SLATE Access Control Policy and Procedures

Version	Comment	Effective date
1.0	In draft	Mar 11, 2022

Purpose 2

On- and Off-Boarding Platform Administrators 3

SLATE Platform Components 3

- Repositories and Registries 3
  - SLATE Platform and Helm Chart Repository 3
    - GitHub 3
  - Container Image Registries 4
    - DockerHub 4
    - OSG Harbor 4
    - GitHub Container Repository 5
- Core Services 6
  - Cloud Core Services 6
    - Amazon Web Services (AWS) 6
      - Identity and Access Management: AWS credentials for management 6
      - Database: DynamoDB 6
      - DNS: Route53 6
      - Storage: S3 7
  - On-premise Core Services 7
    - SLATE API 7
      - SLATE API Service 7
      - SLATE API server 7
    - Key distribution 8
      - Key repository 8
      - Configuration and automation: 8

This document is a policy of the SLATE (Services Layer at the Edge) project, supported by the National Science Foundation Office of Advanced Cyberinfrastructure: "CIF21 DIBBs: EI: SLATE and the Mobility of Capability", award number OAC-1724821.

Monitoring stack 9
Logging and Visualization 9
Alerting and Passive Monitoring 9
Checkmk 9
Active Monitoring 10
perfSONAR 10
PerfSONAR Web Admin (PWA): 10
Development Environment 10
SLATE Development Virtual Machines 11
Continuous Integration/Continuous Deployment (CI/CD) 11
Jenkins CI/CD service 11
GitHub Actions 11
Individual Developer Tools and Prototype Tools 12
MiniSLATE 12
SLATE-lite 12
SLATE User Facing Services 12
Web presence 12
Website 12
SLATE User Portal 13
SLATE Command Line Interface (CLI) 13
SLATE Sandbox 13
SLATE Communication 14
Email Lists 14
Real-time chat 14
Video content 14
Shared presentation, tutorial and file repository 14
Overleaf 15

Purpose

SLATE team members require various levels of access to central components of SLATE. This document describes the location of each component and how different member roles obtain access. The Overview of SLATE Platform Internals and Security architectural document[1] describes the relationships and interactions of the various components.

[1] Overview of SLATE Platform Internals and Security architectural document: https://docs.google.com/document/d/11IzFGkoTis4KOLmwThpWSHZ5NFEhau1T/edit?rtpof=true

On- and Off-Boarding Platform Administrators

The SLATE project provides on-boarding documentation to acquaint new team members with essential services, systems and processes. During this period new team members are granted access to systems with appropriate role-based permissions. For team members departing the project, there is an off-boarding process to revoke all privileges. For some individuals leaving the project, continued access may be granted for future collaboration and project continuity purposes. The SLATE project will perform a periodic review of team member access with the SLATE principal investigators (PIs).

The on-boarding process requires Multi-Factor Authentication (MFA). The SLATE team configures MFA for each service and component that can support it. For service accounts which do not support MFA, the SLATE team makes other provisions, such as tokens.

SLATE Platform Components

Repositories and Registries

The SLATE project utilizes several repositories and registries for hosting the codebase of the SLATE Platform and for applications which SLATE (co-)manages or references. Each repository and registry accommodates different aspects.

SLATE Platform and Helm Chart Repository

GitHub

The SLATE Platform and Helm chart repository resides in GitHub[2]. The SLATE principal investigators and senior SLATE team members (with coordination of the SLATE PIs) grant access to the repository for creating, editing, and modifying the SLATE Platform components. The access to the repository also allows the modification of the SLATE Helm charts for applications.

- *Where maintained:* The SLATE git repository for the software and applications resides at the public SLATE GitHub.
- *Granting/Revoking access:* The SLATE Project PIs or SLATE Operational team members designated as GitHub repository owners grant/revoke access based on review of need of collaboration level with the SLATE team. Senior SLATE Operational team members consult with SLATE PIs prior to adding members.
- *Credentials:*
    - *Password:* The SLATE GitHub repository requires individual password authentication to access the site.

[2] SLATE Platform GitHub instance: https://github.com/slateci

- *Multi Factor Authentication:* The SLATE GitHub repository requires multi factor authentication as defined/allowed by GitHub[3].
- *Token access:* The SLATE CI/CD infrastructure utilizes token access for automation of the CI/CD pipeline.

Container Image Registries

The SLATE project uses multiple container registries for container images. SLATE team members and the SLATE CI/CD infrastructure both access each of these resources in different ways, as explained below.. The three main image registries are the following:

- DockerHub

  DockerHub provides the primary mechanism for accessing general community images and base OS container images. Several science applications also maintain their container images on Docker. SLATE has maintained the bulk of its images on Docker as well.
  - *Where maintained:* SLATE uses the public DockerHub[4] for access to some container images.
  - *Granting/Revoking access:* The SLATE PIs or SLATE Operational team owners review and revoke authentication and authorization for write privileges of images maintained by the SLATE team. Read access and pull access is public.
  - *Credentials:*
    - *Password/Token Access:* The DockerHub registry requires individual password authentication to access the site for putting up containers. The DockerHub registry does not require passwords for reading.
    - *Multi Factor Authentication:* The DockerHub registry currently does not require multi factor authentication.
    - *Token access:* The SLATE CI/CD infrastructure uses token access with a SLATE service account to push/pull images.

- OSG Harbor

  OSG Harbor is an image container repository which focuses on distribution of scientific containers, such as those maintained by the Open Science Grid community. SLATE is migrating many of its images there too.
  - *Where maintained:* SLATE uses the OSG Harbor repository[5] for science-based applications.
  - *Granting/Revoking access:* The SLATE PIs or SLATE Operational team owners review and revoke authentication and authorization for write privileges of images maintained by the SLATE team. Upon review with the SLATE PIs, OSG sets up

[3] GitHub 2 Factor Authentication: https://docs.github.com/en/authentication/securing-your-account-with-two-factor-authentication-2fa/configuring-two-factor-authentication
[4] Dockerhub: https://hub.docker.com/
[5] OSG Harbor repository: https://hub.opensciencegrid.org/

one of the following roles for the individuals: maintainer, developer, user, guest, limited guest, or project. Read access and pull access is public.

- *Credentials:*
  - *Password/Multi Factor Authentication:* SLATE team members access the OSG Harbor registry using federated authentication via CILogon, using their respective home institutions' credentials. This mechanism allows the user access to the site for putting up containers for use with the SLATE catalog. The OSG Harbor registry does not require authentication for reading the containers. SLATE team members' home institutions require MFA on all federated logins.
  - *Token access:* The SLATE CI/CD infrastructure uses token access associated with a SLATE service account in OSG Harbor to push/pull images

- GitHub Container Repository

SLATE has used the GitHub Container Repository for a handful of images in support of the SLATE infrastructure.

- *Where maintained:* SLATE uses the GitHub Container Registry[6] located as a service of GitHub.
- *Granting/Revoking access:* Upon consultation with the SLATE PIs, the SLATE team reviews and revokes authentication and authorization for write privileges of maintained images. These privileges tie directly to the permissions of the GitHub slateci user in the slateci organization. Read access and pull access is public. Any repository in the slateci organization has read/write capabilities to the repositories.
- *Credentials:*
  - *Password:* SLATE team members access the GitHub Container Registry with GitHub accounts. These only require password authentication to access the site for writing containers. The GitHub Container Registry does not require passwords for reading containers.
  - *Multi Factor Authentication:* The GitHub Container Registry requires multi factor authentication.
  - *Token access:* The SLATE CI/CD infrastructure uses token access associated with a SLATE service account in the GitHub Container Registry to push/pull images.

[6] GitHub Container Registry: https://ghcr.io

Core Services

Cloud Core Services

The SLATE project currently uses Amazon Web Services (AWS)[7] cloud services for certain central services.

Amazon Web Services (AWS)

- **Identity and Access Management:** AWS credentials for management
    - *Billing:* The SLATE main PI at the University of Chicago has responsibility for all billing.
    - *Grant/Revoking Access:* The SLATE main PI and senior University of Chicago SLATE personnel review and grant access to AWS services as needed by the project.
    - *Credentials:*
        - *Password:* SLATE team members login to the AWS Identity and Access Management (IAM) service using an IAM account requiring password authentication to access the site.
        - *Multi Factor Authentication:* The AWS Identity and Access Management requires multi factor authentication.
        - *Token access:* The AWS Identity and Access Management provides token access for some of the AWS services. The University of Chicago SLATE PI and Operational staff review and setup token access for the AWS services based on need.
    -
- **Database:** DynamoDB
    - *Purpose:* The DynamoDB retains all data for the SLATE API server.
    - *Grant/Revoking Access:* The SLATE main PI and senior University of Chicago SLATE personnel review and grant access to AWS services as needed by the project.
    - *Token access:* The University of Chicago SLATE PI and Operational staff review and setup token access for the DynamoDB server based on need.
    -
- **DNS:** Route53
    - *Purpose:* SLATE uses the AWS Route53 DNS service for inserting slateci.io DNS records, both forward and reverse. SLATE also uses Route53 for other DNS informational entries
    - *Grant/Revoking Access:* The SLATE main PI and senior University of Chicago SLATE personnel review and grant access to AWS services as needed by the project.

[7] Amazon Web Services (AWS): https://aws.amazon.com/

    - *Token access:* SLATE scripts utilize AWS tokens for Route53 to automate DNS entries. The University of Chicago SLATE PI and Operational staff review and setup token access for the Route53 service based on need.
- **Storage:** S3
    - *Purpose:* SLATE uses AWS S3 storage to store backups for DynamoDB which hosts the SLATE API database. The backups happen once a week.
    - *Grant/Revoking Access:* The SLATE main PI and senior University of Chicago SLATE personnel review and grant access to AWS services as needed by the project.
    - *Token access:* SLATE stores the S3 credential on the API server for use with the backup process. The University of Chicago SLATE PI and Operational staff review and setup token access for the S3 service based on need.

On-premise Core Services

SLATE API

SLATE API Service

The SLATE API service, which runs on the SLATE API server, manages the central database, and federation information. The SLATE API service also provides the central interface for the SLATE Client, portal and other tools which manage the deployment of applications across the federated environment. The SLATE API service also provides the policy interface for user / group role access. See Section 7 of the "Overview of SLATE Platform Internals and Security" document for more information.

- *Where maintained:* The SLATE API service resides on the dedicated SLATE API server virtual machine at the University of Chicago:
- *Granting / RevokingAccess:* For a cluster administrator to access the SLATE API server, the administrator obtains a SLATE token. The cluster must be part of a SLATE group registered through the SLATE portal. Groups are part of existing Virtual Organizations or are a stand-alone Resource Provider site dedicated to providing resources for Virtual Organizations.
    - Dynamo DB credentials allow extraction of all data though most data encrypted by key on SLATE API server. These credentials require sudo or root access
- *Token access:* SLATE cluster administrators obtain a SLATE token by following the process described in "Obtain a SLATE token"[8]. This token also gives access to the SLATE REST API.
    - A root token/superuser token exists that can access any API object.

SLATE API server

- *Where maintained:* The SLATE API server resides on a virtual machine at the University of Chicago.

[8] Obtain a SLATE token: https://slateci.io/docs/cluster/manual/slate-token.html

- *Granting / RevokingAccess:* The SLATE API server has limited ssh access to the server virtual machine granted by the University of Chicago SysAdmin team with permission by the SLATE Principal Investigator. The SysAdmin team manages ssh access via key access. *(see key distribution)*

Key distribution

- Key repository

  SLATE maintains a key repository for user access to SLATE project servers.
  - *Where maintained:* The Mid-West Tier2 group manages a GitLab server on a virtual machine at the University of Chicago. This GitLab server hosts a SLATE project instance to which SLATE project members have access. This project instance stores the user keys for the SLATE Puppet instance.
  - *Granting/Revoking Access:* Mid-West Tier2 SysAdmins manage this GitLab server. They are able to add/remove SLATE members to the SLATE project with per-user keys to the SLATE project upon request by the SLATE PIs or senior SLATE personnel.

- **Configuration and automation:**

  SLATE maintains a Puppet server for configuration and automation. The SLATE Puppet server distributes keys for users to the API server and for access to the development virtual machines and physical machines.
  - *Where maintained:* The SLATE Puppet server resides on a virtual machine at the University of Chicago.
  - *Granting/Revoking Access:* The University of Chicago MANIAC SysAdmin team manages the SLATE Puppet server and adds/removes individual people to Puppet with per-user keys upon request via SLATE PI or senior SLATE personnel.

  SLATE uses Ansible[9] and Kubespray[10] for automated deployment of Kubernetes. This deployment technique[11] happens at the site and does not require a specific server.
  - *Where maintained:* The automated deployment happens at the local site by the local SLATE cluster administrator.
  - *Granting/Revoking Access:* The local SLATE Cluster administrator provides all access to the cluster
  - *Token access:* As part of the installation, the local SLATE Cluster administrator will register the new STATE cluster by obtaining a SLATE token for federation.

[9] Ansible: https://github.com/ansible/ansible
[10] Kubespray: https://kubespray.io/#/
[11] SLATE Kubernetes automation: https://slateci.io/docs/cluster/automated/introduction.html

Monitoring stack

The SLATE Monitoring stack comprises logging, alerting, passive monitoring, and active monitoring for each of the SLATE project clusters. Other SLATE clusters can request the addition of their clusters to this central monitoring.

Logging and Visualization

SLATE uses Elasticsearch/Logstash/Kibana (ELK) stack[12] for its central logging/visualization**.** For local logging of a site

- *Where maintained:* The SLATE ELK stack resides at the University of Chicago as part of the WLCG monitoring stack.
- *Granting/Revoking Access:* For viewing, a web portal exists that allows some anonymous views. For modifications, members of the WLCG monitoring group make changes or add/revoke additional participants in consultation with the SLATE PIs.
- *Credentials:*
    - *Password:* The ELK stack currently uses only passwords for authentication.

Alerting and Passive Monitoring

SLATE uses Checkmk[13] for its Alerting and Passive Monitoring. Checkmk relies on a combination of probes and SNMP polling for its alerting and passive monitoring. The SLATE project sites each run a local instance of Checkmk which monitors locally and reports to a central Checkmk instance at the University of Michigan.

Checkmk

- *Where maintained:* The SLATE Checkmk central service is at University of Michigan, with local instances at the University of Utah and University of Chicago.
- *Granting/Revoking Access:* University of Michigan system administrators grant/revoke access for modifications to the central instance of checkmk. University of Michigan system admins work with University of Utah and University of Chicago system administrators to grant/revoke access to the local instances respectively.
- *Credentials:*
    - *Password:* Each Checkmk site maintains individual local usernames for admin and operator roles. The respective users only have access to the Checkmk service and not the underlying server.

[12] Elastic ELK stack: https://www.elastic.co/what-is/elk-stack
[13] Checkmk: https://checkmk.com/

Active Monitoring

SLATE uses perfSONAR[14] for its Active Monitoring. The perfSONAR monitoring provides feedback on bandwidth and latency capabilities. Each new SLATE site has the ability to run a perfSONAR testpoint to check its capabilities with respect to the main SLATE project clusters.

perfSONAR

- *Where maintained:* The University of Chicago, University of Michigan, and University of Utah each maintain a separate full instance of the perfSONAR toolkit dedicated to SLATE.
- *Granting/Revoking Access:* The system administrators of each respective site grant/revoke access to the underlying perfSONAR instance host OS and associated web portal. The perfSONAR toolkit allows third party usage of the tools under constrained circumstances. For additional requirements, respective system administrators may make adjustments to handle active testing.
- *Credentials:*
    - *Password:* Each perfSONAR site has both web password credentials and an OS set of credentials. These credentials are unique to each site.

PerfSONAR Web Admin (PWA):

The perfSONAR Web Admin software enables the management of multiple sets of tests to respective perfSONAR testpoints.

- *Where maintained:* The perfSONAR Web Admin interface runs as a virtual machine at Michigan State University..The WLCG monitoring group manages the virtual machine and service configuration.
- *Granting/Revoking Access:* For modifications, members of the WLCG Throughput Working Group adds/removes users from the perfSONAR Web Admin web portal interface in collaboration with the SLATE PIs. The service supports fine-grained authorization and individual accounts can be given management access for hosts, host groups, test parameters and mesh configurations. SLATE personnel do not require access to the underlying host.
- *Credentials:*
    - *Password:* The perfSONAR Web Administrator portal requires password authentication, although it also supports x.509 credentials, and, if configured, third party authorization (Google, GitHub).

Development Environment

The SLATE team maintains a development environment for the SLATE API. This development environment also supports scale testing of applications in the transition from incubator to production catalog.

[14] perfSONAR: https://www.perfsonar.net/

SLATE Development Virtual Machines

- *Where maintained:* The SLATE Development clusters reside at the University of Utah. In addition, developers use local mini-slate instances on their local individual machines.
- *Granting/Revoking access:* The Puppet system instantiates people to have access at the SLATE Development clusters for the team. The full SLATE team has ssh and kubectl access to these clusters for trying out deployments and for testing versions of the SLATE client and SLATE server.
- *Credentials:*
    - *Password:* The SLATE Development clusters require key-based ssh for access by SLATE team members.

Continuous Integration/Continuous Deployment (CI/CD)

Jenkins CI/CD service

The SLATE project uses Jenkins for its Continuous Integration. The server resides behind multiple levels of security controls with limited developer access. Please see Section 3 of "Overview of SLATE Platform Internals and Security"

- *Where maintained:* The SLATE Jenkins server resides at the University of Chicago on its virtual machine farm.
- *Granting/Revoking access to the Jenkins server:* Upon approval by SLATE PIs, the University of Chicago SysAdmin team adds access to the members of the development team. These members jump through a gateway box to access the virtual machine. The University of Chicago SysAdmin team manages the gateway box in similar fashion. The Jenkins web interface does not have access to the internet. SLATE team members who have access utilize an ssh tunnel with an ssh key.
- *Credentials for the Jenkins service:*
    - *Password:* Jenkins uses a password-protected service account, currently one assigned to a SLATE team member, to push images to the Dockerhub repository.

GitHub Actions

The SLATE project uses GitHub Actions to provide a "GitOps" look and feel to development/deployment. The SLATE team uses GitHub Actions processes for both SLATE infrastructure and for some SLATE catalog applications.

- *Where maintained:* GitHub Actions are part of the slateci GitHub repository.
- *Granting/Revoking access:* GitHub Actions utilizes the same processes as the GitHub repository for granting/revoking access. For SLATE infrastructure builds, GitHub Actions maps to a single designated user. The account should be a service account, i.e., used only for this specific purpose, but is currently provisioned as a SLATE team member account. The GitHub Action executes with this account using credentials stored in a GitHub repository account that no one is able to read. This process prevents stealth commits and requires an approval process.

Individual Developer Tools and Prototype Tools

MiniSLATE

MiniSLATE is a personal development environment available for all application developers. MiniSLATE provides a self-contained complete with a local copy of the web documentation. All developers can download this codebase and develop their applications and workflow before pushing to the SLATE platform for scale testing.

- *Where maintained:* The MiniSLATE development environment resides on GitHub[15].
- *Granting/Revoking access:* Access to the MiniSLATE code follows the SLATE GitHub repository access process. No access requirements exist for the local instances.
- *Credentials:*
    - *Password:* The containers utilize default username/pwd that come with the images

SLATE-lite

SLATE-lite is a small lightweight version of SLATE that allows quick easy prototype deployments for testing federated deployments in a virtual testbed or other testing environment.

- *Where maintained:* The SLATElite codebase resides on GitHub.[16]
- *Granting/Revoking access:* Access to the SLATElite code follows the SLATE GitHub repository access process.
- *Credentials:*
    - *Password:* The containers utilize default username/pwd that come with the images.

SLATE User Facing Services

Web presence

Website

SLATE maintains a web presence for documentation, security policies and a link to the project portal. The web presence is a primary location of information for SLATE. There are no access requirements for the website. The website codebase resides on GitHub and allows appropriately authenticated team members to modify.

- *Where maintained:* The SLATE website[17] resides on GitHub[18] in the SLATECI repository similar to the applications.

[15] Minislate GitHub repository: https://github.com/slateci/minislate
[16] SLATElite GitHub repository: https://github.com/slateci/slatelite
[17] SLATE website: http://slateci.io
[18] SLATE website GitHub repository: https://github.com/slateci/slateci.github.io

- *Granting/revoking access:* Update access to the web site code follows the SLATE GitHub repository access process. The web site contains links for all of the blog posts as well.
- *Credentials:*
    - The website requires no authentication mechanism outside the development repository which follows the SLATE GitHub processes.

SLATE User Portal

The SLATE User Portal allows a SLATE cluster administrator, a SLATE developer, a SLATE application administrator and others to have a web based interface for installing applications from the catalog. The SLATE portal also allows for a subset of monitoring and management functionality for SLATE clusters.

- *Where maintained*: The SLATE portal[19] resides on GitHub[20] in the SLATECI repository.
- *Granting/revoking access:* Access to the web portal code follows the SLATE GitHub repository access process.
- Credentials:
    - *Password:* The SLATE User Portal requires institutional credentials through the Globus web authentication/authorization mechanism for access to the portal web pages.
    - *Token:* The SLATE User Portal has a privileged API token to pull out additional information.

SLATE Command Line Interface (CLI)

The SLATE Command Line Interface provides a command line interface for managing SLATE clusters. This interface runs on local machines to which a SLATE cluster administrator installs the packages. The software is part of the SLATE Client / Server API codebase. For development of the client, the SLATE team must authenticate with GitHub. The SLATE development workflow utilizes Jenkins and GitHub actions to build components

- *Where maintained:* The code for the SLATE Command Line Interface (CLI) resides on GitHub in the SLATECI repository.
- *Granting/revoking access:* Access to the SLATE CLI codebase follows the access to GitHub process.
- *Credentials:* The SLATE CLI accesses the SLATE API service using the SLATE API access token issued to the user through the SLATE User Portal, described above.

SLATE Sandbox

The SLATE Sandbox[21] is a web interactive tool that allows users to experiment with the SLATE CLI without having to install anything on their local machines.

---

[19] SLATE portal: https://portal.slateci.io/slate_portal
[20] SLATE portal GitHub repository: https://github.com/slateci/slate-portal
[21] SLATE Sandbox: https://sandbox.slateci.io/

- *Where maintained:* The SLATE Sandbox codebase resides on GitHub in the SLATECI repository similar to the applications.
- *Granting/Revoking access:* Access to the web site code follows the SLATE GitHub repository access process.
- *Credentials:* The SLATE Sandbox codebase requires SLATECI GitHub authentication and authorization. The actual SLATE Sandbox web portal requires access via institutional login or to sign up as a user.

SLATE Communication

Email Lists

- *Where maintained:* The various SLATE email lists all reside on Google Groups (https://www.googlegroups.com). The current email lists are:
    - slateci@googlegroups.com
    - slateci-discuss@googlegroups.com
    - slateci-news@googlegroups.com
    - slateci-ops@googlegroups.com
- *Granting/Revoking access:* Those listed as the Google Groups Owners grant/revoke access for different members. The Google Groups Owners consist of the SLATE PIs and appointed senior SLATE staff members.

Real-time chat

- *Where maintained:* The SLATE real-time chat hosts on Slack[22].
- *Granting/Revoking access:* The SLATE PIs and appointed senior members of the SLATE team act as managers of the SLATE Slack channels.

Video content

SLATE utilizes YouTube for distributing training videos.

- *Where maintained:* The main video channel for SLATE hosts at YouTube.com[23]
- *Granting/Revoking access:* The main SLATE PI grants update access to the YouTube channel.

Shared presentation, tutorial and file repository

SLATE utilizes a Shared Google Drive for storing internal documentation, tutorials, presentations and paper links.

- *Where maintained:* The SLATE Shared Google Drive is a part of the University of Chicago's contract with Google and resides in Google's cloud service
- *Granting/Revoking access:* The SLATE PI grants update access to this shared folder and its folder hierarchy.

[22] SLATE Slack channel: https://slateci.slack.com
[23] SLATE YouTube channel: https://www.youtube.com/channel/UCbJ654YHcv-4nni-8tzrINg

Overleaf

SLATE utilizes Overleaf for submitted papers.

- *Granting/Revoking access:* The SLATE PI grants update access to this shared folder and its folder hierarchy.