows service
- encrypted buffer storage on the Domain Controller when the target directory is not accessible
- definition of the users who have to be synchronised by Container or LDAP filter
- transfer by LDAP or LDAPS, HTTP or HTTPS
- setting of the password as X509-encrypted clear text or in the form of following hashes:, SHA, SSHA, SHA512, MD5 or PBKDF2 hash
- alternatively to the setting of the password an external script can be executed
- logging of the synchronisation processes

4 Appendix

4.1 didmos Authenticator environment variables

Following variables can be configured as docker environment:

Variable name	Default value	Description or example	
SATOSA_BASE_HOST		e.g. auth.didmos V2.de	*
SATOSA_STATE_ENCRYPTION_KEY		Random value used for enc of state cookie	*
SATOSA_SSO_ENCRYPTION_KEY		Random value used for enc of sso cookies	*
SATOSA_DISOVERY_ADDITIONAL_PARAMS		If set, additional config parameters are used for the discovery module. Currently the only purpose is settings this to "bypass_target: Saml2UCS" to bypass discovery and directly enter the specified authentication method	
Internal LDAP authentication			
SATOSA_LDAP_ACTIVE	Yes	Activate local didmos V2 login	
SATOSA_REGISTRATION_URL		e.g. https://didmos V2.de/selfreg	*
MongoDB connection			
SATOSA_MONGODB_USERNAME	satosa	Username for mongodb service	
SATOSA_MONGODB_HOST	mongo	Host for mongodb service	
SATOSA_MONGODB_PORT	27017	Port for mongodb service	
SATOSA_MONGODB_DATABASE	satosa	Database name	
SATOSA_MONGODB_PASSWORD		Password for mongodb service	*
OIDC Frontend			
SATOSA_OIDC_DYNAMIC_REGISTRATION	No	Allow dynamic registration of oidc clients	
Internal LDAP Credentials			
SATOSA_INTERNALLDAP_URL	ldap://ldap:389	Internal LDAP Host	
SATOSA_INTERNALLDAP_BIND_DN	uid=satosa,ou=accounts,ou=DSA,dc=didmos,dc=de	Bind DN for internal LDAP	
SATOSA_INTERNALLDAP_BIND_PASSWORD	PdefaultWsatosaD	Bind Credential for internal LDAP	
SATOSA_INTERNALLDAP_SEARCH_BASE	ou=data,ou=default-tenant,dc=didmos,dc=de	Search base for users in internal LDAP	
SATOSA_INTERNALLDAP_CREATE_BASE	ou=social-people,ou=data,ou=default-tenant,dc=did	Base DN for creation of shadow accounts	

	mos,dc=de	
PrivacyIdea MFA		
SATOSA_MFA_PRIVACYIDEA_ACTIVE	No	Activate privacyIDEA MFA for local accounts
SATOSA_MFA_PI_URL		privacyIdea URL
SATOSA_MFA_PI_USERNAME		privacyIdea admin user
SATOSA_MFA_PI_PASSWORD		privacyIdea admin password
SATOSA_MFA_PI_CHALLENGE_TOKENTYPES		Tokentypes, which require challenge & response, e.g. "email, sms"
Facebook Social Login		
SATOSA_FACEBOOK_ACTIVE	No	Activate Facebook login
SATOSA_FACEBOOK_CLIENT_ID		
SATOSA_FACEBOOK_CLIENT_SECRET		
Google Social Login		
SATOSA_GOOGLE_ACTIVE	No	Activate Google login
SATOSA_GOOGLE_CLIENT_ID		
SATOSA_GOOGLE_CLIENT_SECRET		
LinkedIn Social Login		
SATOSA_LINKEDIN_ACTIVE	No	Activate LinkedIn login
SATOSA_LINKEDIN_CLIENT_ID		
SATOSA_LINKEDIN_CLIENT_SECRET		
GitHub Social Login		
SATOSA_GITHUB_ACTIVE	No	Activate GitHub login
SATOSA_GITHUB_CLIENT_ID		
SATOSA_GITHUB_CLIENT_SECRET		
SAML2 Login		
SATOSA_SAML2_ACTIVE	No	Activate Saml2 login
SATOSA_SAML2_METADATA		URL to Saml2 metadata
SATOSA_SAML2_WAYF_ACTIVE		Use WAYF yes/no
SATOSA_SAML2_WAYF_URL		URL to WAYF
SATOSA_SAML2_METADATA_SIGNED		Set to "No"
External LDAP Login		
SATOSA_EXTERNALLDAP_ACTIVE	No	
SATOSA_EXTERNALLDAP_LDAPURL		
SATOSA_EXTERNALLDAP_BINDDN		
SATOSA_EXTERNALLDAP_BINDPWD		
SATOSA_EXTERNALLDAP_SEARCHBASE		
SATOSA_EXTERNALLDAP_SEARCHATTRIBUTE		
SATOSA_EXTERNALLDAP_IDATTRIBUTE		
UCS Login		

SATOSA_UCS_ACTIVE	No

didmos V2 specific settings

DIDMOS2_CLIENT_PROVISION	If set, automatic provisioning didmos V2 lui client with the value of its redirect_uri is triggered during startup. E.g.: DIDMOS2_CLIENT_PROVISION: https://didmos2-demo.daasi.de