Gemini Program Platform

Software Conceptual Design

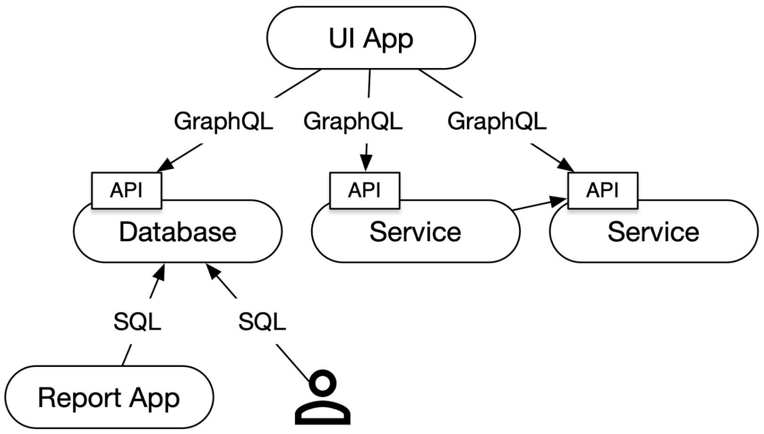
Version 1.1 - Last updated: 2019 December 4

1 Introduction

This document proposes a software architecture for implementing the proposals in Section 4 of the Operational Concepts Document. We propose a set of communicating software components, broken into two main categories: user-facing applications, and back-end support systems. We describe the system as a whole, followed by descriptions of the primary functions of each component. We conclude by proposing a cloud-based development and deployment strategy, to provide the physical runtime environment for GPP.

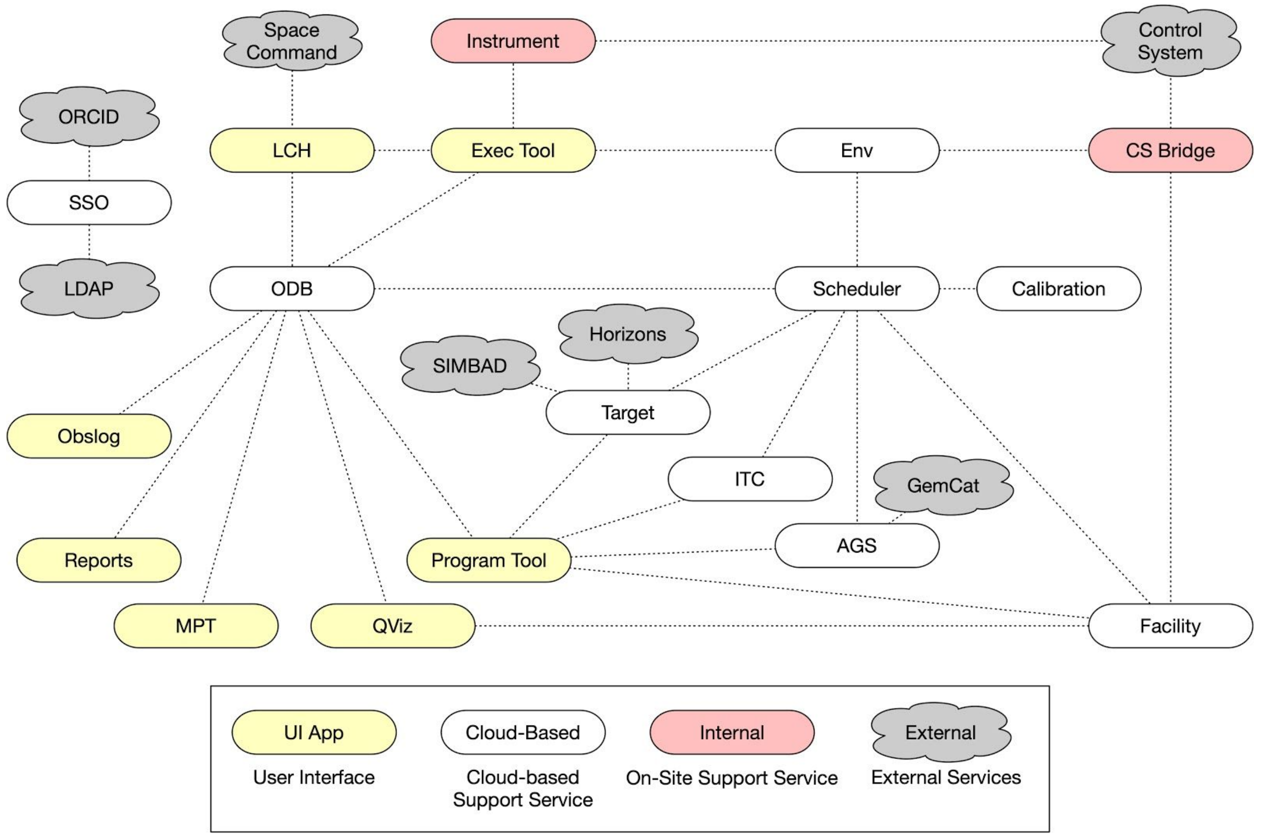
2 Software Concept

The GPP will offer an array of special-purpose UIs (User Interfaces) for external PIs and operations staff members. Supporting these tools is a centralized database and a collection of special-purpose services.



Here, a user interface application is shown communicating with two services and a central program database via their public APIs. Services will also use the public APIs of other services. A reporting application might bypass the public API and directly query the database, and query access should also be available to users.

The particular constellation of applications and services envisioned for this project is depicted below.



User facing services (yellow) provide user interfaces intended to be used by PIs, NGOs and/or science and operations staff. Support services (white) provide API access primarily for programmatic use. Internal support services (red) interact with Gemini facility hardware and must be installed on-site. External services (gray) provide necessary functionality but are not controlled by Gemini. Dotted lines indicate communication between systems.

A brief description of each service is provided in the next section. Each description enumerates the service's **primary functions**, each of which is either

A. new or substantially improved, in which case we reference a technical proposal from the OCD (marked [P-3] for example); or
B. existing and reasonably necessary (see OCD §1), and will have continuing support (marked [CS]); or
C. a consequence of the system architecture (marked with [A])

Note that technical proposals [P-27, P-28] apply across the system architecture and are not associated with specific primary functions.

While this document is focused on the conceptual design, prototyping of key services that are independent of the operational concept has been ongoing for at least a year. A number of important implementation decisions have been tentatively taken and are a useful indication of the progress that has already been made. We highlight these implementation decisions as shown below.

> GPP will be implemented in the Scala programming language in a functional style. This decision is based upon negative experience with maintaining loosely typed imperative programs at Gemini. Our experience with functional programming in Scala over the last several years has confirmed that it generally yields reliable systems that are easier to maintain.
>
> GraphQL has been selected for service APIs because it is convenient for programmatic access even by non-expert developers, allows the client to specify just the subset of data that is needed, and permits the API to evolve without explicit versioning.

2.1 User Facing Applications

Making use of the cloud-based services are a collection of (mostly) web applications. The Program Tool is intended for PIs and NGOs as well as Gemini staff, while all others are for internal use.

> To take advantage of the Scala program models developed for use throughout the remainder of the system, all user interfaces will leverage Scala-js. Scala-js combines strong static typing and interoperability with JavaScript libraries and has been successfully employed in a recent project at Gemini, the predecessor of the Execution Tool discussed below.

2.1.1 Program Tool

The Program Tool is the main end-user web application for external PIs, NGOs and staff. It assists in proposal preparation, evaluation, and management; as well as engineering tasks.

Primary functions:

- Provide a modern, intuitive web-based user-interface. [P-1, P-2, P-5]
- Allow users to experiment anonymously by specifying science goals/constraints and viewing compatible instrument configurations and estimated time required. [P-3, P-6, P-7, P-8, P-9]
- Allow users to promote an experimental program into a persistent program by authenticating, and provide additional information necessary for proposal submission. [P-7, P-8, P-9]
- Allow user to specify timing and ordering constraints among observations. [P-12, P-13, P-14]
- Allow users to submit the program for approval and await time allocation. During this period most editing features are disabled for PIs. [CS]

- Allow users to manage programs by making tweaks (such as specifying dither patterns or additional calibrations), and monitor execution progress. [CS]
- Allow users to visualize the science field, guide stars, and offset positions. [CS]
- Allow users to view archived programs after completion, and use these as templates for creating new programs. [CS]
- Allow users to view a history of changes. [P-4]
- Automate tedious, error-prone tasks such as specifying the acquisition and science sequences and defining associated calibration observations. [P-9, P-10, P-11]
- Manage auxiliary files (finding charts for example) related to science programs. [CS]
- Allow staff to set quality assessment status for data sets. [CS]

2.1.2 Execution Tool

The Execution Tool is the main application used for observation execution. It will be an enhanced version of Gemini's new Sequence Executor. Primary functions:

- Log weather conditions that cannot be reliably determined from sensors. [P-24]
- View the Scheduler's suggestions for upcoming observations and select from among them (or enter an arbitrary observation). [P-21]
- Visualize and execute a sequence, including today's SeqExec pause/abort, etc. [CS]
- Enter requested feedback during sequence execution as necessary. For example, to determine when an acquisition completes, whether an exposure time adjustment is necessary after the first science dataset(s), or when a non-signal-to-noise observation completes. [A]
- Enter comments that will be interleaved into the execution history. [P-24]

2.1.3 Laser Clearing-House (LCH)

The LCH application manages laser timing windows as determined by requests to the Space Command authority. The existing LCH application will be adapted to work with the new model and will gain an API layer so it can be used by other services. Primary function:

- Maintain an up-to-date list of approved observing windows for all active observations, and make this information available to client applications such as the scheduler. [P-20, P-23]

2.1.4 Queue Visualization (QViz)

The Queue Visualization application is an interactive reporting tool showing instrument feature demand and target visibility across the database of science programs. It is used by Gemini staff as part of facility planning (filter swaps, engineering periods, etc). The existing desktop application will be adapted to work with the new model. Primary function:

- Provide a visualization showing instrument feature demand and target visibility across the database of science programs [CS]

2.1.5 Manual Planning Tool (MPT)

The Manual Planning Tool (previously known as the Queue Planning Tool) is a desktop application used to construct observing plans for a single night. Largely supplanted by the automated Scheduler in the GPP, it may still be used by Gemini staff for special purpose nights such as classical runs or instrument commissioning. The existing application will be adapted to work with the new model. Primary function:

- Visualize a planned night of observing and assist with queue construction, matching each queue to possible observing conditions. [CS]

2.1.6 Reports

Reports will be available via SQL queries to a read-replica of the observing database, using available commercial or open-source tooling. The software group will ensure that required information is available, identify and make recommendations for query tools, and write an initial set of queries. Primary function:

- Provide software endpoints for arbitrary reporting purposes, and guidance on using these endpoints. [P-29]

2.1.7 Observing Log (Obslog)

The Obslog provides an overview of the night’s activities, changing weather conditions, telescope faults, and running commentary from operations staff. Primary functions:

- Provide an integrated interface to view observing events, sky almanac information, reported faults, and observer notes in a single timeline. [P-24, P-25]
- Provide an interface for manual entry of events including arbitrary observer comments (optionally private to staff) and observed conditions. [P-24, P-25]

2.2 Support Services

All support services play important roles but two services in particular occupy a central place in the GPP architecture. These are the Observing Database, where science programs are kept, and the Scheduler, which suggests which observation(s) should be executed next. The remainder of the services provide support for specific tasks such as integration time calculation and automatic guide star search. The following sections provide an overview of each of the cloud-based services.

2.2.1 Observing Database (ODB)

The ODB stores science programs, execution history, and other information necessary to support the user-facing applications. It is the central source of truth in the system and its model is the foundation of most GPP features.

A science program may consist of observations that will ultimately be executed at either site depending upon target location and required instrument features. For this reason, and because the scheduler needs a consistent view across all available observations, the GPP features a single centralized database of science programs.

> We have successfully prototyped a relational database ODB implementation using PostgreSQL. As with other services, end-user applications and tools will access the database via GraphQL queries.

Primary functions:

- Store program and observing event data in a secure, reliable, and consistent way. [CS]
- Provide software endpoints that allow end users and client applications such as the Program Tool to retrieve and manipulate program data (such as time account reporting and corrections). [P-19, P-26, P-29]

2.2.2 Scheduler

The automated Scheduler plays a central role in improving observing efficiency at Gemini, particularly as we transition to site-independent observations and a single shared observing database. It examines the pool of available observations and weighs factors such as target availability, the current weather conditions, program completion, and facility schedules to suggest the next “best” observation at any time.

While useful regardless, the Scheduler is absolutely required to handle the complexity of followup urgent or interrupting observations originating from survey telescope (e.g., LSST) events. Efficiently managing a queue under these conditions without automation is not possible.

Primary functions:

- Maintain an up-to-date schedule for both telescopes, based on observing conditions, available resources, and science priorities. [P-15, P-16, P-20, P-21, P-22]
- Provide software endpoints that allow end users and client applications such as the Execution Tool to retrieve scheduling data. [P-29, A]

2.2.3 Single Sign-On (SSO)

All users of GPP services must be authenticated, with the exception of experimental programs in the Program Tool. Primary functions:

- Authenticate external users via ORCID, or similar open identity/authentication standard. [A]
- Authenticate internal users via Active Directory. [CS]
- Provide a secure token that can be verified and used by any other GPP service. [A]
- Provide a mechanism to grant and revoke user roles and permissions. [CS]

> SSO will maintain a database mapping users to roles, and will issue a Json Web Token (JWT) containing these roles, for use by other systems. The JWT will be signed with an asymmetric key, allowing other services to verify authenticity.

2.2.4 Target Database

The Target Database assigns unique identifiers to targets and provides name resolution services. It provides a cache of names, magnitudes, and coordinates (including ephemerides via JPL's horizons). It will provide the answer to the question, "where is target *t* now", by applying proper motion corrections for sidereal targets or interpolating non-sidereal ephemeris elements. This facility is used broadly but in particular by the Program Tool and the Scheduler.

Primary functions:

- Look up and store target names, magnitudes, and ephemerides using existing catalog services. [CS]
- Keep target information up to date, ensuring in particular that sufficiently precise ephemerides are available for both sites, over a reasonable time window (a year into the future, perhaps). [CS]
- Provide a mechanism for storing user-defined targets. [CS]
- Provide software endpoints to allow end users and client applications such as the Program Tool and Scheduler to query and manipulate target information. [P-19, P-29, A]

2.2.5 Automated Guide Star (AGS) Service

The Automatic Guide Star service takes an observation description, a time and an observation duration and calculates a mapping from particular guiders to guide stars such that the observation can be executed. It will need the target service to determine where the target and guide stars will be over the course of the observation. It makes use of the external (to this project) Gemini Catalog Service to find guide star candidates. While the Gemini Catalog

Service is currently unavailable outside of Gemini, we will need to make it accessible from external services.

Primary function:

- Compute guiding configurations. [CS]
- Provide software endpoints to allow end users and client applications such as the Program Tool to query guiding configurations. [P-29, A]

2.2.6 Integration Time Calculator (ITC)

ITC provides estimated exposure times and signal-to-noise ratio. It is used by the Program Tool UI, the Scheduler and other services that need to estimate exposure times. Initially this will be an encapsulated version of the existing Gemini ITC, with a GraphQL API front end.

Primary function:

- Compute exposure times and counts given an instrument configuration, target information, and a desired final S/N. [P-6, P-7, P-9]
- Provide software endpoints to allow end users and client applications to request ITC calculations. [P-29, A]

2.2.7 Facility Service

The Facility Service will track instrument and instrument feature (masks, filters, etc.) availability along with instrument port assignment. In addition to supplying current status, it will track past and, to the extent that it is known, future availability. The Scheduler presents the Facility Service with a set of constraints (required filter, mask, etc.) and receives a set of timespans during which the constraints are met. It uses that information to ensure that only executable sequences are suggested. In addition, we can compute probability of execution by asking similar questions spanning the remainder of the semester. AGS also relies on this service because port assignment can impact the reachability of OIWFS guide stars.

Primary functions:

- Track availability of observatory resources (telescopes, instruments, filters, masks, etc.) through time, maintaining a historical record. [P-20, CS]
- Provide a user interface so science staff can update resource availability for spans of time. [CS]
- Provide software endpoints so end users client applications such as the Scheduler and Program Tool can query resource availability at a point in time. [P-29, A]

2.2.8 Calibration Service

The calibration service determines which calibrations (arc, flat, standard …) are required for a given data set, and maintains a database of existing calibrations. The Scheduler will make use of the service to ensure that calibrations are scheduled as necessary.

Primary functions:

- Compute required science calibrations for a given observation. [P-17, P-18]
- Compute required daytime calibrations. [P-18]
- Maintain a database of references to calibration data, to support cases where existing calibrations can be reused (e.g., longslit baseline standards, twilight flats, biases, etc). [P-18]
- Provide software endpoints so end users and client applications such as the Program Tool and Execution Tool can query required calibrations. [P-29, A]

2.2.9 Environmental Monitor (Env)

The Environmental Monitor aggregates weather and other environmental conditions from available monitors, as well as human input (via the Execution Tool) in cases where automated data is not available or not reliable. This information is used by the Scheduler. The Env service reads EPICS channels (see Control System Bridge Service) that publish weather data.

Primary functions:

- Maintain a database of observing conditions, as measured through time. [P-20]
- Record changes as measured by automated environmental monitors. [P-20]
- Provide software endpoints so end users and client applications such as the Scheduler and Execution Tool can record observed conditions, and query conditions at a point in time and space. [P-29, A]

2.3 Internal Support Services

Internal support services require direct access to EPICS or other local information, filtering it and making it securely available offsite.

2.3.1 Instrument Service

The instrument service is the back end for the Execution Tool. It applies configurations to instruments, records execution events, and collects values required by FITS headers. This service is currently under development and will require updates to work with the new program model and database.

Primary functions:

- Translate logical sequences into low-level instructions for instruments and related telescope resources. [CS]
- Monitor events as reported by telescope systems. [CS]
- Provide software endpoints to allow client applications such as the Execution Tool to execute sequences and monitor events. [A]

2.3.2 Control Systems Bridge

Gemini low-level telescope command, control and status is accomplished via EPICS. Most facility instrument control systems utilize EPICS as well. Newer instruments like the Gemini Planet Imager employ the Gemini Instrument API (GIAPI) as an alternative to EPICS. Taken together, we refer to all of the low-level command, control and status systems as Control Systems. The Control Systems Bridge exposes low-level information via a web API so it can be observed by services running in the cloud.

Primary functions:

- Maintain a set of EPICS channels or GIAPI topics that remote applications will be allowed to monitor. [A]
- Provide software endpoints to allow cloud-based client applications such as the Environment Service to monitor EPICS channels or GIAPI topics. [A]

3 Deployment Approach

Section 2 introduces the logical system components and their interactions. This section specifies how these systems will be deployed, monitored, and managed on physical hardware.

3.1 Continuous Deployment

Modern software practices encourage development as a continuous series of minor improvements, which are less disruptive and encourage a tighter and more effective feedback loop between users and developers. In support of this practice GPP will adopt a continuous integration strategy, in which all software changes are automatically verified against the test suite; and a continuous deployment strategy, in which verified changes are automatically released to the staging environment for user testing. Promotion from staging to production will initially require management approval, but may become automatic once we gain confidence with the deployment process.

3.2 Cloud-Based Deployment

It is a requirement that programs be permitted to use instruments at either site, perhaps chosen based on real-time conditions, and the telescopes must be scheduled together. Therefore a central, shared system is required at least for scheduling functionality. In addition, modern software practices tend strongly toward cloud deployments rather than self-hosted data centers. These factors led us to select a cloud-based deployment model.

We investigated Google Cloud, Microsoft Azure, and Amazon Web Services cloud platforms, and consulted with industry developers who are using these technologies. They uniformly advised against using one of the "big" platforms directly, and instead recommended using a PAAS (Platform as a Service) called Heroku, which hides the complexity of the underlying cloud platform (Amazon Web Services in this case) and is appropriate for "small" systems such as GPP. To our knowledge there is no equivalent for Google or Microsoft cloud platforms.

We identified the following desirable properties, all of which are met by Heroku.

- Application-oriented deployment, without the need to allocate and configure physical or virtual machines. GPP is a small system by industry standards, and current provisioning solutions such as Kubernetes are much too complex for our use case, however manual provisioning via ITS service request or a cloud platform console does not meet our automation requirements. Heroku offers a middle ground, allowing for deployment of an application without concern for provisioning, in exchange for limited flexibility, without tying our code to the platform. It will be possible to deploy services unmodified on other platforms or on Gemini hardware should the need arise.

- Continuous deployment directly from GitHub, without custom tooling. We will build and stage applications for deployment directly from GitHub, rather than maintaining in-house deployment infrastructure. Heroku provides this facility, and also deploys test applications for all proposed changes.

- Multiple environments for test, staging, production. Heroku provides these environments, with automated deployment of test applications, automated deployment to staging, and on-demand promotion to production.

- On-demand resizing and scaling with integrated load-balancing. In response to increase in demand we need the ability to resize our application servers and scale out multiple replicas of an application, with load balancing. Heroku provides this via their web and command-line interfaces. Server power and configuration can be updated at any time, and we pay only for instance we use, prorated to the minute, billed monthly.

- Managed PostgreSQL with managed backups. Heroku provides secure, managed PostgreSQL databases with managed backups, as well as some other options such as read replicas which may be helpful in the future.

- Container support. Most of our applications will be built for the Java Virtual Machine (a hardware abstraction that minimizes the need to tailor software to a specific processor or operating system), but in some cases we may need native components. In these cases we will need to deploy containers (isolated environments for programs that run on Linux), and Heroku provides Docker support for this purpose.

- Integrated logging, monitoring, and alerts. System monitoring and cross-service monitoring is another source of complexity that we wish to avoid. Heroku provides logging, monitoring, and alerts for applications; as well as offering several add-on services for systemwide monitoring. Logs are stored in S3 (Amazon's storage system) but can also be drained to central syslog if desired.

3.2.1 Cybersecurity

Heroku provides for infrastructure security including access to physical/virtual machines, storage, and backups; and provision of SSL for encrypted communication. GPP will provide application-level security to ensure that services are accessible only to authenticated/authorized users. It is anticipated that authentication itself will be delegated to Active Directory for internal users and ORCID for external PIs.

All code that is committed to source repositories will be reviewed by the Software department for cybersecurity impact.

3.3 On-Site Deployment

A small number of GPP sub-systems require direct access to Gemini hardware via EPICS and GIAPI interfaces, and these will be deployed locally at each site.

- The Instrument Service controls instruments while executing observing sequences and must be located on-site. Note that its user interface is available remotely.

- The EPICS Bridge connects to the local real-time network and forwards read-only information to the cloud-based Facility and Environment services.

On-site services will be deployed on local VMs (Virtual Machines) or via VSphere Integrated Containers, supported locally by the Gemini Information Technology Services group.