

TSD SYSTEM DESCRIPTION



Foreword

Way back in 2010 sensitive research data was handled on “offline” laptops with no backup. Then came massive amounts of data from sequencers, imaging etc, and all existing solutions collapsed. As an early adopter of high throughput sequencing at the Oslo University Hospital I saw the enormous medical upside of reading the entire DNA and RNA from cancer patients, but we had to manage all IT ourselves, and we soon ran out of floorspace, cooling, money and competence.

In 2012 my unsuccessful research career ended when signing with the University of Oslo IT-Department (USIT) as project-leader of enabling IT-services for sensitive data. This is now known as TSD – “Tjenester for Sensitive Data” in Norwegian (a fun-fact says it was first named LSD for the Norwegian “Løsninger for Sensitive Data”). It turned out that I was much more suited as a (project) leader than researcher, and the IT department had the perfect competence, timing and open-source spirit for such a project. The first years were tough, manual, had lots of hurdles and a lot of interesting problems and solutions. The official launch of our then premature baby was in the spring of 2014, and we had been running a minimum viable product almost since the very beginning.

TSD has now turned into a fully-fledged, mature Platform as a Service for Sensitive Data supporting more than 2000 research projects, two clinics, and a few start-ups. TSD is used by more than 80 organisations and more than 10.000 users. The range of services described in this document is extremely impressive, and the infrastructure is complex but modular, API-driven and clean. TSD was recently documented as the most flexible (range of services), mature and widely used trusted research environment in Europe by the EOSC Entrust project.

Nationally we have been identified as a possible official provider of infrastructure for sensitive data together with our friends at SAFE (UiB) and HUNTCloud (NTNU). The future of this initiative is called NORTRE – Norwegian Trusted Research Environment. As the IT-director of UiO and seen by many as “the father of TSD” I am extremely focused on presenting TSD as a joint venture of the entire IT-department and some very important external sponsors and stakeholders. TSD is a beautiful piece of teamwork and is an example of what local competence, ownership and grit can generate. I am immensely thankful for all the effort everyone has contributed, and is contributing every day, to make TSD what it is, and I am sure that TSD will have a bright future.



Gard Thomassen
IT Director, UiO

Services for Sensitive Data (TSD)

Leon du Toit, UiO-IT

2025-05-28

Abstract

TSD is a special purpose eInfrastructure for sensitive data. It is a multi-tenant platform-as-a-service (PAAS), offering secure login, remote access to virtual machines, APIs for application development, and user support. This is offered to customers such as research projects, clinics, students, centers of excellence, private companies and more. TSD processes data on behalf of its customers based on the establishment of data processing agreements, and service-level agreements. Customers in turn, as data owners, are responsible for establishing their own legal basis for data processing, approved by recognised ethical approval bodies such as data protection officers. TSD develops a set of core research services and APIs in-house, and consumes infrastructure services from other parts of the IT-department at the University of Oslo (UiO). Other parts of the organisation, in turn, consume TSD APIs in order to build research services according to user needs. *Keywords: PAAS, Remote Login, APIs, Sensitive Data*

Introduction

TSD offers time- and purpose-limited secure collaborative extensible computing in the form of projects. A project has a defined set of roles and rights which allow customers to manage access control to their computing resources. Each project operates within a strict set of mandatory access control policies designed for high data security. Most notably, projects do not have internet access from inside TSD, and are completely separate from one another. This isolation from other networks and separation between projects is enforced at multiple levels of the architecture, and will be explained in detail in the rest of this text.

TSD has its own Identity and Access management (IAM) system, which is used to provide APIs for IAM, data transport, resource management, notifications, and High Performance Computing (HPC). TSD has web services for project management, data transfer, survey collection, dynamic digital consent, data sharing, AI-powered transcription of media files, and risk management. Some of these services are available from the internet, while others are only available inside TSD's private network.

TSD also offers secure remote login to virtual machines (VMs). Project members have a variety of options for installing the software they need in order to work with their data - either by themselves, or by requesting installation. Computing services include VMs, containers, and HPC. TSD offers central storage with offsite backup, and managed relational databases. Customers can host their own APIs, and purchase hardware which can be used in the HPC system.

TSD's datacenter is on-premise at UiO, and uses network segmentation and firewalling for implementing project separation and network isolation.

Section 1 of this paper covers the definition of projects. Section 2 explains TSD from a functional perspective. Section 3 gives an overview of core research services, while Section 4 covers optional research services. Section 5 explain how TSD's infrastructure works. The paper ends with Section 6, which covers the operational environment.

1. Projects

Definition

TSD projects are:

1. Time- and purpose-limited
2. Secure
3. Collaborative
4. Extensible
5. Computing
6. Enclaves

Projects are enclaves because, by default, it is not possible for data to flow directly between projects, nor for administrative data in one project to bring about change in another. Additionally, all data flows into and out of projects are controlled. TSD is a multi-tenant platform, where projects and their resources are separate from one another at many levels. Projects are further time- and purpose-limited because TSD users, as data controllers, are granted ethical approval for computing on sensitive data for a specific purpose, and for a specific time period. Before a TSD project has started, and after it has ended, it is not possible for users to access project resources or data.

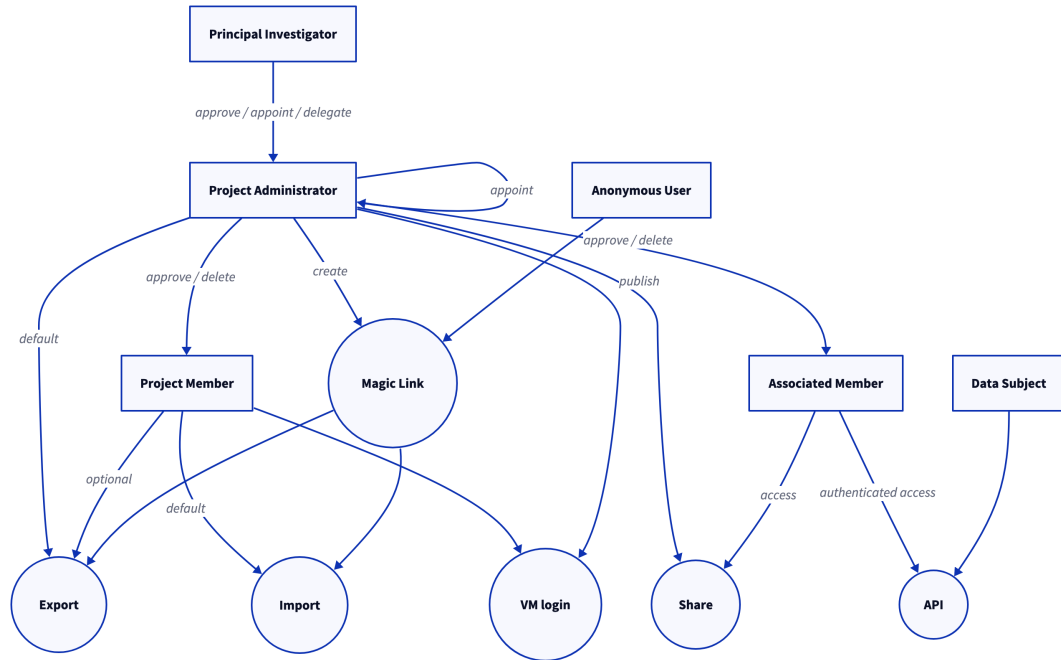
Projects are secure in the sense that TSD guarantees data privacy, confidentiality, integrity and availability in accordance with national laws and regulations, the University of Oslo's policies, and TSD's information security policies. Projects are collaborative environments, where users can work together with data. There are no technical limitations on who can get a user account in a project, or on where the service can be used from. TSD provides protocols for the secure provisioning of users, and secure remote access of data to support highly collaborative projects, across institutional and international boundaries.

TSD projects are extensible in a number of ways. Users can install software on virtual machines (VMs) in user space. TSD offers software repository mirrors for popular scientific programming languages. Software packages that require administrative rights for installation can be installed by TSD staff on request. Projects can also run their own software in containers. Projects can also use TSD's APIs to build their own research services where their use cases require custom business logic and/or user interfaces. Projects can also host their own REST APIs and serve that to clients via TSD's API proxy, re-using TSD's HTTP authentication and authorization services. API clients that are building web services against TSD's data APIs can also request using third-party Identity Providers for authentication on top of TSD identities. Projects can also finance hardware which is added to the High Performance Computing (HPC) cluster.

TSD projects have access to a variety of computing services: in web services, via APIs, on VMs , with containers such as podman and apptainer, or on the HPC cluster.

Roles, responsibilities, and rights

The following figure shows default TSD project roles (square boxes), rights (circles), and their relations (arrows):



- Principal Investigator (PI)
 - responsible for the project’s ethical approval
 - send application for project
 - approve applications for new users
 - appoint Project Administrators, thereby delegating responsibility for such tasks
 - optionally login to VMs, if they choose to create a user for themselves
- Project Administrator (PA)
 - perform the same administrative tasks as the Principal Investigator
 - appoint other Project Administrators
 - export data
 - import data
 - login to VMs
 - create Magic Links
 - approve Associated Members
 - share data with Associated Members
- Project Member (PM)
 - import data
 - login to VMs
- Associated Member (AM)
 - fetch data shared with them
 - authenticate to selected APIs
- Anonymous Users (AU)
 - can use magic links (magic links can be optionally password protected to establish weak authentication)
 - apply for project membership

- Data Subjects (DS)
 - authenticate to selected APIs
 - manage consent

The following default inheritance of rights applies:

- All PIs are PAs
- All PAs are PMs
- All PAs have export rights (opt-out on request)

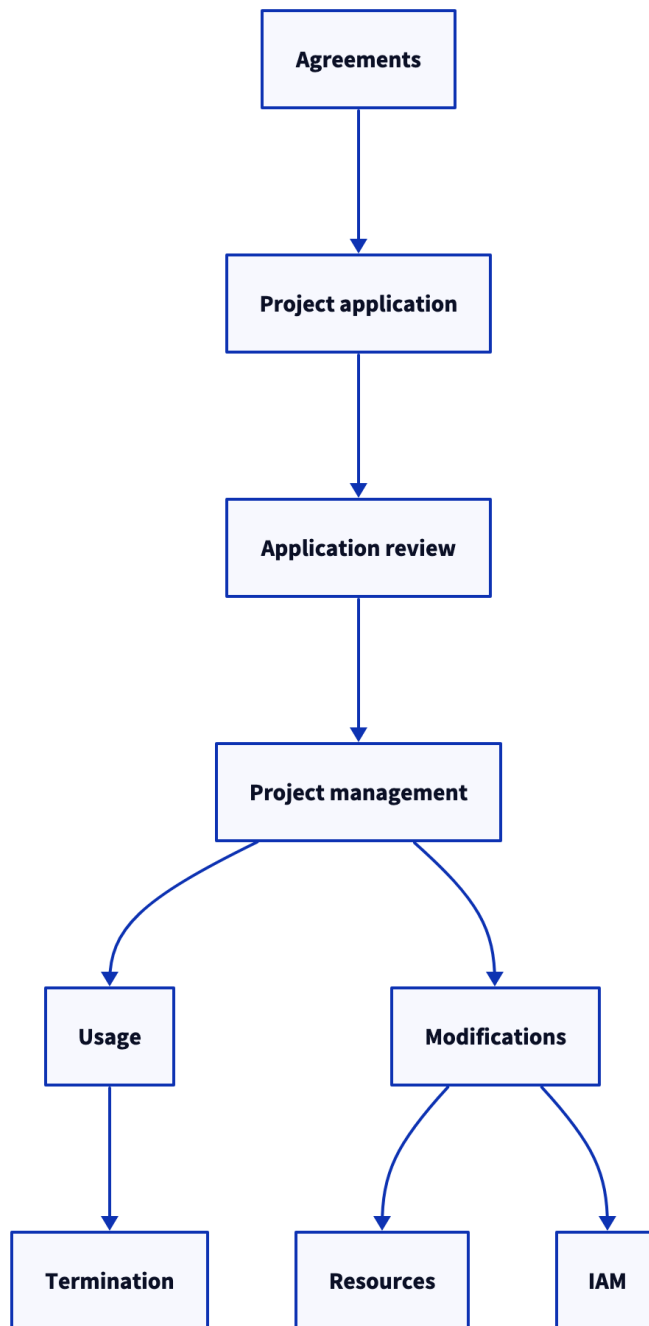
Project life-cycle

Projects establish Service-Level- and Data Processing Agreements with TSD, and are themselves responsible for obtaining ethical approval for processing sensitive data from the relevant Data Protection Office. Applications for the creation of new TSD are submitted via TSD's survey data collection service Nettskjema. There are two scenarios here: one in which the applicant has a Norwegian electronic identity, and all the rest. When applicants can authenticate via ID-porten, Nettskjema fetches their ID number, name, and contact details from Folkeregisteret. Otherwise we request this information from applicants manually, as part of the application form.

TSD staff process such applications. Upon approval, the project, its resources and the person object of the PI are created, and the PI is affiliated with the project. PIs can perform subsequent project management via the selfservice portal. They can login with ID-porten, or they can provision a TSD user with credentials for themselves. The latter is needed in case they want to log in to VMs.

Project management enables service usage for others via the approval of new user applications. Project management also includes modifications of resources such as storage, VM memory and CPU, and Identity and Access Management (IAM) objects such as group memberships to steer access control.

At the time of project application, PIs specify the end date of the project in accordance with their ethical approval. TSD notifies PIs about their project's impending termination thirty days before the end date. When the termination date is reached, it is no longer possible to access any project resources. Ninety days after the end date, project resources are deleted from TSD's systems. If the project requested data archival, only virtual machine resources are deleted.



2. Functional description

This section presents TSD from a functional point of view, describing the processes performed by three classes of persons that interact with TSD: users, staff, and providers. These processes are described with reference to the project roles and life cycle, explained in section 1. Each process has a corresponding process identifier (ID) for the reference purposes.

Users

The following list of processes cover which functions TSD offers to users at a high level. With reference to the inheritance structure of roles, all process marked as applying to PAs apply to PIs, and all processes marked as applying to PMs apply to PAs. Processes marked with the role named **Any** can be performed by anyone regardless of their relation to TSD, or their TSD project status.

ID	Role	Process
P-U1	PI	Project application submission
P-U2	PI	Self-provision Posix user
P-U3	PI	Appoint initial additional project administrator(s)
P-U4	PA	Review project membership applications
P-U5	PA	Appoint additional project administrator(s)
P-U6	PA	Grant export rights
P-U7	PA	Create Magic Links
P-U8	PA	Share data with Associated Members
P-U9	PM	Export data (if authorized)
P-U10	PM	Import data
P-U11	PM	Login to VM
P-U12	PM	Process data inside TSD
P-U13	PM	Uses TSD APIs
P-U14	AM	Fetches data shared with them
P-U15	AU	Use Magic Link
P-U16	DS	Manage their consent
P-U17	Any	Create support case

Staff

TSD staff perform three types of functions: support (SUP), operations (OPS), and software development (DEV). User support can sometimes take the form of either operations and/or software development, but TSD staff also perform operations and software development outside the scope of user support. Many support related tasks will be self-service for users in the future, but are still listed here due to current technical or policy constraints.

ID	Type	Process
P-S1	SUP	Project application review
P-S2	SUP	Grant TSD API usage to projects
P-S3	SUP	Manage storage resources
P-S4	SUP	Manage compute resources
P-S5	SUP	Setup of granular filesystem access control
P-S6	OPS	System configuration
P-S7	OPS	Incident handling
P-S8	OPS	Debugging

ID	Type	Process
P-S9	DEV	Software development
P-S10	DEV	Release, build, deploy

TSD staff members also manage the TSD administrative project p01, and as such processes P-U4 to P-U15 are relevant too.

Providers

TSD's internal service providers deliver data center and software infrastructure services. As such, all the operational and development processes that apply to TSD staff also apply to them.

ID	Type	Process
P-P1	OPS	System configuration
P-P2	OPS	Incident handling
P-P3	OPS	Debugging
P-P4	DEV	Software development
P-P5	DEV	Release, build, deploy

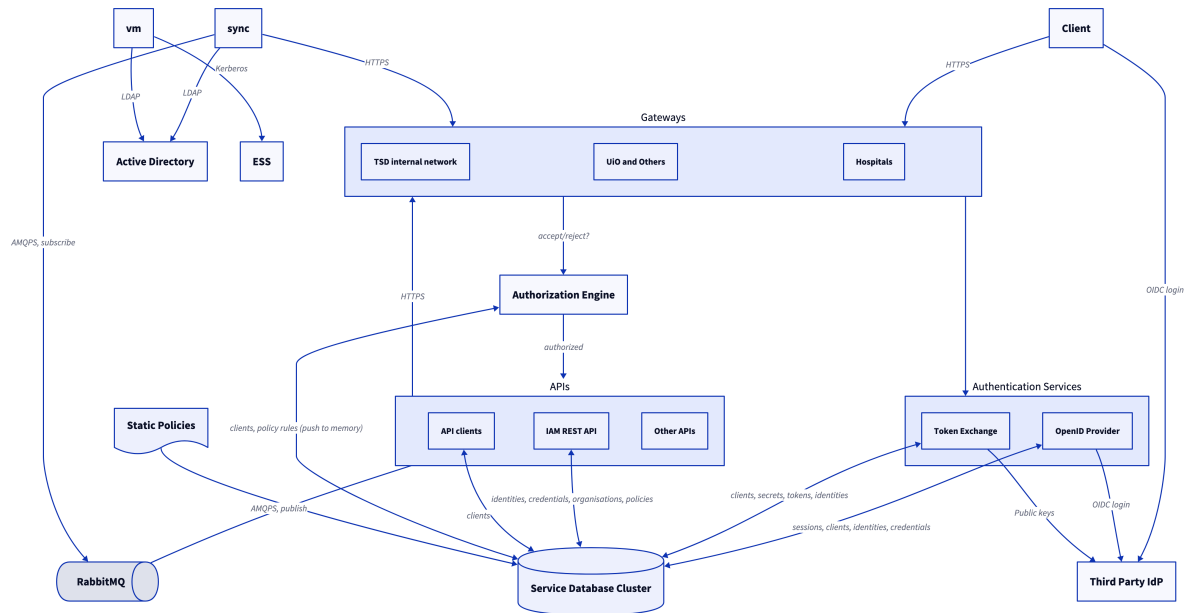
3. Research Services

User Support

Anyone can create a user support case by sending an email to TSD's support email addresses, or by filling in an online form on TSD's help page (TSD (2025e)). TSD's user support organisation is organised as follows: all support cases are handled first by UiO-IT's (University of Oslo's Center for IT) helpdesk. If they are unable to solve the case, it is given to a helpdesk team of TSD superusers. This team receives dedicated training by TSD staff, and is given more access to TSD's operational systems, with the provision that they sign the TSD NDA. If the superusers are unable to solve the case it is given to TSD's support team. Technical specialists and other teams are involved as needed until the case is solved. In addition to daily user support for operations, administration, and policy matters, TSD staff provide paid-for support in the form of hourly rates, and Advanced User Support (Sigma2 (2025a)) funded by Sigma2 (Sigma2 (2025b)).

API architecture and IAM system

All research services make use of the TSD Identity and Access Management (IAM) system. Additionally, it is also a service that is available to users. The API architecture, along with this system is pictured in the figure below:



Gateways

The TSD API has three different gateways: an external gateway hosted at `api.tsd.usit.no` which can be used by clients running outside of TSD's private network, and internal gateway hosted at `internal.api.tsd.usit.no`, available only inside TSD, and a gateway for hospital networks hosted at `alt.api.tsd.usit.no`. The same authentication and authorization services are exposed at these different gateways, with different configuration matching the relevant security policies - such as whether login without a second factor is allowed or not. All API requests are routed through gateways that offload request authorization to a central Authorization Engine. Gateway access is governed by TSD's central firewall. All API traffic is encrypted with TLS certificates.

Authentication services

There are two types of authentication services: an OpenID Connect Provider (OpenID-Foundation (2025)), and a Token Exchange service. The TSD OpenID provider integrates with TSD identities managed by the IAM system (TSD (2025f)). It supports the Authorization Code Flow ("OIDC Code Flow" (2025)) with optional Proof Key for Code Exchange (PKCE) ("PKCE" (2025)) extension. TSD identities can have linked accounts, used for purposes such as account bootstrapping, credentials reset, and login to selected services via trusted third parties. It is therefore possible to login at the TSD OpenID provider with credentials managed by TSD IAM, and/or with third party identity providers via linked accounts. Such third party login is granted explicitly to API clients, and not available by default. Two-factor login supports both time- (IETF (2011)) and HMAC-based (IETF (2005)) One Time Passcodes (TOTP / HOTP).

To gain access to APIs, clients need to obtain access tokens from the token exchange service. ID tokens obtained from the OpenID provider can be exchanged for explicitly scoped API access tokens. Access tokens act like capabilities in the sense that they grant the holder of such tokens access to a potential set of HTTP operations and API endpoints. Instances of access token issued to API clients are parameterised by user-specific attributes, which further limit the authorized set of HTTP operations and endpoints.

API tokens can also be obtained without *user login*, by means of client authentication or instance-based authentication (used to implement magic links). An instance is a pre-issued access token that can be obtained by clients without user authentication - by providing instance IDs, which can be optionally password protected. All such instances are time- and purpose-limited.

API multi-tenancy is enforced by issuing access tokens as symmetrically signed HMAC-SHA256 JSON Web Tokens (IETF (2015)) with project-specific secrets. It is therefore impossible to use an access token obtained for one TSD project at another project. Access tokens are also gateway-specific, so cross-gateway usage is not possible.

Clients

To gain access to authentication services and APIs, clients must be registered. Registration and client access management is performed by TSD staff. Clients are granted access to specific projects, authentication methods (and OIDC scopes), and access tokens based on their intended usage. Clients have a default lifetime of 6 years in total, after which they have to be renewed by TSD staff.

Once registered, clients get API keys / client secrets, that they can use during token exchange to authenticate themselves. These API keys last one year, after which they can be renewed by the clients themselves. Clients also have a verification switch, which TSD staff can use to halt any and all API access if needed. Once a client's verification has been revoked, API access is no longer possible: it cannot obtain new access tokens, and existing access tokens are invalidated by the Authorization Engine.

Authorization

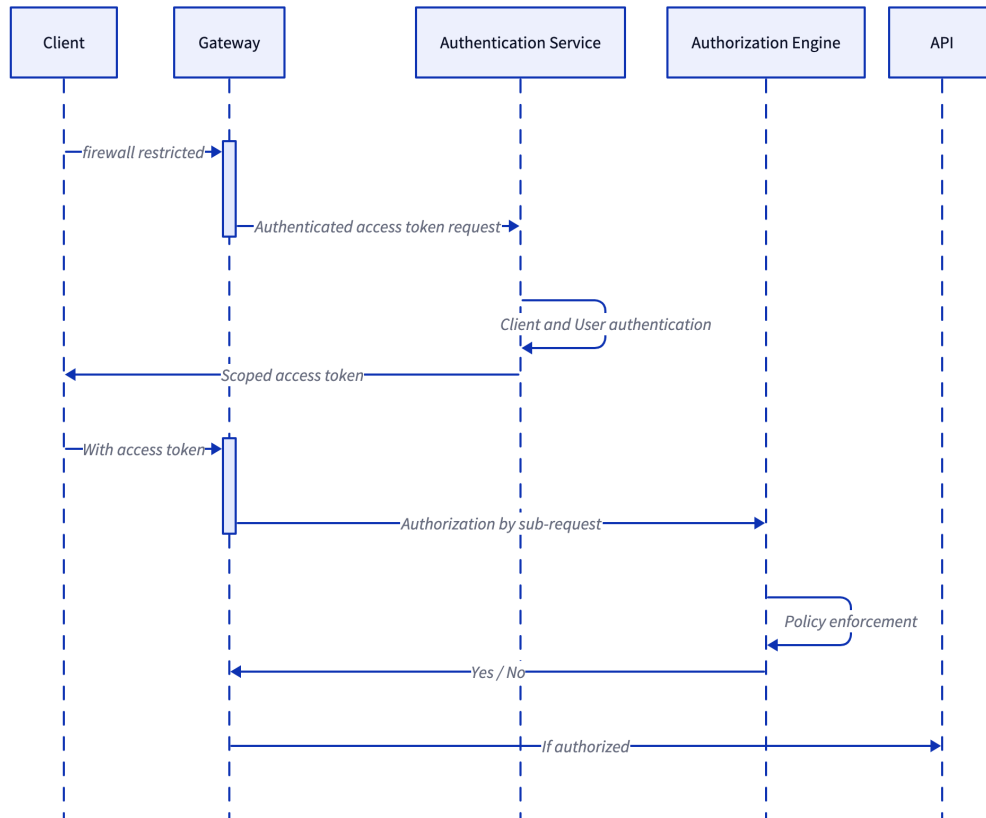
The Authorization Engine acts as a Policy Enforcement Point (PEP) for all HTTP API requests. It works by keeping a set of rules called grants in memory, and evaluating each HTTP request against the relevant set of grants to determine whether it is authorized. Grant data are stored in the IAM database - an open source component called `pg-iam` (TSD (2025a)).

Access control policies are encoded into a set of static grants kept as configuration in a version controlled repository. These static policies are deployed to production by synchronising them into the IAM database. The processes of the Authorization Engine listen to the IAM database on notification channels. New IAM grants are sent to the processes via these channels, and in-memory stores are updated accordingly. Request authorization proceeds according to the following algorithm:

- verify the signature of the access token against a tenant specific secret
- decode the access token claims
 - check whether the client is verified
- find all access control grants matching the current HTTP request: API gateway name, HTTP method, and URI
- for each access control grant, check if:
 - the access token is authorized
 - the client is authorized
 - the user is authorized - determined by group membership, and other attributes
- optionally check
 - time period limitations on the grant
 - limitations on the amount of times the request can be performed
 - any additional client-specific restrictions

In addition to managing IAM grants via version controlled files, the IAM REST API enables such grants to be created, updated and deleted programmatically. This authorization API is utilised by other TSD web services to implement service-specific dynamic access control. All requests, therefore, follow the form shown in the sequence diagram below:

Authenticated API requests



IAM REST API

The Identity and Access Management API provides a REST API for managing authoritative information about:

- persons - TSD unique identifier, name, birth date, credentials, linked IDs
- users - username, POSIX UID
- groups - POSIX groups, web groups
- group memberships - transitive Directed Acyclic Graphs, including temporal constraints
- group moderators - who can manage memberships
- projects - name, start and end date
- institutions - name, hierarchies of organisations
- capabilities - access token parameters
- grants - access control rule parameters
- capability instances - pre-issued access token parameters

- affiliations - relations between objects
- audit - who, what, when for create, update, and delete

The REST API implements TSD-specific semantics on top of the generic `pg-iam` database (TSD (2025a)). Passwords are encrypted at rest. All changes trigger the publication of RabbitMQ messages.

Message Broker

All APIs publish to the central RabbitMQ message broker (Broadcom (2025)) via the Secure Advanced Message Queue Protocol (AMQP). Messages are published to API-specific exchanges with API and endpoint specific routing keys of the form: `k.{version}.{api-namespace}.{endpoint}`. Messages contain request metadata that communicate what was done, but do not transmit actual request data by default.

Any number of queues can be created and bound to exchanges via one or more binding keys. Binding keys are regular expressions that determine which messages are forwarded to the queue from the exchange. Application specific queues can then get access to the messages they need in order to implement their event-driven API integrations. RabbitMQ clients (both publishers and consumers) are configured to require authentication with client credentials, and granted only as much access as needed.

TSD has two instances of RabbitMQ: an internal broker running inside the private network, and an external broker running on the UiO network. Exchanges and queues are federated between the internal and external brokers when applications need to publish or subscribe to messages across the boundary of the TSD network.

Active Directory Synchronisation

Projects, users, groups, and memberships are synchronised to Active Directory (AD) by a microservice that subscribes to IAM events published to RabbitMQ. The microservice processes IAM events, confirms state with the API, fetches any additional data necessary, and uses LDAP to populate the TSD projects tree in AD. Passwords are included as encrypted data in the IAM events. The microservice can then decrypt the passwords in order to set them in AD. A periodic, batch oriented synchronisation job augments the event driven microservice in order to ensure any data inconsistencies due to dropped events are taken care of.

Core APIs

TSD services use many different APIs, some of which offer generic functionality across different services. These APIs typically also have many different clients that integrate with them. As such they are considered core APIs.

File API

The file API is an open source component called the `tsd-file-api` (TSD (2025j)). It offers a REST API for file uploads and downloads, and related remote file and folder operations such as listing, deleting, and fetching file metadata. Both uploads and downloads are resumable. There are no limits to the file sizes that can be transferred. The basic file and folder operations can be used to implement directory synchronisation for both uploads and downloads. API-managed backups of deleted files are configurable at runtime. If this is used, then deleted files are kept by the API for a specific period of time, within which they can be restored.

The file-based API calls also support configurable request hooks, which allow data to be processed by other programs once API calls have completed. This allows isolating functionality that requires escalated privileges from the main API logic. An example of this is setting permissions on the file system subsequent to file uploads.

In addition to offering a file-oriented API, the `tsd-file-api` also offers a REST API for versioned, queryable, auditable documented oriented JSON data. This backend, which uses an open source library `pysquirl` (TSD (2025b)), allows clients to manage arbitrary JSON data via API.

All API requests can be configured to generate RabbitMQ messages, allowing for event-driven integrations. All API operations on both files and JSON data are recordable in an audit log, which is also available for querying via API. Exactly which REST operations are recorded is configurable at runtime.

End-to-end encryption of API traffic is supported, using libsodium (“Libsodium” (2025)) with X25519 and XSalsa20-Poly1305 ciphers. To use this, clients can:

- fetch the API’s public key
- generate their own secret key
- encrypt the data with their secret key using libsodium’s secret stream
- encrypt their secret key using libsodium’s secret box
- send their encrypted secret along with encrypted data in a HTTP header

The API will then:

- decrypt the encrypted secret key with its own secret key (counterpart to the public key)
- use the decrypted secret to decrypt the encrypted data
- write clear-text to disk

The file- and JSON-related functionality is exposed via different backends, each with their own semantics, and when used in production, each with their own set of access control, implemented via the TSD IAM system. The following backends are defined:

- file import and export - uploads and downloads of files to and from TSD
- survey API - used by Nettskjema
- apps API - files and JSON data, multiple applications are possible per project
- publication API - used for sharing data with associated members
- audit logs - read-only API for viewing request logs

TSD provides a reference implementation of a command-line application for using the API, called `tac1` (TSD (2025d)).

Resource API

The resource API is a REST API for managing reference data on billable resources. Such resources include:

- project costs
- virtual machine resources
 - memory
 - vCPU
- storage quota
- High Performance Computing CPU hours
- Microsoft license usage
- hardware purchases
 - servers
 - HPC compute nodes
- user support (over and above the normal service)
- select API access

The API contains prices and a set of resources that can be managed. These values are managed in a version controlled configuration repository. All changes are subject to review.

The API has logic for processing resource orders in two steps: 1) a request for a resource change is submitted, and 2) an external application approves the request. Some resource requests can trigger application approval automatically, while others may require manual intervention and approval by staff. The API publishes messages to RabbitMQ. Once approved, event-driven infrastructure management microservices propagate the resource settings to relevant systems. All such changes and approvals are audited.

Notification API

The notification API is a REST API for sending notifications to users via various communication channels. Any given notification combines events, and specific message data.

An event defines the specific condition under which a notification is triggered. Each event represents a specific condition that triggers a notification and is uniquely identified by its name, which matches the basename of a file within the events directory. Events are defined in YAML format and are managed in a version-controlled configuration repository. They specify when a notification should be sent, which communication channel(s) to use, who should receive the notification, and the content and format of the message. Historical messages are stored and queryable by API clients. The backend is implemented using pysquril (TSD (2025b)), similar to the `tsd-file-api`.

The notification API publishes messages to the internal RabbitMQ broker. The notification exchange is federated to the external RabbitMQ broker. A notification sender microservice runs outside of TSD’s private network, and listens to RabbitMQ. Upon discovering new notification events, it fetches the relevant event and message data from the notification API, and uses the specified notification channel to send the message. The message status is updated in the API. Relevant access control for the microservice is governed by the IAM system.

HPC API

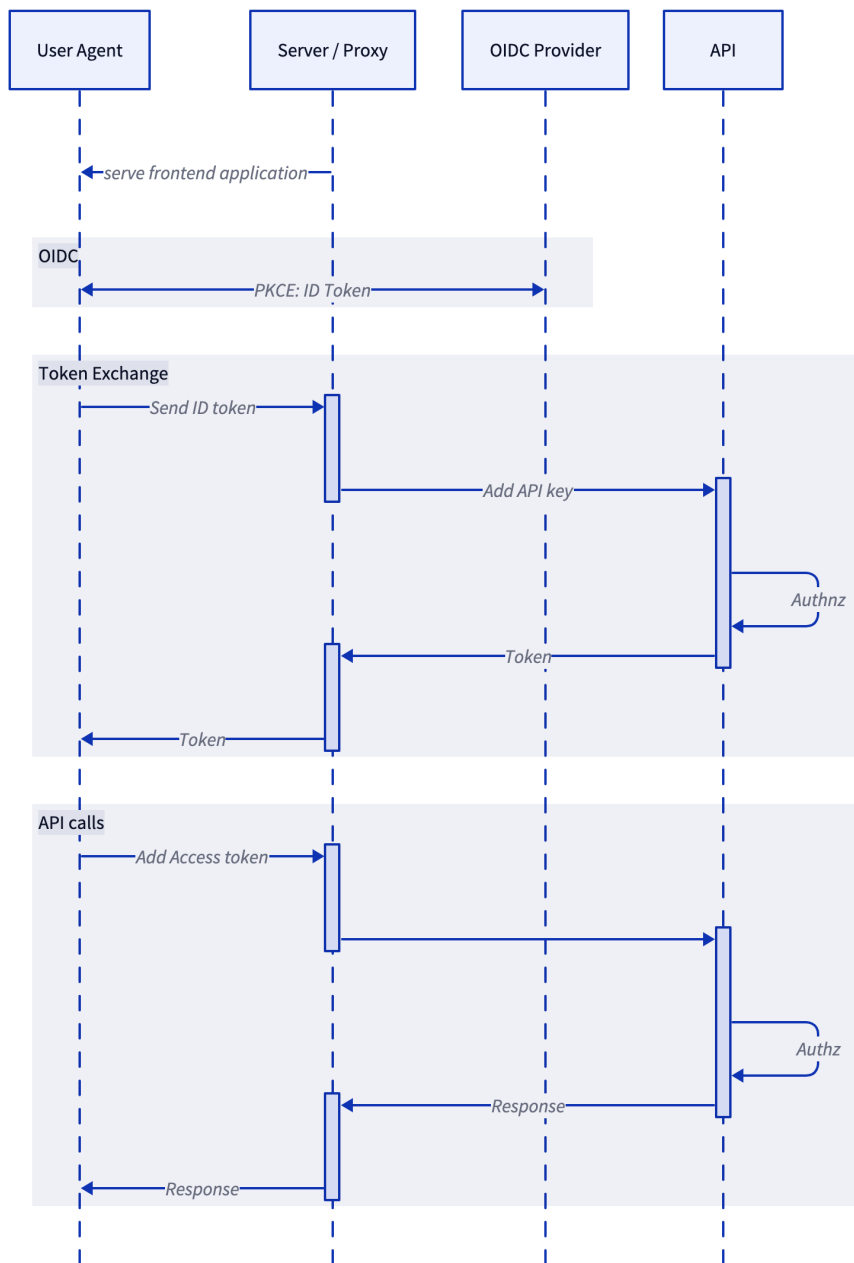
The High Performance Computing cluster, Colossus, uses the Slurm (“Slurm” (2025)) workload manager for scheduling jobs. The HPC API is a proxy for the Slurm REST API (“Slurm REST API” (2025)). It exposes the Slurm API in a way that makes adoption of the TSD IAM system easier. After request authorization by the Authorization Engine, the HPC API decodes the access tokens and re-encodes it in format that is compatible with the Slurm API’s requirements. The JSON payload is passed along unaltered, and API responses are sent to the client as is.

The HPC API can be used by API clients that have the need for HPC resources, such as GPU. It enables serverless web applications, and gives microservices a way of running user-specific workloads in a sandboxed environment: the job itself runs as the POSIX user, which means that the microservice doesn’t need the right to impersonate users, nor does it need wide access to the storage system.

Web services

TSD’s external web services run outside of TSD’s private network, and are hosted in UiO’s datacenter. Internal web services run inside TSD’s private network and are available from project VMs. Such web services integrate with TSD authentication services and APIs. The data flow involved in such integrations is shown in the sequence diagram below:

Data flow



Websites are served to users by web proxies, which also proxies all subsequent requests. Clients running in the user's browser authenticate users and themselves using the OIDC code flow protocol with the PKCE ("PKCE" (2025)) extension to obtain ID tokens. The PKCE protocol helps mitigate session hijacking while running the Authorization Code Flow, and is mandatory for browser-based TSD API clients.

Clients then exchange the ID token for specific API access tokens. This exchange requires client authentication, to ensure that only ID tokens issued to the client served by the web proxy can be used at the API. Access tokens contain signed information about the authentication event, the client and the user.

API calls made from browser-based applications require client authentication in addition to API access tokens. The client sends the access token stored in the browser session when calling the API via the web service proxy, and the proxy adds the client's API key in an additional header. This ensures that access tokens used via the web service proxy must have been issued in connection with authentication handled by the same application. This restriction is enforced by the Authorization Engine. The specific API call is enforced by the Authorization Engine as usual.

Selfservice

The selfservice portal, an instance of an external web service, implements functionality to facilitate user processes P-U1 to P-U7, and more (TSD (2025g)). The website allows login with TSD credentials and with linked accounts via ID-Porten (Digdir (2025)). The portal provides services for the following:

- reviewing and managing personal data
 - profile: name, date of birth, email, ID number
 - review privileges: what you are entitled to do
 - set password (according to UiO's password guidelines)
 - set passcode (for Two-Factor login)
- project administration
 - review you project
 - manage membership applications
 - invite users without Norwegian eIDs
 - manage current project members: delete, disable
 - manage user privileges: export, admin, and more
 - moderate group memberships: advanced privilege management
 - access to VMs
 - create and manage magic links
 - review export logs
- registering applications
 - project membership
 - new TSD projects

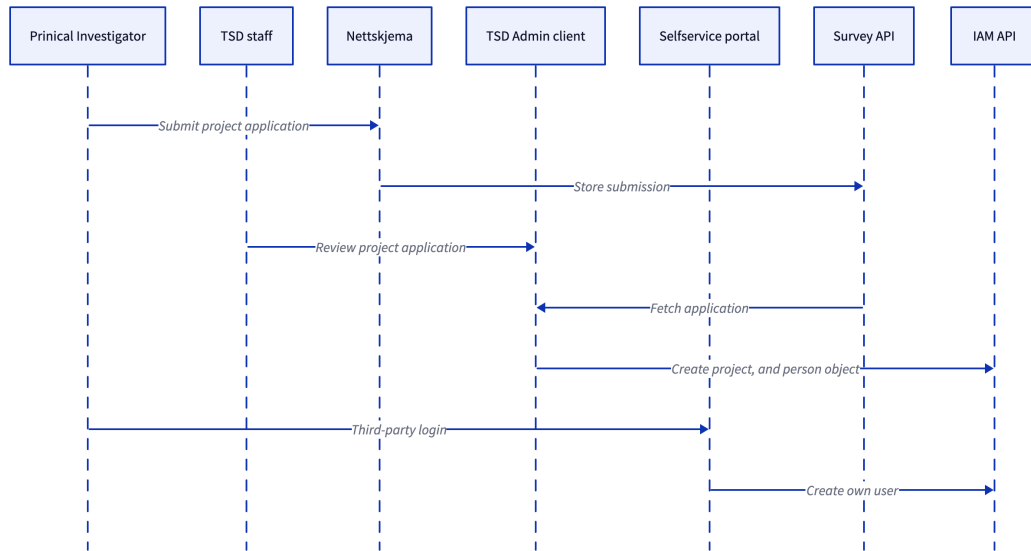
All functionality is implemented by integrating with TSD's APIs, and calling these APIs on behalf of authenticated users: File-, IAM-, Resource-, and the Notification API, all of which are embedded in the IAM system. The portal, therefore, does not have any special API access other than enabling it to act on behalf of users in the necessary ways, and does not implement any access control itself.

User creation protocols

The selfservice portal supports user creation for three different cases: 1) Principal Investigators *with approved electronic IDs*, 2) Project Members *with approved electronic IDs*, and 3) Project Members *without approved electronic IDs*.

The protocol for PI user creation, with approved electronic IDs, is shown below:

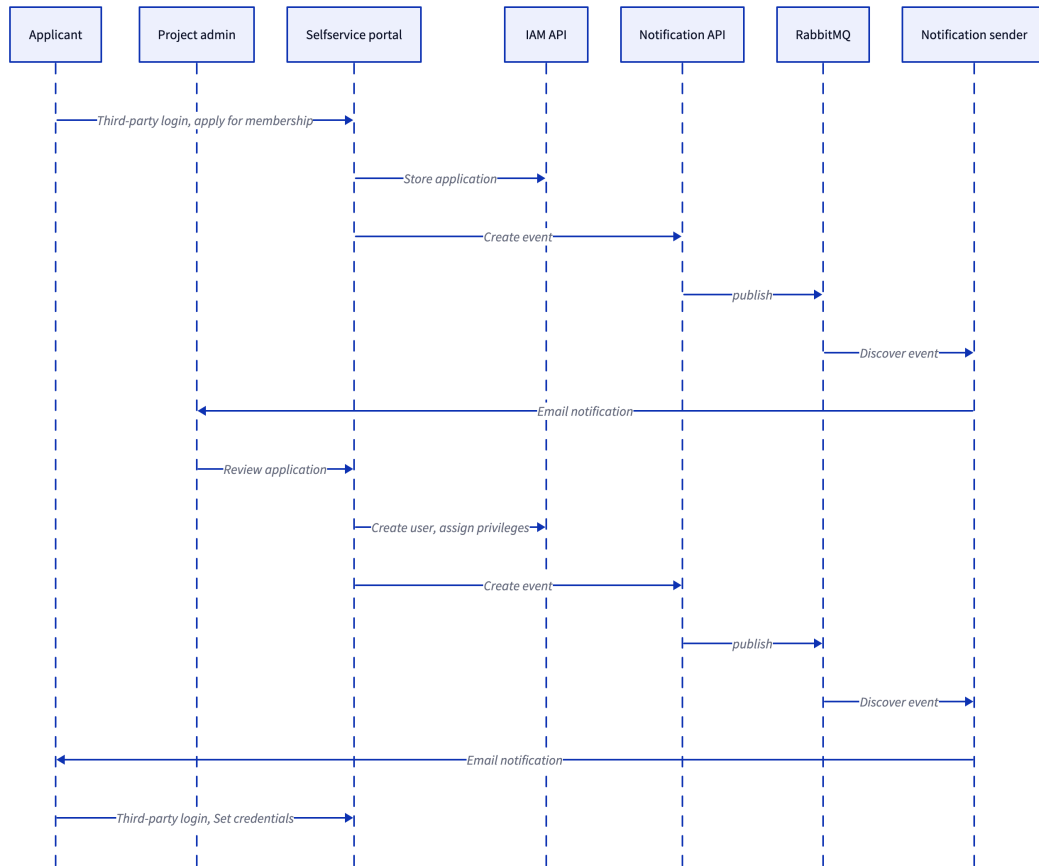
PI user creation



PIs submit TSD project applications with Nettskjema. In doing so they authenticate with ID-porten, and their national ID numbers are stored inside the application itself. TSD staff then reviews the application. Upon approval, the project and person object are created in the IAM system. The reference to the national ID number allows the PI to login to the selfservice portal with ID-porten as a third party. They can create their own TSD user.

The protocol for creating project members with approved electronic IDs, is shown below:

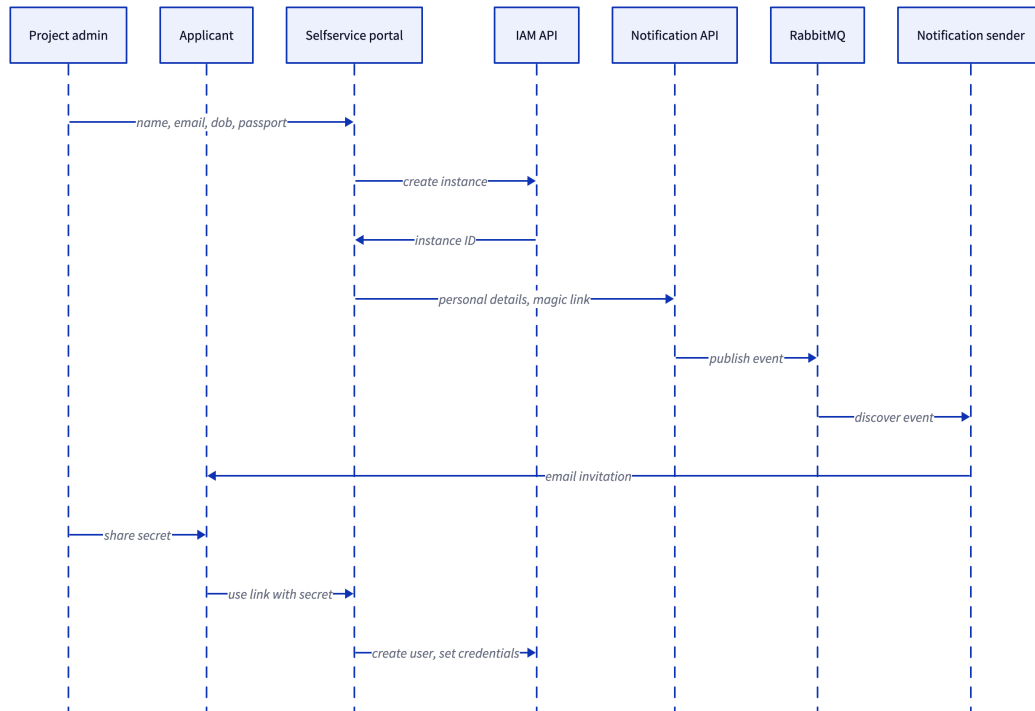
eID-based user creation



Applicants use third-party login in the selfservice portal to submit a project membership application. The selfservice portal stores their national ID number along with their application. Project admins are notified about the new application via the notification API and notification sender. No personal details are sent. Project admins review the application for membership. If approved, the user is created, and the applicant is notified via email. They can then login to the selfservice portal, and set their TSD user credentials.

The figure below illustrates the protocol for creating project member without approved electronic eIDs.

Non-eID user creation



A project admin uses the selfservice portal to invite a user. To begin with, they provide the applicant's name, email, date of birth, and passport details. TSD does not provide services which the admin can use to collect such information, so it happens outside of the scope of the selfservice portal itself. Once provided, the selfservice portal calls the IAM API, creating a time-limited authentication instance (a deferred scoped access token), storing the information about the applicant in the instance metadata. The returned link is sent to the notification API creating a notification event. The event is published to RabbitMQ, and discovered by the notification sender, which sends an email to the applicant. The admin can see the link-specific password in the selfservice portal and shares this with the applicant in a out-of-band channel. TSD recommends a video call. The applicant then follows the link, uses the link-specific password, reviews their information, and triggers the creation of their own user account, resulting in an API call to the IAM API. At this point they have a valid login session which means they can immediately set their credentials.

File transfer

Data portal

The data portal (TSD (2025i)) is an instance of an external front-end only web service that uses the File API, offering the following functionality:

- login with SSO with TSD credentials
- import files

- resumable, no limit on file size
 - set access control by specifying group permissions
 - with an import link, using instance login
- export files
 - no limit on file size
 - with an option to delete the file
- media capture
 - capture audio and video in the browser, streaming it to TSD

API clients

TSD maintains a reference implementation of a file transfer client, in the form of a command-line program called the `tsd-api-client`, or `tacl` (TSD (2025d)). `tacl` can be used from whitelisted IP addresses that have access to the TSD API. It is published on the Python Package Index, and can be installed from there in user space. Once installed, users must register their installation *per project*. The process of registration requires the user to authenticate to a specific TSD project. Upon success, an API key which is valid for a year is issued and stored in local configuration. This API key allows the user to authenticate themselves when doing file transfers, obtaining an access token and making authenticated HTTP requests as needed. `tacl` can also be used with standalone API keys that are issued by TSD staff, and configured to get access to specific projects.

The `tsd-api-client` support the following functionality:

- listing files and directories
- resumable file- and directory upload and download
- resumable synchronisation of directories
- controlling sync behaviour (whether to propagate deleted files)
- exclusion of specific files based on pattern matching
- deleting files in the export directory
- group-based access control
- end-to-end, streaming encryption
- automation of upload and download with API keys
- import-link based authentication (with an API key)

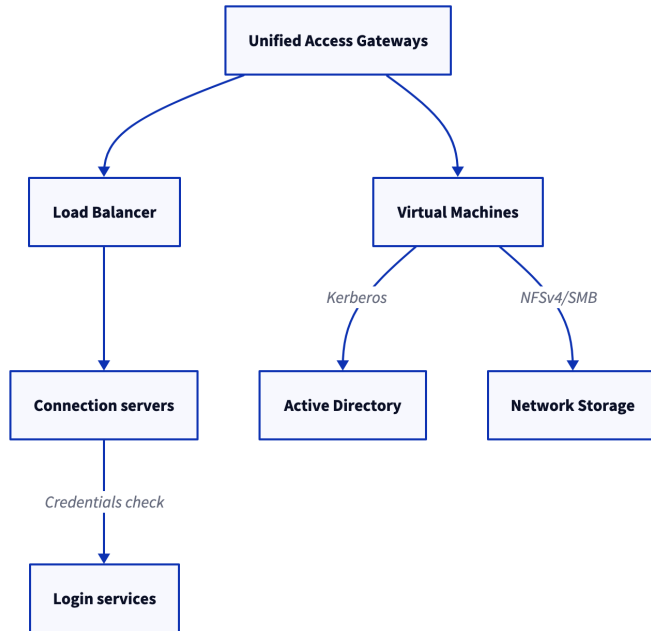
In addition to the File API, TSD has limited support for uploads using s3 - only a small subset of the s3 protocol is exposed via a TSD maintained wrapper, `tsd-s3cmd` (TSD (2025l)), for the open source `s3cmd` software (“S3cmd” (2025)). This is set up on a per-project basis. s3 clients need to register in the same way as for the `tsd-api-client`. Once files have been uploaded to remote buckets, they are accessible via folders on the filesystem inside TSD.

Disk-based import

TSD also offers physical disk-based import. Users can deliver encrypted disks to TSD, and TSD staff will import this to their project’s storage using a physical host inside the TSD datacenter.

Remote login to VMs

All TSD projects can request the provisioning of virtual machines that allow them to get interactive access to their data through secure remote login, using TSD’s login services. Login is controlled by Unified Access Gateways (UAGs), while login and VM connection is managed by load balanced connection servers. TSD has two UAGs: `view.tsd.usit.no` which is available from anywhere, and `view-ous.tsd.usit.no` which is available from the Oslo University Hospital (OUS) client network. Copy-paste and printing is disabled on `view.tsd.usit.no`. Users do not have root access on VMs. TSD’s administrative/infrastructure project (which runs central services) runs on separate virtualisation clusters than customer projects.

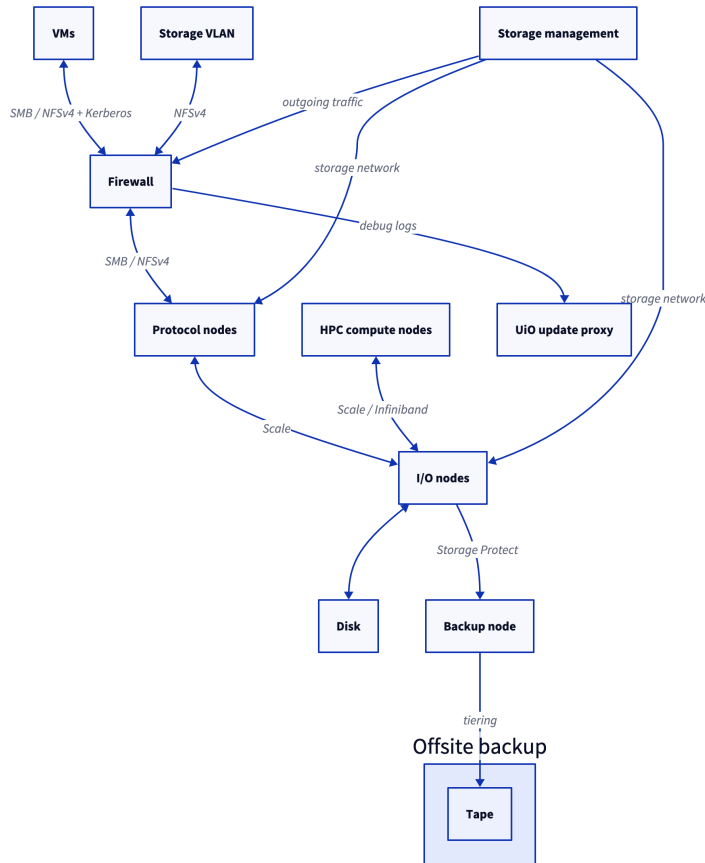


VMs can have either Windows or Redhat Linux operating systems. Virtual machines have a default amount of CPU and memory, which can be changed upon request. VMs have access to the project’s storage area via SMB and NFS protocols. Users must have valid Kerberos (“The Kerberos Network Authentication Service (V5)” (1993)) tickets in order to access their project storage. File system access control is enforced with folder and file permissions, based on groups.

Each project has its own IP subnet in which VMs are provisioned. Project subnets are separated from one another based on private VLANs implemented by the TSD firewall (more details in the infrastructure section). Project storage mounts are exported to the specific subnets. All project virtual machines have access to TSD’s internal API gateway, internal web services, and the HPC queue system.

Storage and backup

TSD uses IBM Storage Scale (“IBM Storage Scale” (2025)) for data storage, and IBM Storage Protect for backup (“IBM Storage Protect” (2025)). The storage service is organised in the following way:



As explained in the section on remote login, TSD project VMs access storage using either NFSv4 or SMB with Kerberos authentication, via the TSD firewall. The IBM storage system consists of disk, I/O nodes, and protocol nodes - the latter serve both NFS and SMB traffic. NFS exports are project-specific, and SMB uses ACLs, signing and encryption. All project filesets are exported to the HPC cluster. The following directories are created for all projects:

- /tsd/{project}/data - shared project storage
- /tsd/{project}/home/{user} - user-specific directories
- /tsd/{project}/data/durable - shared by all project members, backed up (offsite, with 90 day retention)
- /tsd/{project}/data/no-backup - shared by all project members, *not* backed up
- /tsd/{project}/.snapshots - nightly snapshots, retained for 7 days

The storage cluster is managed from a management node via a separate internal network. All changes are audited. The firewall allows for sending operational logs to IBM via UiO's update proxy, and for sending email if needed.

Software

Users can run software on their VMs, and on the HPC cluster. Software is made available in the following ways.

User-space installation

Users can upload source code through the standard TSD file import services and install software in user space. This can be done on Linux VMs, and the software can be used there and in HPC jobs.

Containers

TSD supports rootless containers such as Apptainer (“Apptainer” (2025)) and Podman (“Podman” (2025)). Apptainer is a daemonless container that can be run on linux VMs, and on the HPC cluster. TSD provisions dedicated virtual machines for running podman containers in rootless daemon mode.

Podman containers run as local system users on the dedicated VMs. A corresponding TSD IAM system user is created to grant access to project data. User namespace mapping ensures that commands executed by TSD project users are mapped to the podman system user, which gains access to data via the IAM system user. All other users in the container namespace are unprivileged.

Linux repositories

TSD staff can install software from yum repositories on behalf of users. Access to yum repositories from VMs running inside TSD’s private network is governed by the TSD firewall. TSD grants access to UiO’s own yum repositories, and to official Redhat Network repositories. The latter are exposed via proxies maintained by UiO-IT.

Windows software center

SCCM packages are built by staff in the UiO domain, imported to the TSD management project via the TSD data portal, and then imported to TSD SCCM via the SCCM console. Packages are then deployed to either the device collection with all machines, or a user collection corresponding to a specific license group, if the software requires a license. A scheduled task sets members of project’s admin group as Primary Users of Windows VMs in SCCM. This grants them rights to install programs through the Software Center.

Mirrors

TSD provides the following internal software package mirrors:

- PyPI: Python Package Index - uses the **bandersnatch** (“Bandersnatch” (2025)) mirror tool
- CRAN: Comprehensive R Archive Network - using **rsync**
- Bioconductor: Software for Bioinformatics - using **rsync**
- CPAN: Comprehensive Perl Archive Network - using **rsync**
- Stata: Statistical software for data science - using a git repository hosted by UiO-IT

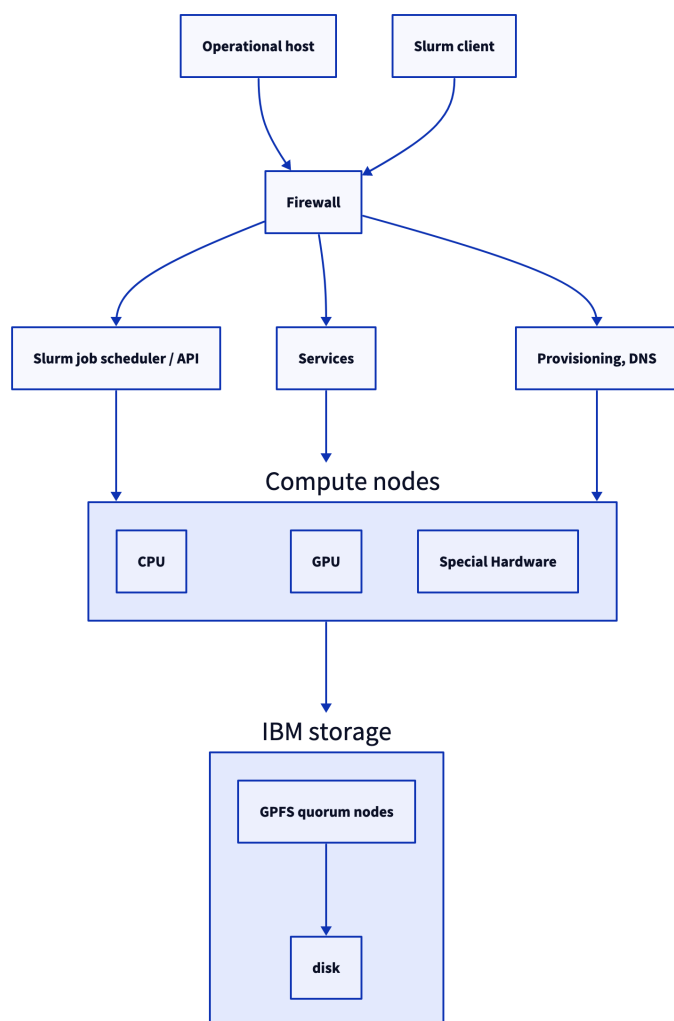
All packages are pulled into TSD, with a two-week delay enforced as a security measure, by programs running in a specific administrative VLAN: a VLAN which has outgoing access to resources outside of TSD, but also access to TSD storage. Once the package data is stored, it is served via a web server proxy inside the TSD network, available to all project VMs, and the HPC system.

Modules

A collection of HPC software, compiled for the target architecture, is built using EasyBuild (“EasyBuild” (2025)). Software is built on a physical host outside TSD, and synchronised to TSD storage in the same way as for software mirrors, using **rsync** from a host in the dedicated TSD administrative VLAN.

HPC

TSD's HPC cluster, named Colossus, consists of CPU, GPU and special hardware nodes (such as Illumina Dragen, and nodes with lots of memory) for computing. These nodes are connected to project storage and are available to TSD users and staff as shown below.

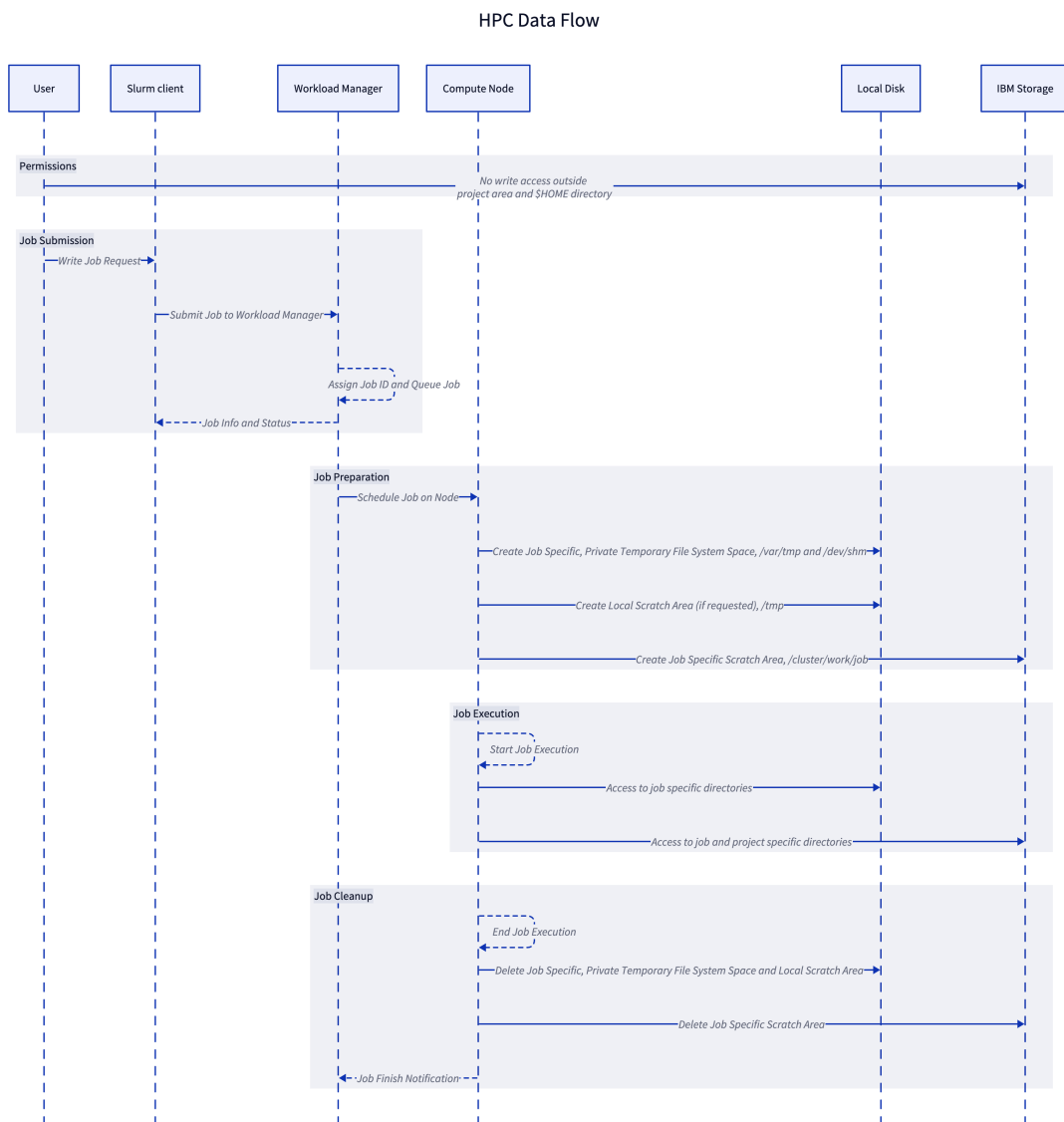


Compute nodes run Rocky Linux (“Rocky Linux” (2025)), and are connected to one another, and the IBM storage system via high performance InfiniBand network. The current setup is:

- 65 compute nodes: 64 cores, 512GB RAM
- 4 big memory nodes: 128 cores, 4TB RAM, 7TB NVM3 local scratch storage
- 6 GPU nodes: 2 Nvidia V100, 4 Nvidia A100 each
- 3 Illumina Dragen nodes (“Dragen” (2025))

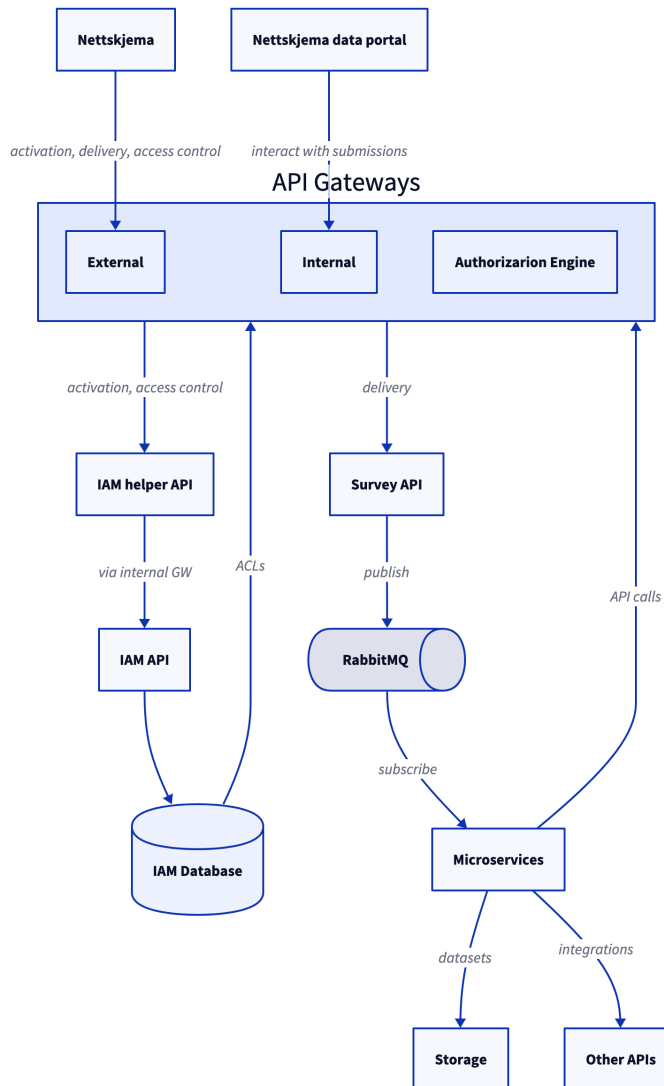
Staff members can access the HPC services host and the provisioning host (which runs internal DNS) via the firewall, from the TSD operational host. Internal services run on a dedicated service host. Users can interact with the HPC cluster via the Slurm (“Slurm” (2025)) job scheduler. Slurm clients can be project VMs using the `sbatch` command, or API clients using the HPC API (which itself interacts with the Slurm REST API).

All compute nodes have access to all TSD projects’ storage. Slurm jobs run as the Posix users that submit the jobs, and as such have only the filesystem access that the users themselves have. Jobs can use software installed in user space, and/or software managed by the EasyBuild module system. Compute nodes have access to TSD API’s via the internal gateway, but otherwise have no external network access. An overview of the job submission process is shown below:



Nettskjema

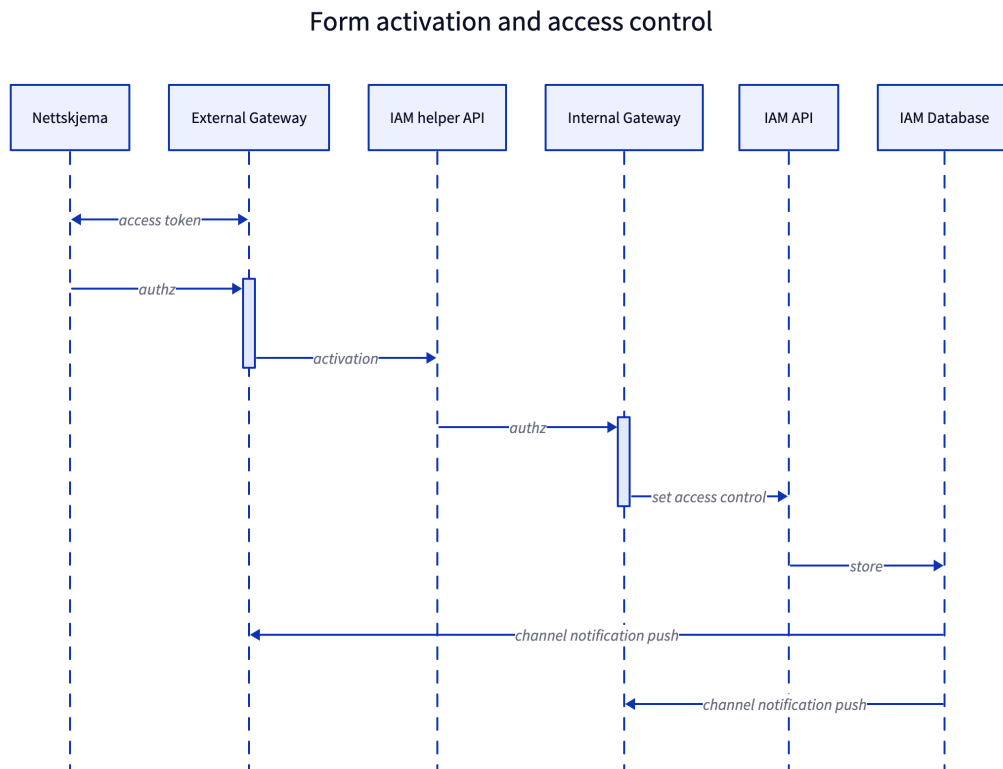
Nettskjema is a survey data collection service developed and operated by UiO-IT, hosted at UiO's datacenter. It offers self-service tools for building survey forms, supports multiple identity providers for authenticated data submission, signed digital consent forms, and is integrated with TSD via the survey API (implemented by the `tsd-file-api` and the TSD IAM system). The overall architecture of the integration is pictured below.



The Nettskjema service has two user facing applications that integrate with TSD's APIs: 1) the Nettskjema frontend that integrates with the external API gateway, and 2) the Nettskjema data portal that integrates with the internal API gateway. The Nettskjema frontend is responsible for form activation, submission delivery, and setting form-specific access control. Activation and access control is done by calling the IAM helper

API, which in turn calls the IAM API via the internal gateway. The IAM helper API simplifies the access control interface offered by the IAM API. Data delivery uses the survey API backend of the `tsd-file-api`. Event-driven microservices that process Nettskjema data, subscribe to RabbitMQ messages published by the survey API, and manage their own API integration as normal API clients, controlled by the TSD IAM system.

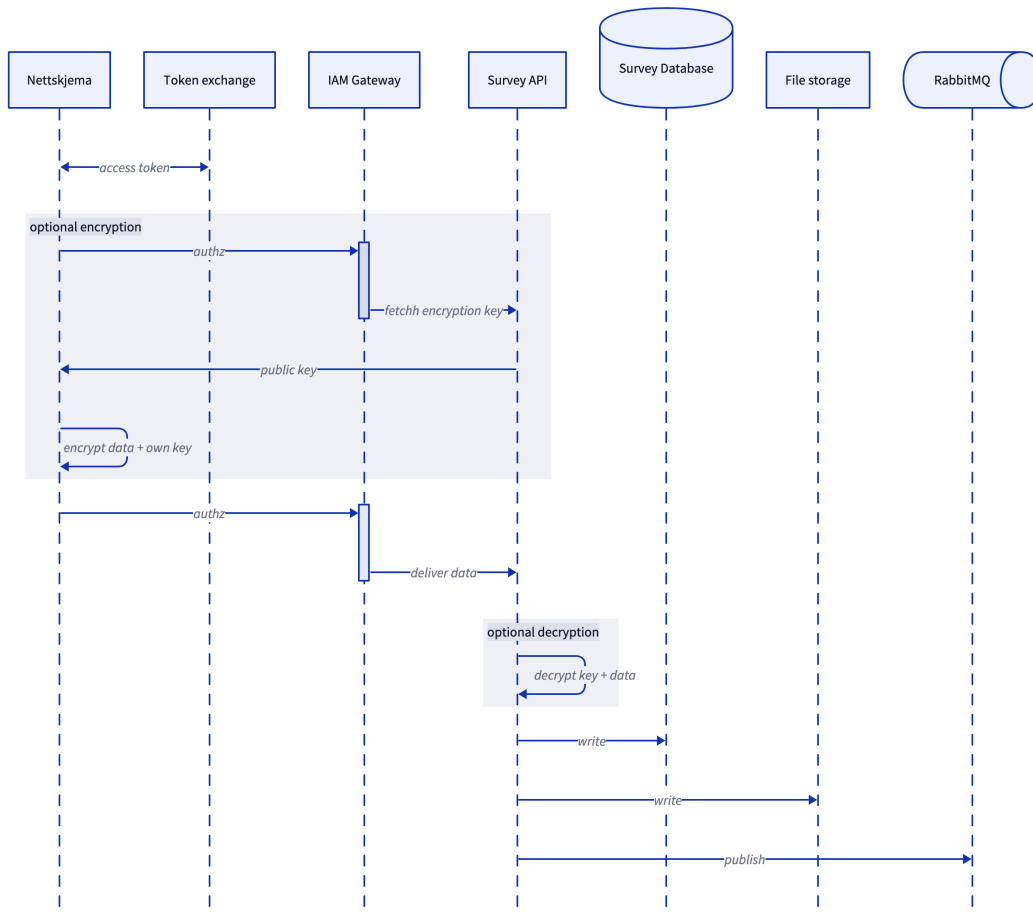
The sequence diagram below shows a detailed view of form activation and access control:



Data delivery is not possible before forms are activated for delivery to specific projects. Such activation requires that users define form specific parameters, perform OIDC login with their chosen TSD project and choose activation. The Nettskjema application obtains an access token, and calls the IAM helper API via the external gateway. The gateway determines whether the request is authorized by sending a sub-request to the Authorization Engine. The IAM helper API translates the Nettskjema API call into the appropriate set of IAM API calls, and does so via the internal gateway. These calls are also checked with the Authorization Engine. The IAM API stores new access control rules in its database. The database pushes the new access control rules to the Authorization Engine via channel notifications. The exact same process is followed when setting access control on forms at both the API and filesystem level. Such access control determines who can read, update, and delete form submissions in the Nettskjema data portal.

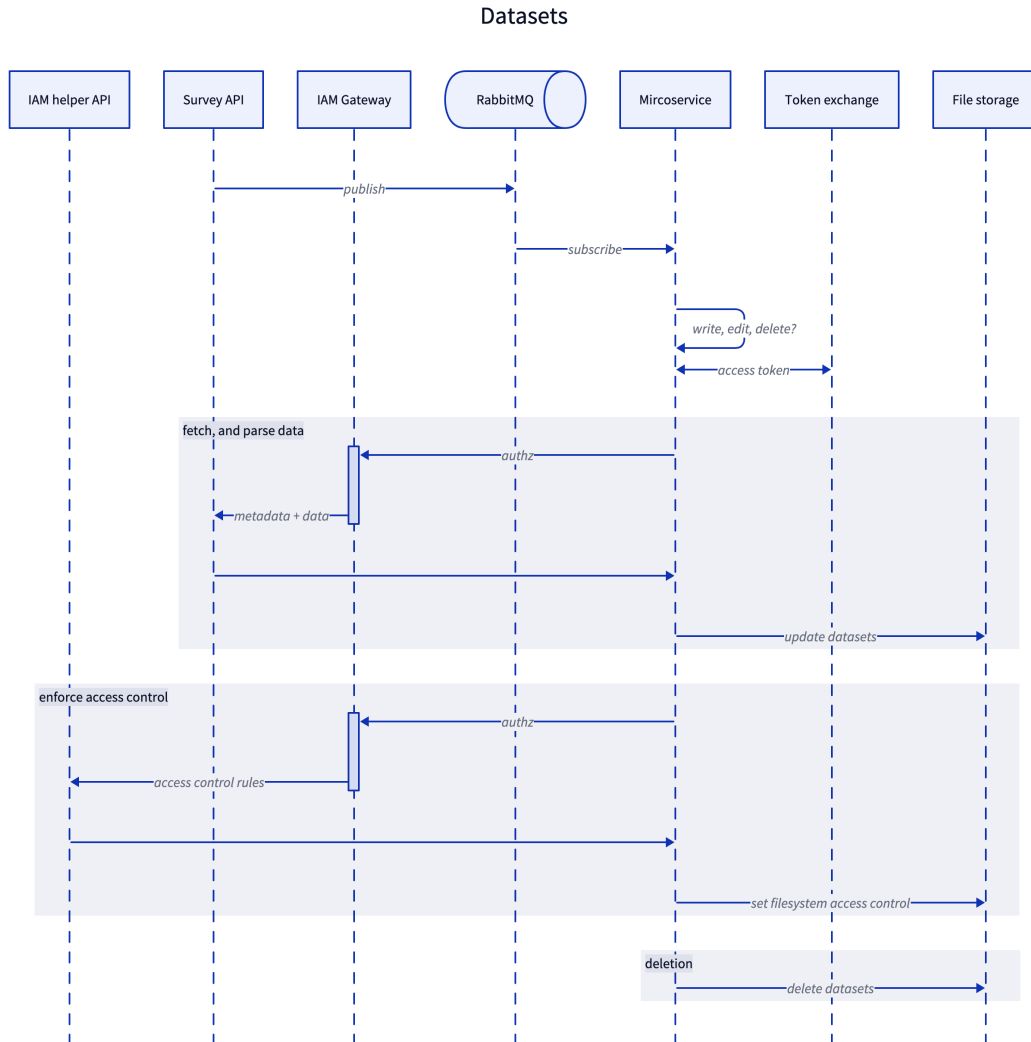
The data delivery sequence diagram is shown below:

Data delivery



Nettskjema obtains an access token for data delivery to a specific project. If the survey API is unavailable, Nettskjema can use the end-to-end encryption functionality of the `tsd-file-api` to securely cache data. This allows Nettskjema to potentially have higher uptime than the TSD API without compromising data security. Nettskjema then sends data along with the access token to the survey API via the external gateway. The request is authorized by the Authorization Engine, ensuring that the form has been activated for the given project. If the data was encrypted, the survey API decrypts the data, and stores it. In the case of JSON data, it is stored in a multitenant PostgreSQL database. In the case of attachment data, files are stores on TSD's central file storage, in a project-specific folder, that is only accessible by the API process. RabbitMQ messages are generated after successful data delivery.

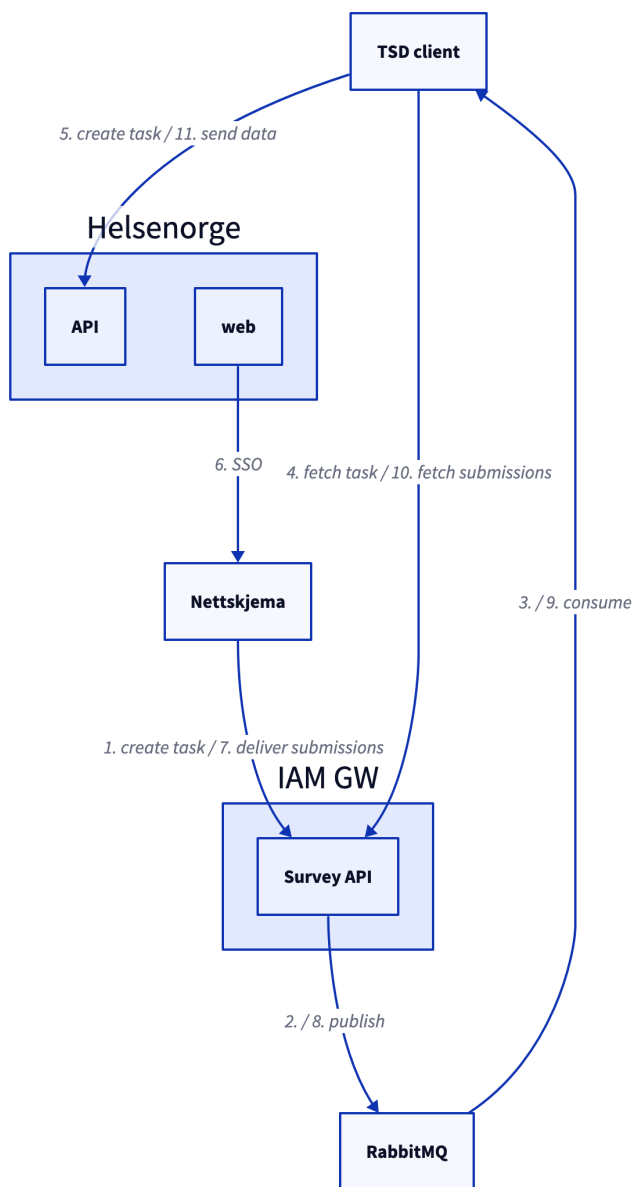
In addition to having access to form submissions via the Nettskjema data portal (and API), TSD automatically produces datasets in CSV format, stored in the project's storage area. The relevant sequence diagram is shown below:



The microservice subscribes to RabbitMQ messages produced by the survey API. It processes incoming messages determining which action is appropriate based on whether it is a *write*, *edit*, *delete* or *access control* operation. In the case of *write* and *update* operation - when new submissions or attachments have been delivered - it fetches an access token for the specific project, fetches the new data from the API, parses it with reference to form metadata if necessary, and updates the datasets accordingly. Handling *delete* operations is simpler, because no new data needs to be fetched from the API. Relevant submissions or attachments can simply be removed from datasets. Access control is enforced on folders and files according to preferences expressed in the IAM API.

Helsenorge integration

Nettskjema and TSD offer projects the ability to integrate their forms with Helsenorge (“Helsenorge” (2025)). A pre-requisite for doing so is that the institution responsible for the TSD project, must have the right to use Helsenorge - something that is clarified with Norsk helsenett (“Norsk Helsenett” (2025)). Once clarified, TSD and Nettskjema set up the integration for specific forms in dialogue with the TSD project. Integrated forms are configured in a version controlled repository, subject to code review by both TSD and Nettskjema teams. Once configured, forms are activated for delivery and Helsenorge usage by the project.

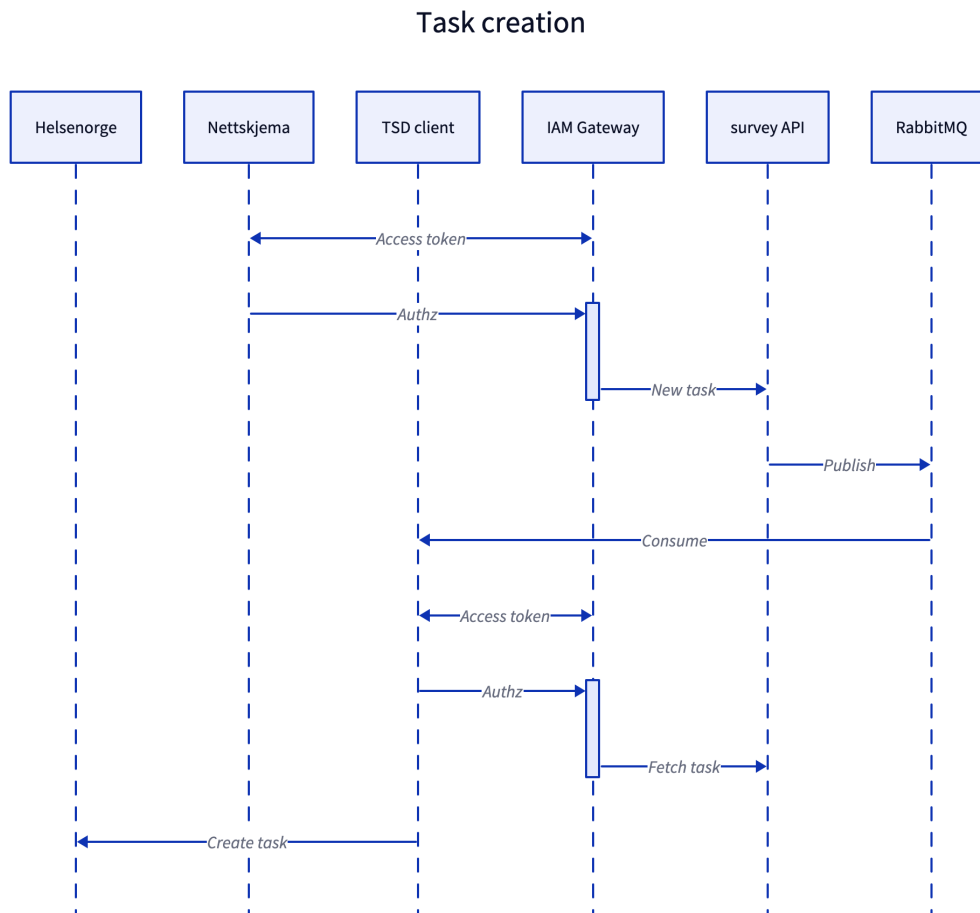


The technical integration has 11 distinct steps:

1. A task is created in Nettskjema, calling the survey API
2. The survey API published a RabbitMQ message
3. The TSD client consumes and processes the RabbitMQ message
4. The TSD client fetches relevant task data

5. Discovering that a new task has been created, the TSD client calls the Helsenorge API
6. A citizen logs into Helsenorge, clicks a link to Nettskjema with Single Sign-On
7. Data is delivered to one or more forms connected to the task
8. The survey API published a RabbitMQ message
9. The TSD client consumes and processes the RabbitMQ message
10. The TSD client fetches task-specific submission data
11. The TSD client calls the Helsenorge API, sending a report to the citizen

These steps can be grouped into two main actions: 1) task creation, and 2) task completion. Task completion can be further divided into data delivery, and reporting. The sequence diagram for task creation is shown below:

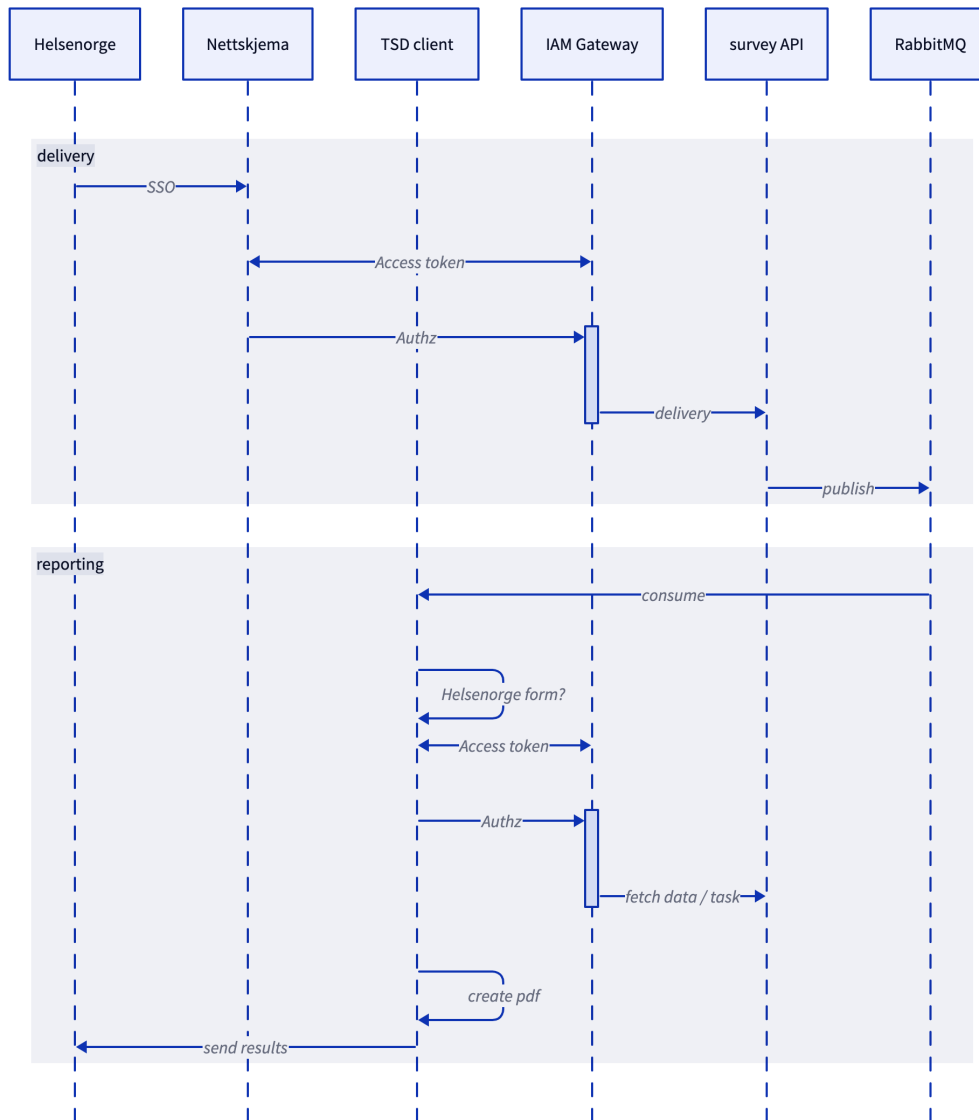


Nettskjema obtains an access token subsequent to a user login with the TSD OIDC provider. The user provides the national ID number of the person for which the task should be created. Nettskjema generates a unique task ID, submits the data along with the access token to the survey API via the external gateway.

The request is checked by the Authorization Engine. A new task is created in the survey API, which creates a corresponding RabbitMQ message. This message is consumed by the TSD client. A project specific access token is fetched by the client, the relevant task data is fetched from the API, and a task is created by calling the Helsenorge API. The unique task ID, which maps to the national ID number of the person for the specific form (for the specific project) is included in the link that is presented to the citizen.

The sequence diagram for task completion (encompassing data delivery and reporting) is shown below:

Survey completion



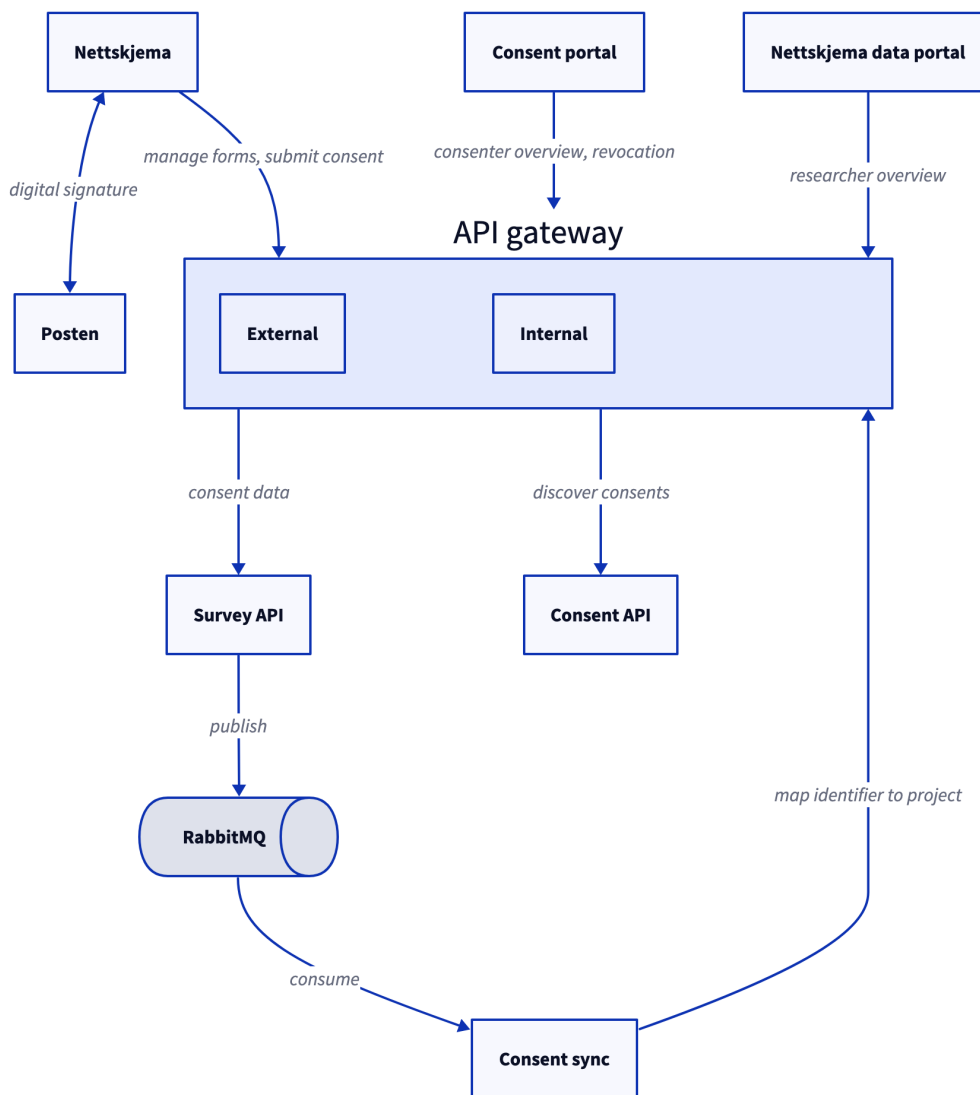
Once the citizen follows the link, in their inbox, to Nettskjema, the unique task ID, and their national ID number is propagated into the submission data of the form (or sequence of forms). Login to Nettskjema, for survey completion, implements SSO with the Helsenorge OIDC provider. This only works for forms which have been configured as Helsenorge forms. Nettskjema proceeds with data delivery as normal, but with an additional check to see whether the task has been defined, whether the national ID is linked to the task, and

whether the task has been completed before. Once data has been delivered, a RabbitMQ message is published by the survey API.

The second part of this process is generating a report for the citizen. The TSD client consumes RabbitMQ messages, checks whether the form has been configured for Helsenorge, and if so, gets the submission ID from the RabbitMQ message. The client fetches an access token, and the form metadata for the specific submission ID. An access control rule in the IAM system allows the client to fetch only form *metadata* for a specific submission ID. This allows the client to discover the task ID. Now the client fetches a new access token, and provides the task ID while doing so. The Token Exchange includes the task ID in a signed claim in the access token. The client then fetches all submission data that contains the task ID in their metadata. An access control rule in the IAM system allows doing this only if the client references a specific task ID in its URI search query, and only authorizes the request if that value matches the value included in the signed claims of the access token. Having found all task specific data, a pdf is generated and sent to Helsenorge.

Dynamic digital consent

The dynamic digital consent system leverages Nettskjema to enable researchers to collect digitally signed consent from study participants. It also enables these study participants to see an overview of their various consents, and provides the ability to revoke such consent. The consent system consists of the following services:



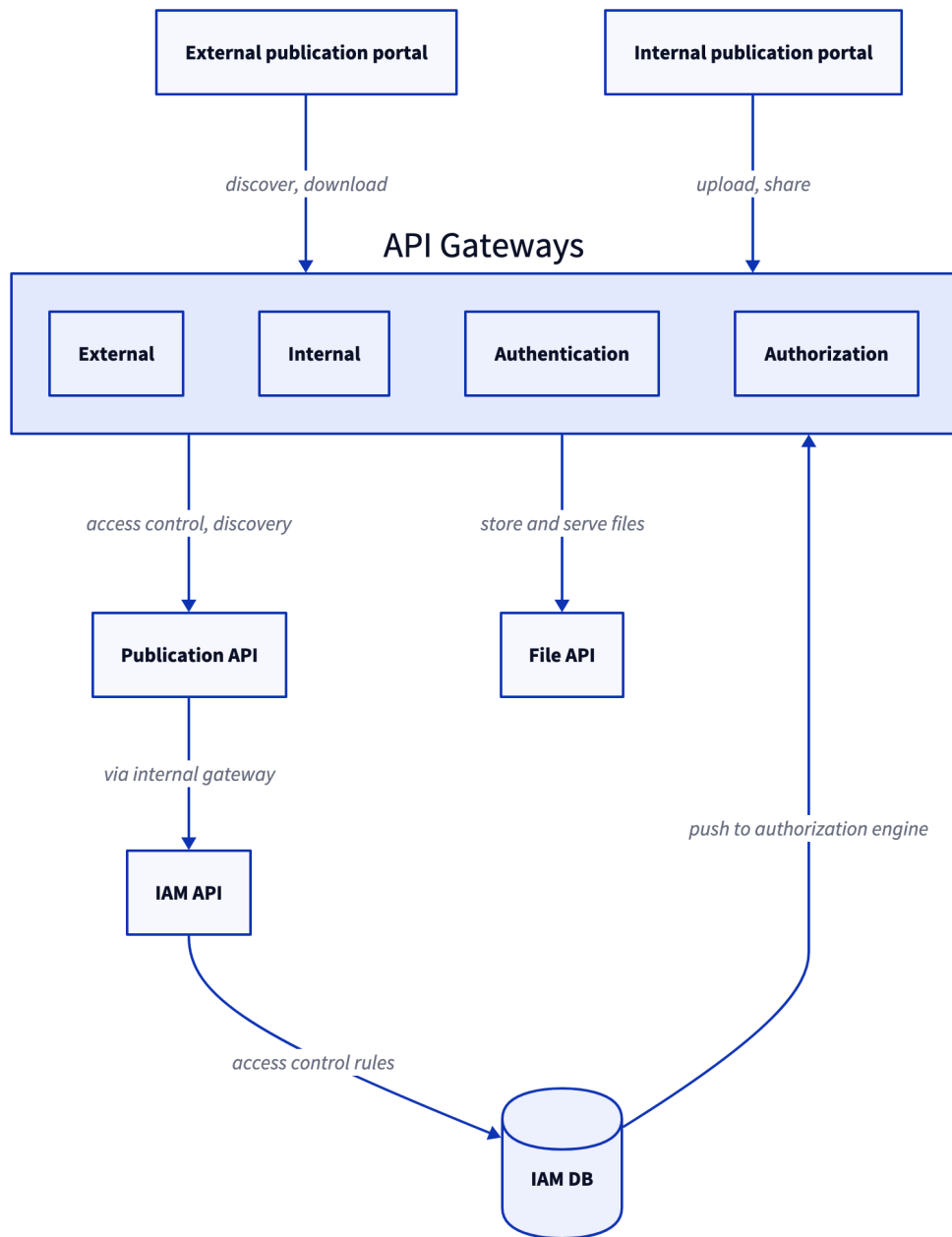
Researchers use Nettskjema to define consent forms. Such forms can be configured to require digital signature, implemented by integrating with Posten. Study participants submit their consents as normal Nettskjema responses, which are stored in the survey API. The survey API publishes a RabbitMQ message, which is picked up by the consent sync. The sync, in turn, calls the consent API, creating a mapping between the respondent's ID, and the TSD project where the form is located.

Study participants log in to the consent portal (TSD (2025h)) with ID-porten. The portal calls the consent API to discover which projects and which forms have consent data submitted by the logged in participant. If the participant wanted to revoke their consent, the consent portal would redirect them to Nettskjema.

Researchers have an overview of *all* consents in the Nettskjema data portal, available inside TSD.

Publication

The publication system allows researchers to share files located in their TSD project with specific persons. The system consists of the following services:



Researchers use the internal publication portal, inside TSD, to upload files to the File API's publication backend. Any such file can then be shared with associated members, or groups of associated members. The

portal calls the publication API in order to manage access control. The publication API is a simplification of the IAM API, which provides simpler semantics for managing lower-level access control rules. It calls the IAM API on behalf of the user, either by re-using the user's access token, or by requesting its own access token when needed. Changes to the IAM database result in access control changes being propagated to the Authorization Engine. Files can be shared with restrictions being placed on: the start and end date for availability, and how many times it can be downloaded.

Associated members log in to the external publication portal (TSD (2025k)) with ID-porten. The portal calls the publication API's discovery endpoint, which allows it to display all files which are shared with the associated member - either directly, or via a group. The files can then be downloaded. All such downloads are logged.

Tables

Tables is a web service that is available inside TSD's private network. It uses OIDC login, and the File API's apps backend. It provides collaborative audited editing on tabular datasets. Users can provide schemas to customise the behaviour and layout of their tables. TSD can set up integrations with other services such as Nettskjema on request.

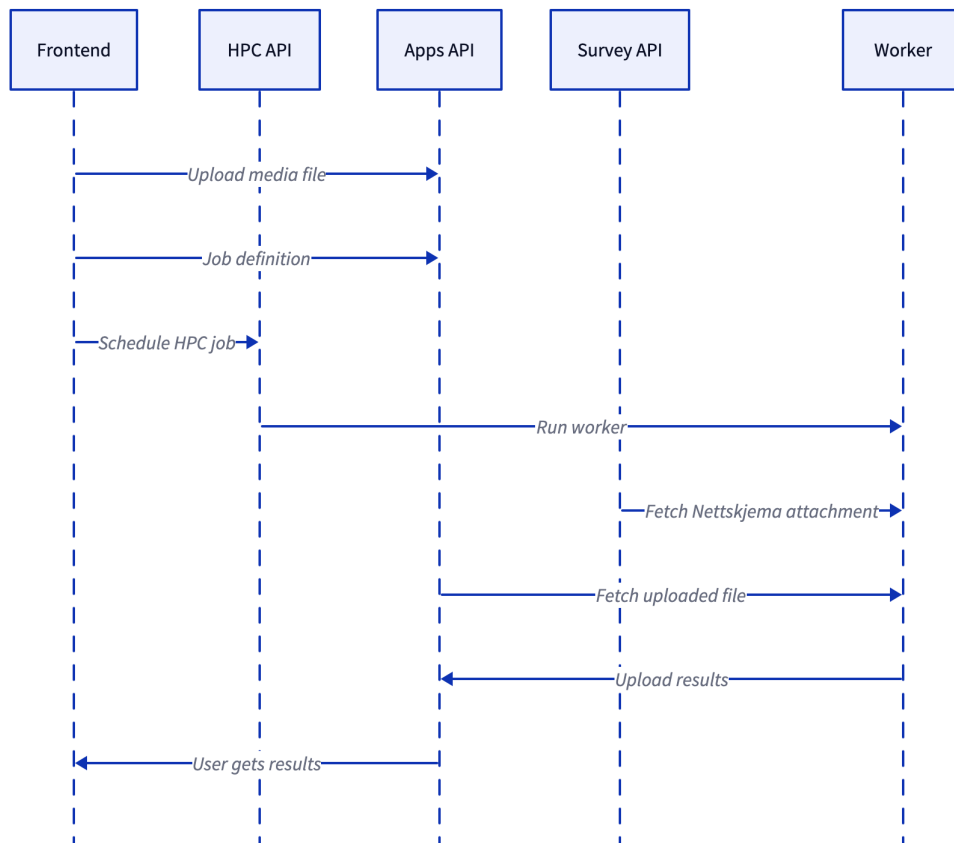
Risk

The system that TSD uses to record its risk assessments is available as a service to all TSD customers too (TSD (2025c)). It is an instance of the Tables service, with specialised table schemas suitable for risk assessment, and runs outside of TSD's private network.

TSD Autotekst

TSD Autotekst is a web service, available inside TSD's private network, that provides AI-powered transcription of media files. It is a frontend application that integrates with TSD APIs via the IAM system. Users login to the frontend with the TSD OIDC provider. The frontend performs token exchange to get the necessary access tokens for interacting with the Apps API, HPC API, and Survey API.

Transcription



Users can choose to upload media files stored in their TSD project, or to use Nettskjema attachments. The results of a transcription job can be either private (visible only to the user themselves) or project-wide (visible to all project members). The frontend defines the transcription job and stores it in the Apps API. The HPC API is called, passing along the set of access tokens needed to fetch the media file. The HPC API schedules a HPC job. When the job is started by Slurm, the worker fetches the media file from the API, transcribes it, and uploads the results to the Apps API. Users can see the job status, and fetch their results from the frontend.

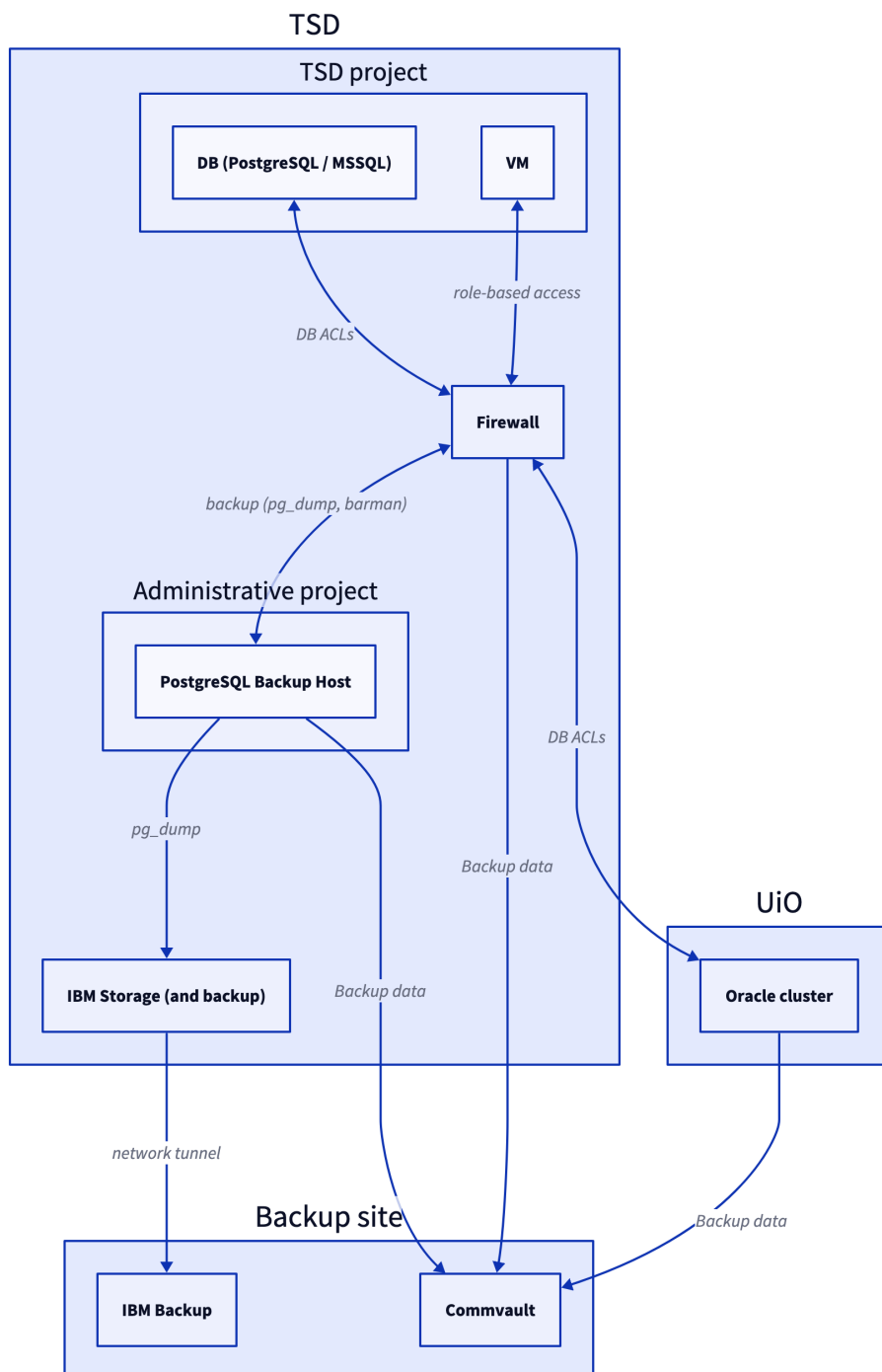
API hosting

TSD projects can run their own REST APIs in their project-specific networks (for example in a container), and expose these APIs for clients running outside TSD. This is done by routing traffic at a specific HTTP endpoint, via TSD's external API gateway to the project's API. Usage of the API requires integration with

TSD's authentication and authorization services. Projects can decide to re-use existing access control rules (that implement default project roles and rights), or to set up their own access control, in consultation with TSD staff.

Databases

TSD offers managed database servers for PostgreSQL ("PostgreSQL" (2025)), MSSQL ("Microsoft SQL Server" (2025)), and Oracle ("Oracle DB" (2025)).



PostgreSQL database clusters run on project-specific virtual machines and give projects the ability to create multiple databases. PostgreSQL clusters have database-specific IP-based access control rules. All traffic to the databases are routed through the TSD firewall. PostgreSQL databases have two backup systems: daily snapshots, and point-in-time transactional backups. The former is done by running the `pg_dump` facility from the central PostgreSQL backup host. These backup files are stored in project-specific directories, that are then backed up to IBM storage. The latter runs `barman` (“Barman” (2025)) from the same backup host. The `barman` backups are stored on low-latency VM block storage. The VM itself is then backed up to Commvault, hosted in the UiO backup site.

MSSQL database clusters run on project-specific virtual machines and give projects the ability to create multiple databases. TSD uses an Oracle cluster that runs in the UiO domain in order to minimise license costs. Oracle databases run on a dedicated virtualisation cluster, in a dedicated network segment. TSD project VMs gain access to MSSQL and Oracle via the TSD firewall, and via IP-based access control on the database level. Transaction logs are backed up to Commvault, hosted at the UiO backup site.

Hardware management

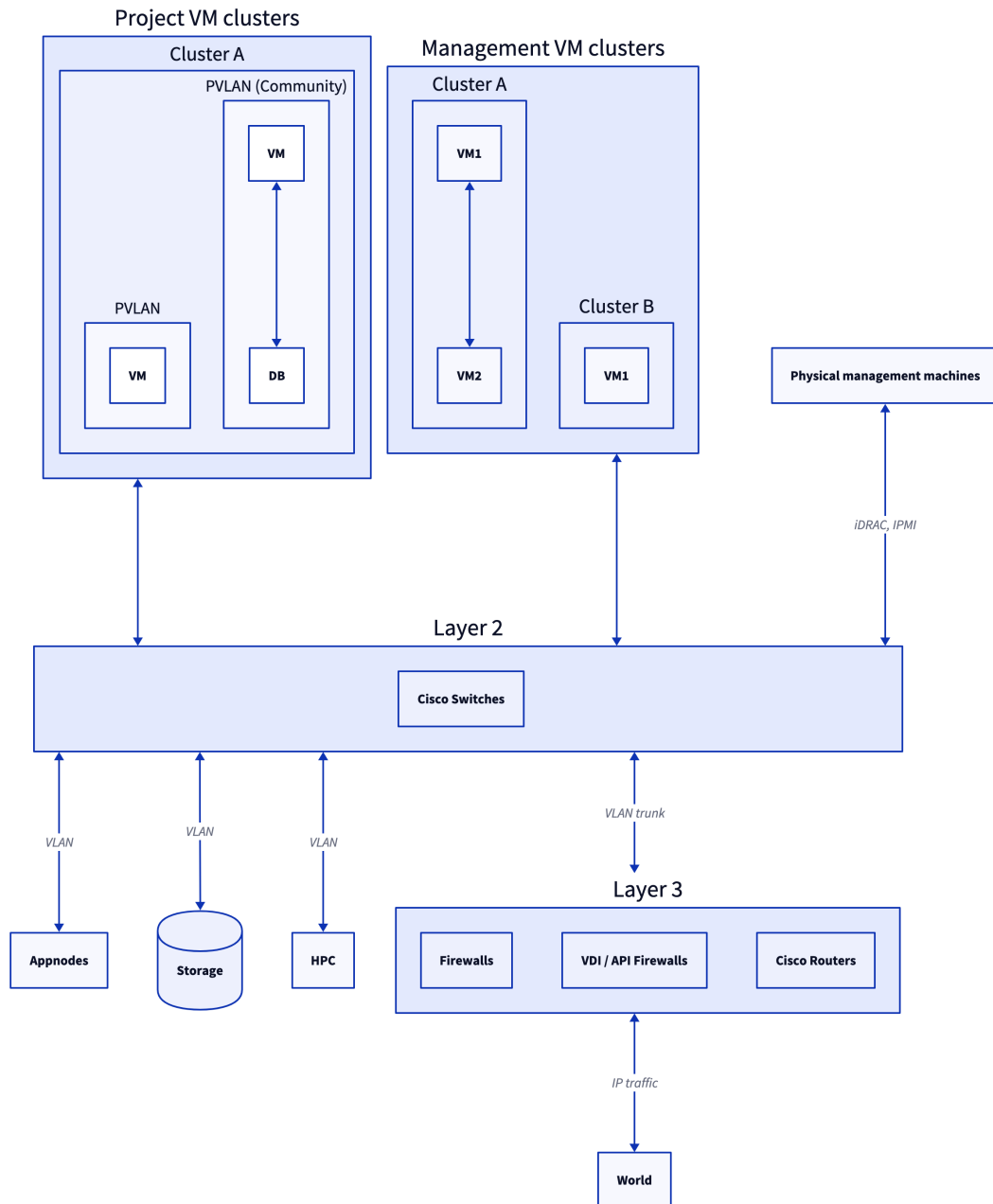
Projects can purchase their own hardware with support from TSD staff, covering the entire life-cycle: procurement, racking, and operating system installation. Such hardware can be made available as standalone servers inside TSD projects, or as compute nodes in the HPC cluster. When installed as a standalone server, projects first log in to their VMs, and then connect to the server via `ssh`. When the hardware is a part of the HPC cluster, projects get priority access via a dedicated Slurm queue. As with VMs, custom software can be requested. It is also possible to use the HPC module software. Backup of local storage is available on request.

4. Infrastructure

This section presents details about how TSD’s datacenter infrastructure functions, and which infrastructure services are used. The section ends with a discussion of project resource life-cycle management, in the context of infrastructure services.

Networking

TSD’s internal network is organised as such, using terminology from the OSI model (“OSI Model” (2025)):



Layer 2 (ethernet) networking inside TSD is handled by a set of Cisco switches, and by the virtual switches of the virtualisation clusters. Layer 3 networking (IP) is handled by: the TSD firewall, the VDI / API firewall, and Cisco routers. TSD uses a rfc1918 (IETF (1996)) private address range for IPv4 internally. All machines have IPv6 addresses. TSD has two sets of virtualisation clusters: one set for the management project, and

another set for customer's projects.

The management project has several VLANs, each dedicated to a specific purpose. For the management clusters, if VMs are located on the same VLAN and virtualisation cluster, network communication between them is sent via the virtual switch, and governed by operating system level firewalls and configuration. Intra-cluster networking, within the same VLAN is sent via the cisco switch in addition to the virtual switches. Intra-VLAN networking is not governed by the TSD firewall.

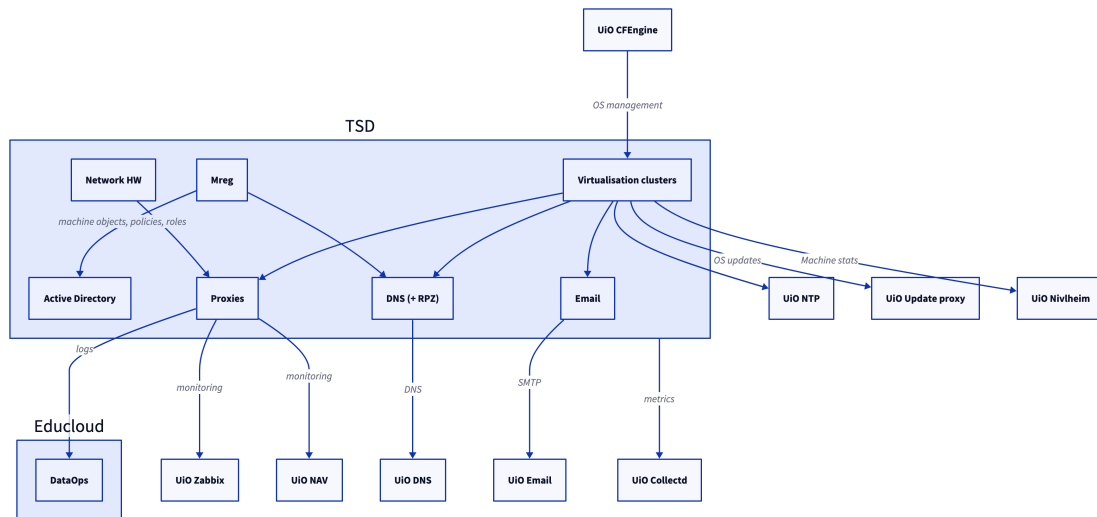
Customers' TSD project VMs are configured to run in an isolated private VLAN ("Cisco Systems' Private VLANs: Scalable Security in a Multi-Client Environment" (2010)) by default. This means that they can only communicate through their IP gateway - which is the TSD firewall, not other VMs on the same private VLAN. The firewall then enables traffic to central services running on the management clusters. Some TSD projects need communication between their VMs (and/or with physical app nodes), such as when projects run databases, container hosts, or HPC submit nodes. These projects are located in community private VLANs where they in addition to traffic through their IP gateway are allowed to communicate with other project VMs on the same community VLAN.

In addition to segmenting traffic inside TSD, the TSD firewall also governs outgoing and incoming traffic to/from services hosted in the UiO domain, and other networks. The TSD API and Horizon login is exposed via the VDI / API gateway, which allows ipv4 networking. Traffic is then routed to the TSD firewall, and further along inside TSD as needed.

Physical management machines, and HPC are not routed via the TSD firewall, with access control being implemented via ACLs Cisco routers.

Infrastructure Services

TSD's infrastructure services are deployed in three different environments: TSD's internal network, UiO's network, and Educloud. The figure below shows which services run where, and the communication between them:



All machines (VMs and physical nodes) are registered in Mreg (UiO (2025a)), the machine registration system which assigns IP addresses. Mreg also stores machine policies and roles - logical (and hierarchical) tags

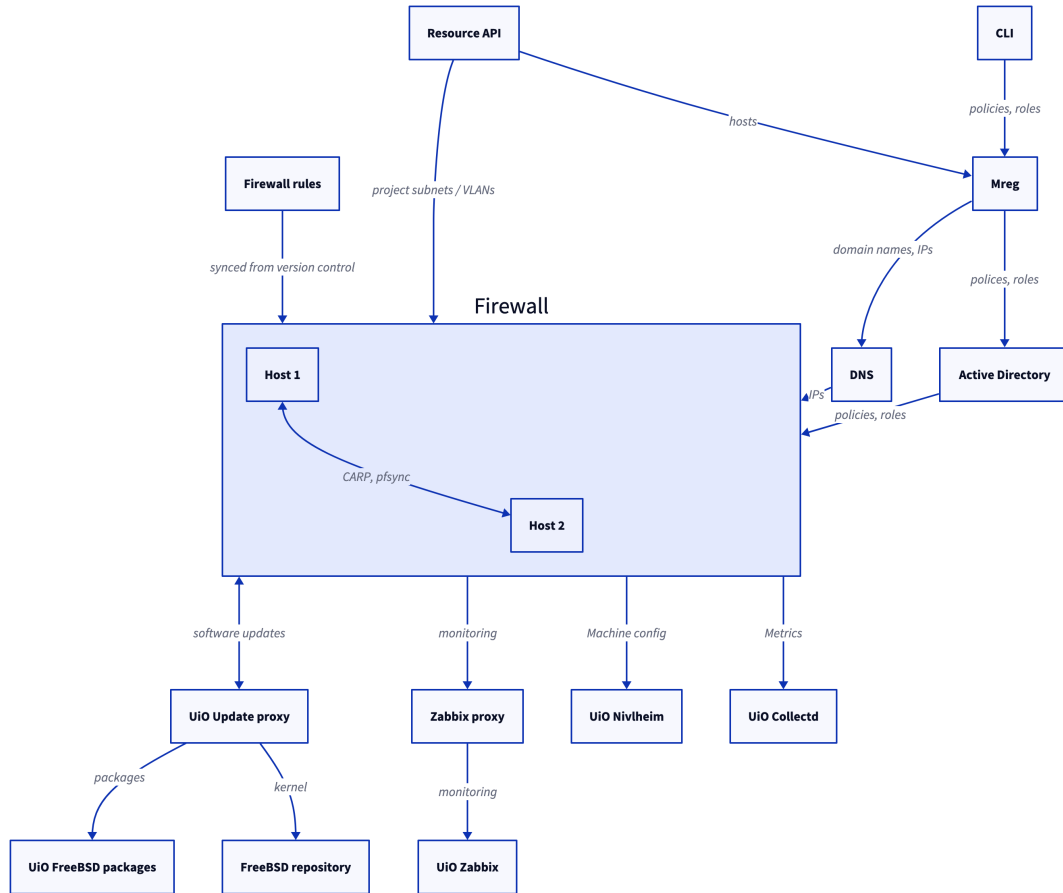
attached to machine objects. Mreg generates zone files for the DNS system, which also runs in TSD. The internal DNS is used by VMs and hardware. TSD's DNS runs with Response Policy Zones (RPZ) ("DNS Response Policy Zones" (2025)), essentially a DNS firewall that allows TSD to control which domain names can be resolved. Domain names registered to machines outside of TSD's network can only be resolved if they are added to a managed white-list. TSD's IPv6 DNS records are shared with UiO's DNS.

All system event logs are sent to the DataOps service that runs in Educloud - a network environment owned and operated by UiO. Staff get access to logs using their operational accounts, subject to access control determined by group membership in the operational IAM system (explained in section 7). System monitoring is handled by the Zabbix service ("ZABBIX" (2025)), running on UiO. Zabbix events are sent there via the Zabbix proxy. Network equipment is monitored using NAV ("NAV (Network Administration Visualized)" (2025)), and similar to monitoring and logging, data is sent to the UiO service via a proxy. TSD has an internal email server which can be used by systems to send event logs, or crash reports via UiO's email system. All machines report configuration data to UiO's Nivlheim (UiO (2025b)).

The virtualisation clusters in TSD use UiO's NTP (Network Time Protocol) servers for clock synchronisation. Linux VMs get their operating system kernel updates and official software packages from official vendors via the UiO update proxy. Linux VMs are centrally managed using CFEngine ("CFEngine" (2025)), running on UiO. Windows VMs are patched in a two-step fashion: an admin host connects to the UiO update proxy to download packages, and VMs connect to this admin server for downloads. All hosts in TSD report live metrics to UiO collectd.

Firewall

TSD's firewall is implemented using FreeBSD's ("FreeBSD" (2025)) operating system firewall, **pf** ("Pf" (2025)). The firewall routes all TSD internal IP traffic (except for two management VLANs), and in- and out-bound traffic. All VMs are configured with the firewall as their default gateway. FreeBSD's **pf** is a stateful firewall that is configured with a set of rules. Rules take the generic form: **{action} {interface} {protocol} from {source-table} to {destination-table} port {port}**, where tables are sets of IP addresses. TSD's automation setup allows for the source- and destination tables to be populated dynamically, enabling agile firewalling capabilities. The overall setup is shown in the figure below:



Rules are kept in a version controlled repository, are subject to code review, and synchronised to the firewall hosts. At any given time, one of the hosts are acting as a primary in a high availability clustering setup. High availability is achieved by using the Common Address Redundancy Protocol (CARP) (“Carp” (2025)) to allow for a floating IP address between the two hosts, and using **pfsync** (“Pfsync” (2025)) to synchronise the **pf** state table across hosts. Together, these protocols allow for zero-downtime failovers in case of operational issues or during upgrades or patching.

The Firewall fetches information about project subnets and VLANs from the resource API. Hosts that are defined in the resource API are added to Mreg, where they are assigned IP addresses and policies and/or roles. DNS is populated from Mreg’s zone files, and policies and roles are synchronised to AD in a specific tree. Firewall tables, referenced in the rules, are populated dynamically from either LDAP lookups (to AD), or from DNS lookups.

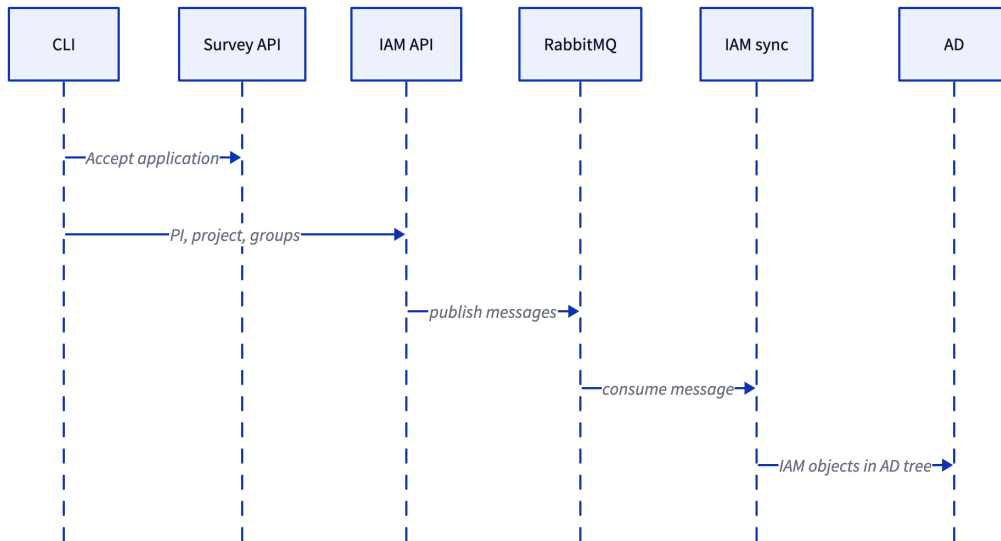
TSD maintains its own FreeBSD package repo (“Poudriere” (2025)), which mirrors official packages and provides custom built packages used for communication with Zabbix and Nivlheim. Access is mediated via the UiO update proxy. Operating system updates and patching are performed with standard FreeBSD tooling, fetching source code from official repositories via the UiO update proxy.

Live monitoring data from the Firewall is sent to UiO Zabbix, and configuration data to UiO Nivlheim.

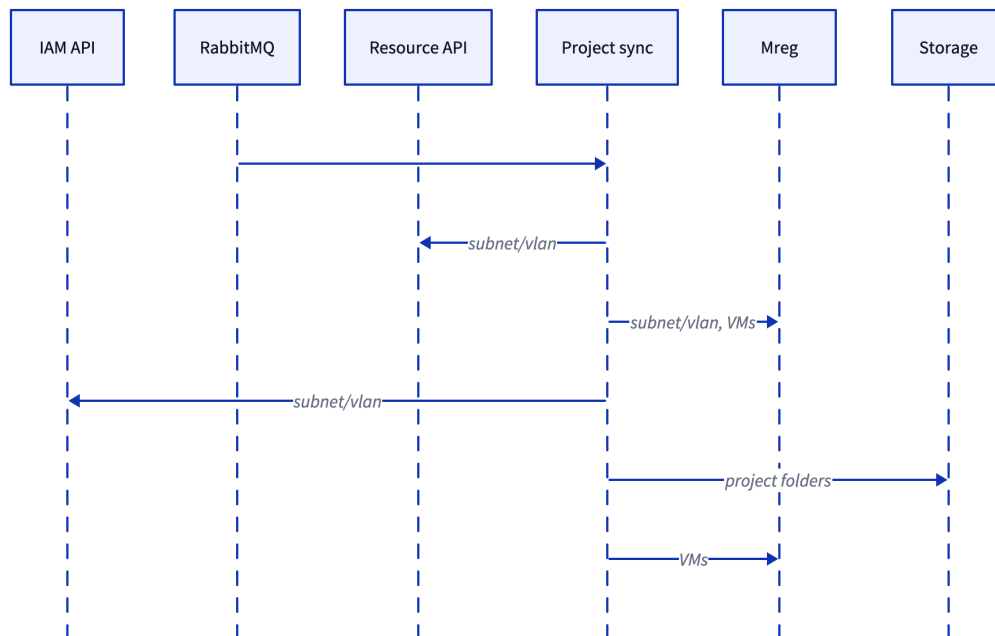
Metrics are reported to UiO collectd.

Project life-cycle management

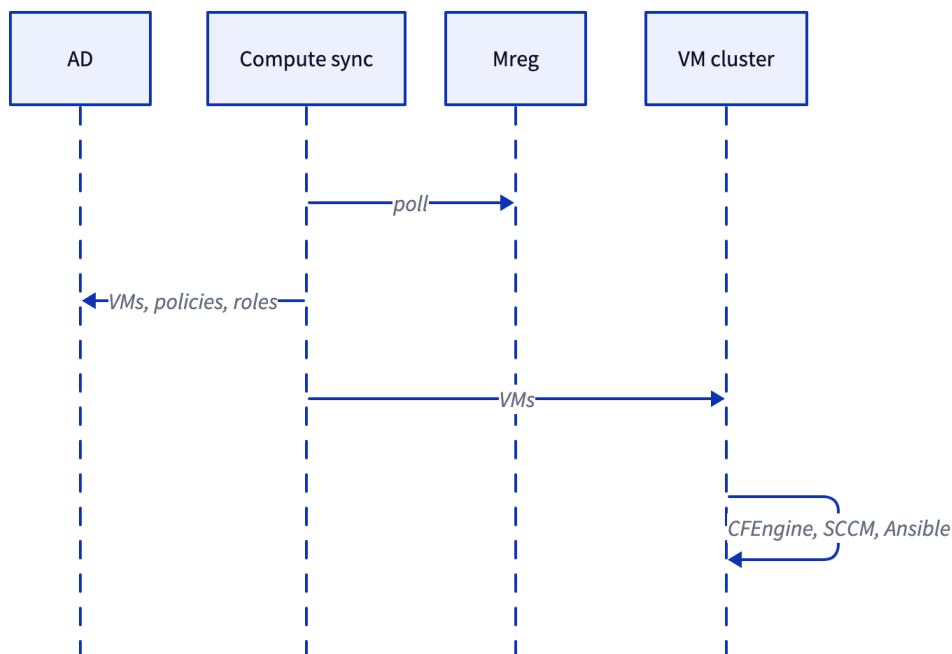
The creation of project-specific resources proceeds as follows:



Applications for new projects are delivered with Nettskjema, and stored in the Survey API. A command-line application is used by TSD staff to review such applications. If accepted, the IAM API is used to create the PI's person object, the project, and standard groups. The IAM synchronisation job propagates these objects to AD.



Next a project synchronisation job picks up the IAM project creation message from RabbitMQ. The project IP subnet and VLAN are chosen, and stored in the IAM- and resource APIs, and Mreg. Next the project's storage area and standard folders are created. User-specific folders are created by the same job at the time of user creation. The VMs that were ordered in the initial application are created in Mreg by the job.



A compute synchronisation job polls Mreg to discover changes in virtual machines. Computer objects are created in AD for the new VMs. After that, VM images (imported manually from UiO) are used with to provision VMs in the virtualisation cluster. Once provisioned, CFEngine, SCCM, and Ansible runs post-install scripts.

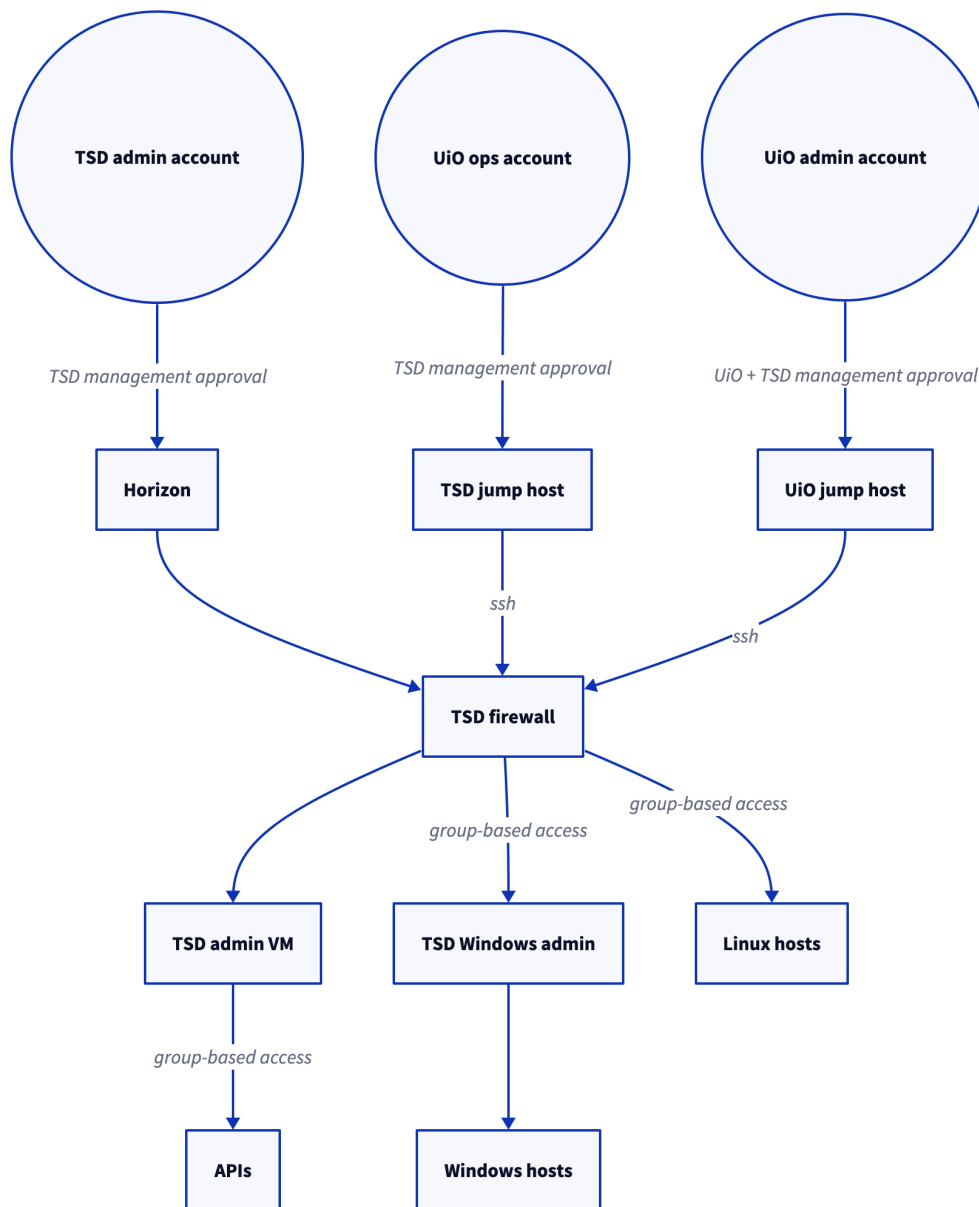
Modifications to storage and compute resources are done per request. Project resources are deleted 90 days after project expiry. User-specific folders are deleted when users are deleted (while a project is still ongoing).

5. Operational environment

There are three types of accounts that can give staff operational access to TSD:

1. TSD administrative accounts, provisioned in TSD's IAM system
2. UiO operational accounts, provisioned in UiO's operational IAM system
3. UiO administrative accounts, provisioned locally on UiO's disaster recovery infrastructure

The method of access relevant to each account is illustrated in the figure below:



TSD admin accounts allow staff members to log in to VMs using Horizon remote login. Support staff use this solution to gain access to a VM where they can perform administrative tasks related to user support. This enables them to get information about users while helping them, and to perform tasks such as creating groups on behalf of users. Windows administration is also done using TSD admin accounts. Windows administrators are members of a specific IAM group, which grants them access to all Windows VMs and AD.

UiO staff can apply for UiO operational accounts. Such accounts can give access to specific operational systems, based on access control policies. TSD has a linux jump host in the UiO domain which where approved UiO operational accounts can log in via `ssh` (Secure Shell) using Two-factor Authentication. Depending on their group memberships, they can then log in to one or more Linux machines in TSD.

Some UiO staff have access to UiO administrative accounts, for use in UiO's disaster recovery infrastructure. This infrastructure has a separate local IAM system, and is designed to be usable when many other system are not operational. Such account allow staff to log in via `ssh` (Secure Shell) using Two-factor Authentication to a jump host. This jump host has access to all Linux machines in TSD.

References

- "Apptainer." 2025. <https://apptainer.org/>.
- "Bandersnatch." 2025. <https://pypi.org/project/bandersnatch/>.
- "Barman." 2025. <https://pgbarman.org/>.
- Broadcom. 2025. "RabbitMQ." <https://www.rabbitmq.com/>.
- "Carp." 2025. <https://man.freebsd.org/cgi/man.cgi?query=carp&sektion=4>.
- "CFEngine." 2025. <https://cfengine.com/>.
- "Cisco Systems' Private VLANs: Scalable Security in a Multi-Client Environment." 2010. <https://datatracker.ietf.org/doc/html/>
- Digdir. 2025. "ID-Porten." <https://docs.digdir.no/docs/idporten/>.
- "DNS Response Policy Zones." 2025. <https://dnssrpz.info/>.
- "Dragen." 2025. <https://www.illumina.com/products/by-type/informatics-products/dragen-secondary-analysis.html>.
- "EasyBuild." 2025. <https://easybuild.io/>.
- "FreeBSD." 2025. <https://www.freebsd.org/>.
- "Helsenorge." 2025. <https://helsenorge.no>.
- "IBM Storage Protec." 2025. <https://www.ibm.com/products/storage-protect>.
- "IBM Storage Scale." 2025. <https://www.ibm.com/products/storage-scale>.
- IETF. 1996. "Address Allocation for Private Internets." <https://datatracker.ietf.org/doc/html/rfc1918>.
- . 2005. "HOTP: An HMAC-Based One-Time Password Algorithm." <https://www.ietf.org/rfc/rfc4226.txt>.
- . 2011. "TOTP: Time-Based One-Time Password Algorithm." <https://datatracker.ietf.org/doc/html/rfc6238>.
- . 2015. "JSON Web Token." <https://datatracker.ietf.org/doc/html/rfc7519>.
- "Libsodium." 2025. <https://libsodium.gitbook.io/doc>.
- "Microsoft SQL Server." 2025. <https://www.microsoft.com/en-us/sql-server>.
- "NAV (Network Administration Visualized)." 2025. <https://nav.uninett.no/>.
- "Norsk Helsenett." 2025. <https://nhn.no>.
- "OIDC Code Flow." 2025. https://openid.net/specs/openid-connect-core-1_0.html#CodeFlowSteps.
- OpenID-Foundation. 2025. "OpenID Connect." <https://openid.net/developers/specs/>.
- "Oracle DB." 2025. <https://www.oracle.com/database/>.
- "OSI Model." 2025. https://en.wikipedia.org/wiki/OSI_model.
- "Pf." 2025. [https://man.freebsd.org/cgi/man.cgi?pf\(4\)](https://man.freebsd.org/cgi/man.cgi?pf(4)).
- "Pfsync." 2025. <https://man.freebsd.org/cgi/man.cgi?query=pfsync&sektion=4>.
- "PKCE." 2025. <https://datatracker.ietf.org/doc/html/rfc7636>.
- "Podman." 2025. <https://podman.io/>.
- "PostgreSQL." 2025. <https://postgresql.org/>.
- "Poudriere." 2025. <https://man.freebsd.org/cgi/man.cgi?poudriere>.
- "Rocky Linux." 2025. <https://rockylinux.org/>.
- "S3cmd." 2025. <https://s3tools.org/s3cmd>.
- Sigma2. 2025a. "Advanced User Support." <https://www.sigma2.no/service/advanced-user-support>.
- . 2025b. "Sigma2." <https://www.sigma2.no>.
- "Slurm." 2025. <https://slurm.schedmd.com>.
- "Slurm REST API." 2025. https://slurm.schedmd.com/rest_api.html.

“The Kerberos Network Authentication Service (V5).” 1993. <https://datatracker.ietf.org/doc/html/rfc1510>.

TSD. 2025a. “Pg-Iam.” <https://github.com/unioslo/pg-iam>.

———. 2025b. “Pysquril.” <https://github.com/unioslo/pysquril>.

———. 2025c. “Risk.” <https://risk.uio.no>.

———. 2025d. “TSD API Client.” <https://github.com/unioslo/tsd-api-client>.

———. 2025e. “TSD Help.” <https://www.uio.no/tjenester/it/forskning/sensitiv/kontakt/>.

———. 2025f. “TSD OIDC.” <https://github.com/unioslo/tsd-api-docs/blob/master/integration/tsd-oidc.md>.

———. 2025g. “TSD Selfservice.” <https://selfservice.tsd.usit.no>.

———. 2025h. “Tsd-Consent-Portal.” <https://consent.tsd.usit.no>.

———. 2025i. “Tsd-Data-Portal.” <https://data.tsd.usit.no>.

———. 2025j. “Tsd-File-API.” <https://github.com/unioslo/tsd-file-api>.

———. 2025k. “Tsd-Publication-Portal.” <https://publication.tsd.usit.no>.

———. 2025l. “Tsd-S3cmd.” <https://github.com/unioslo/tsd-s3cmd>.

UiO. 2025a. “Mreg.” <https://github.com/unioslo/mreg>.

———. 2025b. “Nivlheim.” <https://github.com/unioslo/nivlheim>.

“ZABBIX.” 2025. <https://www.zabbix.com/>.



UNIVERSITETET
I OSLO

