MXL Tech Memo

Document 0.1

Authors:	Jimmy Blanchard
Date:	August 13, 2012
Title:	Mercury2 Preliminary Design

Contents

1 Introduction

Mercury22 is the spiritual successor to the Mercury Ground Station System. It will employ recent advances in web application development and software defined radios to allow satellite operators to easily and efficiently command a network of ground stations across the world.

2 Requirements

Mercury2 must be able to perform several core functions. There will be two separate applications: Mercury2 (the web interface), and Iris (ground station task manager). Their core requirements are defined below.

2.1 Mercury2 Requirements

Mercury2 will run on a central dedicated server.

- Complete user authentication and authorization
  - Supports user registrations and account management
  - Maintains user permissions
- Thorough access logs
- Satellite operator interface
  - Allow satellite operators to automatically schedule passes to best available ground station in specified time window
  - View signal data for prior passes
- Ground station operator interface
  - View ground station schedule
  - Manually download schedules and upload results if not connected to TCP
  - Manual override to disable automatic scheduling
- Complete administration panel
- Easy to use interface
- SSL encryption and protection from exploits such as XSS cross-site scripting and SQL injection

2.2 Iris Requirements

An instance of Iris will run on a computer attached to each ground station radio.

- Support multiple radio drivers
- Periodically connect to Mercury2 GUI via TCP to download pass schedule and connection settings
- Automatically configure radio and transmit/capture data during the time specified by the schedule
- Provide satellite operators with a TCP socket to send commands and receive telemetry data
- Secure, private key verified transmissions

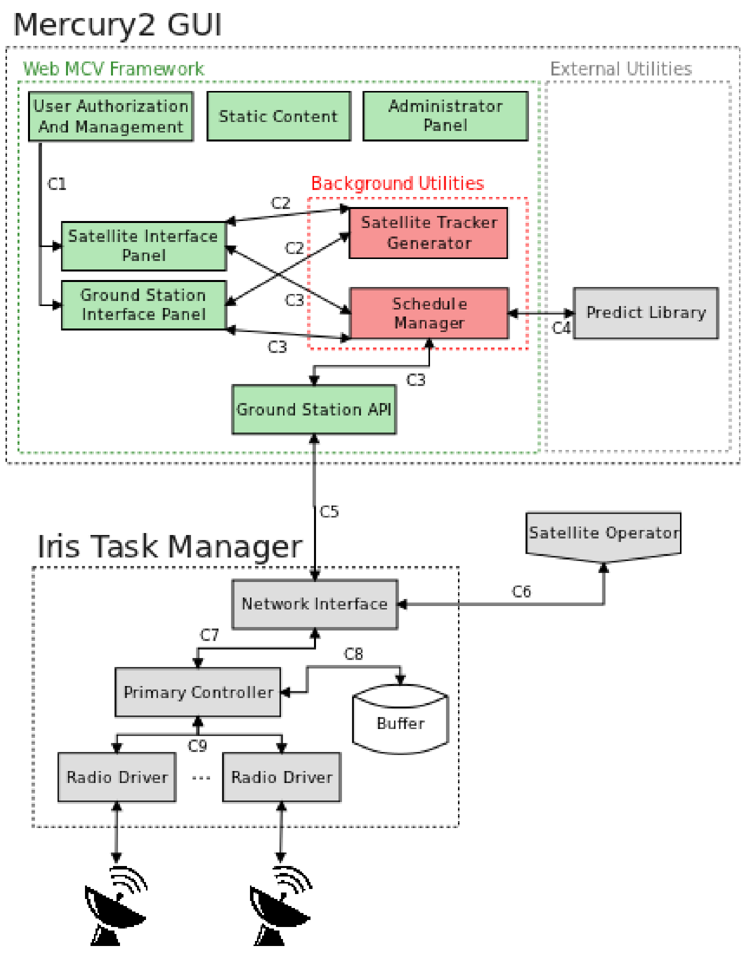

Figure 1: Mercury2 and Iris Architecture Overview

3 Application Architecture

3.1 Mercury2 GUI

The Mercury2 GUI will most likely be a PHP based web MVC framework (e.g. CodeIgniter, CakePHP). An application following this design pattern is broken into models (handle database interactions and various logic tasks), views (the mark up used to display content to the user), and controllers (classes that route users to the appropriate location and join models and views). Figure 1 (above) roughly illustrates the primary controllers that will be used in the application. Each component is explained below.

3.1.1 User Authorization and Management

Controller Class UserController

This section of the website is responsible for managing users and user permissions. It performs several standard user functions, such as account editing, sign-ups, and password recovery. Account creation will be free and open to the public (except, perhaps, in the case of for-profit groups, for which a tiered priced system could be implemented). Once they have made an account, users will be able to submit requests to add satellites (from a list of available TLE's) or ground stations to the website for administrator approval. If approved, the user that submitted the request will gain control of the satellite or ground station's page (described in greater detail in sections 3.1.4 and 3.1.5).

3.1.2 Static Content

Controller Class ContentController

This will simply display administrator editable static content to the user such as the FAQs, API documentation, and Terms of Service.

3.1.3 Administrator Panel

Controller Class AdminController

The administrator panel will allow administrators to manage Mercury2. In addition to being able to do anything that any other appropriately permissioned user can do (such as viewing signal data, pass schedules, etc.), administrators will be able to:

- Manage users
- Configure website settings
- Create and edit static content
- Manually create and edit satellites and ground stations
- Review user requests for new satellites and ground stations
- Review application and API statistics

3.1.4 Satellite Interface Panel

Controller Class SatelliteController

Each satellite that is configured on Mercury2 will have its own set of user accessible pages. Administrators may give any number of users permission to perform some or all actions for any given satellite. In addition, previously authorized users (with the correct permission) may give other users permissions on the page as well. User permissions are managed and verified by the User model and the UserController class described in section 3.1.1. The satellite pages will allow satellite operators to:

- View upcoming passes and reserve radio use
- View streaming signal data (if the connection permits)
- View satellite tracker and AZ/EL chart
- View prior pass history
- Change basic page design elements such as the logo and color scheme
- Upload images and create posts about the satellite
- Send an email or text message reporting pass results/status
- Give other users permissions on the page
- Communicate with ground station operators using private messages

3.1.4.1 Optional Public Page In addition to the restricted satellite management page, each satellite will have an optional public page. Some of the page elements described above such as the image gallery and satellite tracker may be added to the public page.

3.1.5 Ground Station Interface Panel

Controller Class StationController

Each configured ground station will also receive its own set of pages. The ground station interface will be very similar to the satellite interface in that authorized ground station operators will be able to:

- View upcoming and prior passes for the station
- View a satellite tracker focused on the ground station
- Customize design elements and manage images
- View satellite pass history for the station
- Add elements to the public page

Ground station pages will also allow ground station operators to:

- Remove the ground station from the scheduling pool (to take manual control)
- Manage queued scheduled passes
- Download schedule so that they can manually feed it to ground station (in the case of a ground station with no internet connection)
- Manually upload results of a scheduled pass if the ground station was offline at the time
- View ground station status and health attributes
- Send commands to the ground station
- View streamed signal data during passes (if connection permits)
- Manage API connection keys (described in section 3.1.6)
- Communicate with satellite operators using private messages

3.1.6 Ground Station API

Controller Class APIController

The APIController will provide API hooks that will allow ground stations to download schedules, report their health and status, and stream signal data to Mercury2 through a UDP socket. The API will consist of a standard REST route scheme and is explained fully in section 3.3.5.

3.1.6.1 Conflicting Ports Because any two satellites may have overlapping pass times (at any combination of ground stations), the UDP sockets opened by the API for the purpose of collecting signal data must be unique and tied to each pass. In addition, they will need to be closed and freed after each pass ends. This could be achieved using a simple CRON job.

3.1.7 Background Utility: Satellite Tracker Generator

Controller Class TrackerController

This controller will support an internal API that will provide the necessary framework and configuration settings for instances of the satellite tracker and AZ/EL chart on satellite and ground station pages. Explained more in section 3.3.2.

3.1.8 Background Utility: Schedule Manager

Controller Class ScheduleController

The schedule manager will be responsible for managing the requests for ground station use by satellite operators. It will ensure that only one satellite is using any given ground station at a time. All signal and telemetry data collected will be tied to a specified scheduled pass. The schedule management flow is described in below in figure 2.

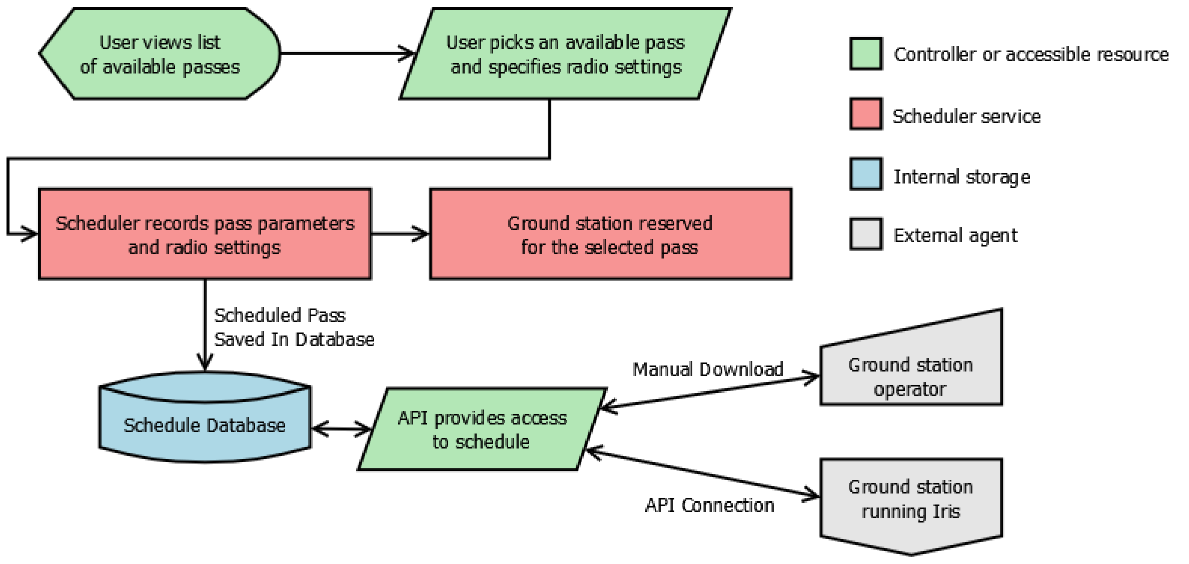
User views list of available passes
User picks an available pass and specifies radio settings
Controller or accessible resource
Scheduler service
Internal storage
External agent
Scheduler records pass parameters and radio settings
Ground station reserved for the selected pass
Scheduled Pass Saved In Database
Schedule Database
API provides access to schedule
Manual Download
Ground station operator
API Connection
Ground station running Iris

Figure 2: Scheduling process flow

3.1.8.1 Socket Ports Because the Iris ground station software must provide TCP sockets to enable satellite operators to directly control the satellite and receive telemetry data which may result in overlap between passes (because telemetry data can't be uploaded at the same rate it's received), the scheduler will also determine the ports to use for these sockets. This will allow a scheduled pass to occur while the results of the last pass are still being streamed. This is described more in section 3.2.2.

3.1.9 External Utility: Predict Library

Type External API

Mercury2 will make use of, at least initially, the popular *Predict Library*[1] by John Magliacane to calculate pass times. *Predict* provides an API via a network socket that will allow Mercury2 to pass it TLE's and retrieve calculated pass times.

3.2 Iris Task Manager

The Iris Task Manager program will consist of a relatively simple Python script running on a computer connected to the radio. It will be responsible for downloading the latest schedule, reporting the health of the ground station, connecting to the radio via a set of drivers, streaming signal data, and providing a command/telemetry connection to satellite operators.

3.2.1 Network Interface

The network interface class allows Iris to communicate with the Mercury2 GUI. Most communications with Mercury2 will be facilited by connecting to the endpoints provided by the Mercury2 REST API (described in section 3.1.6 and 3.3.5), in addition to a UDP socket for signal data streaming. The network interface class will be responsible for the following:

- Downloading the ground station schedule
- Reporting the health of the ground station
- Streaming signal data

In addition, the network interface class must also provide mechanisms to open and close the raw command and telemetry sockets as defined in the schedule.

3.2.1.1 Network Security To ensure secure communication, all interactions between Iris and Mercury2 will be encrypted with a standard SSL encryption scheme. In addition, all communications with Mercury2 will need to be verified using a HMAC-SHA256 private key encryption system which will ensure only the authorized ground stations can access the API.

3.2.2 Primary Controller

The primary Iris controller will initialize the program and create the main program loop. This main loop will be responsible for maintaining the state of the ground station and executing scheduled passes. The tasks that it will perform are described below.

Fetch Schedules Periodically (every few minutes) query the network interface class to download the most recent version of the schedule. Also check for offline versions of the schedule if no network connection is available.

[1] http://www.qsl.net/kd2bd/predict.html

Report Status Every request for a schedule update will also include information about the health and status of the ground station. For example, the currently executing task and radio parameters may be included. This information will be reported on the ground station and satellite operator pages in the GUI. This mechanism is described in 3.3.5.

Execute Scheduled Passes Once a scheduled pass starts, Iris will:

- Attempt to configure the radio via the selected radio driver using the connection settings defined by the schedule
- Update the radio status to reflect the connection
- Open two sockets to allow satellite operators to submit commands to the satellite and receive telemetry from the pass (ports determined by scheduler to prevent conflict)
- Create a thread to maintain the connection during the pass (correct doppler, etc.)
- Create a thread to store raw telemetry data into a buffer and feed it to the satellite operator via the telemetry socket
- Create a thread to stream signal data back to the GUI by invoking the network interface class

Complete Scheduled Passes Once a pass has finished (determined by the schedule calculated by the Predict Library in the GUI), Iris will:

- Update the radio status
- Destroy connection maintenance thread
- Destroy signal data stream thread
- Check if all telemetry data has been transmitted to the satellite operator. If it has, destroy the thread. Otherwise keep the thread until it streams all of the telemetry data.

3.2.2.1 Telemetry Persistence All of the telemetry data collected will be stored for a certain period, in case there was an error transmitting or receiving it the first time. This will also allow operators running ground stations without internet access the ability to manually retrieve the telemetry from a pass and send it to satellite operators.

3.2.3 Radio Drivers

Each type of radio connected will need its own driver class. These classes will be standardized and must provide a mechanism to update the radio settings and gain access to the radio's command and telemetry streams.

3.3 Connection Details

The connections shown in figure 1 (labeled C#) are the primary data routes for the Mercury2/Iris system. They are explained in detail below.

3.3.1 C1 - User Authorization

Connection Type Database Model Interaction

When a user requests a satellite or ground station page in their browser, their permissions will be retrieved from the database to determine what control, if any, they should have over the page. Users can be given an arbitrary amount of control over any satellite or ground station by either website administrators or existing users on the ground station/satellite page with the appropriate permissions (i.e. the ability to add new team members).

3.3.2 C2 - Satellite Tracker Tool

Connection Type Database Model Interaction

When a satellite or ground station page contains an instance of a satellite tracker, it will use the satellite tracker generator (section 3.1.7) to generate the appropriate configuration for that tracker (e.g. available satellites, ground stations, visual options, etc.). By seperating the tracker configuration generation from the satellite and ground station controllers, it is much more modular and can be included in any configuration on any arbitrary page.

3.3.3 C3 - Schedule Manager

Connection Type Database Model Interaction

These connections will allow any page on Mercury2 (particularly the satellite and ground station control panels) to view a subset of the pass schedule. For example, when a user loads a satellite interface page the schedule manager will be queried, through this connection, for the relevant pass schedule. This connection will also allow authorized users to add passes to the schedule or cancel upcoming passes.

3.3.3.1 API Use The Mercury2 API will connect to the schedule manager using the same paths and access methods as any other page to retrieve the needed schedule. The only difference is how it will be presented to the end user (e.g. human-readable table vs machine readable API format).

3.3.4 C4 - Predict Library API

Connection Type External API Connection

Mercury2 will use the *Predict Library* (section 3.1.9) to calculate the times of upcoming passes. *Predict* is a C application that runs independent of the Mercury2 MVC framework server application. *Predict* provides a TCP socket based API that will allow the Mercury2 schedule manager to load current TLE's and retrieve upcoming passes.

3.3.5 C5 - Primary Ground Station API

Connection Type External API Connection

This API will be used by any configured Iris ground station to connect to Mercury2. It will allow ground stations to retrieve the most recent version of its schedule, report its health and status, and stream signal data back to Mercury2. This API will be split into two seperate sections. The first will be a standard HTTP REST API with the following paths and functionality.

/api/schedule/:station-name.format The ground station will POST to this path with its current health and status. The Mercury2 API will respond with a .format (either JSON or XML) encoded string containing the ground station's schedule and radio connection settings for each scheduled event. It will also transmit a checksum to allow the ground station to verify the integrity of the transmission.

In addition, the API will also provide a UDP socket (with a port unique to each pass scheduled) that will allow the ground station to stream back its signal data (signal strength, etc.) to the Mercury2 GUI.

3.3.5.1 API Security API connection security will consist of standard SSL encryption to protect against middle-man attacks and HMAC-SHA256 private key encryption to prevent unauthorized access. Ground station operators will be able to generate and retrieve API keys from the ground station interface page.

3.3.6 C6 - Satellite Operator Connections

Connection Type External Socket Connection

When a pass starts, the ground station will open two sockets for use by the satellite operators. One for transmitting commands to the satellite, and one for receiving telemetry data. These sockets will be opened on the ports specified in the schedule (which will be reported to satellite operators when they schedule a pass). After a pass ends, these sockets will be closed.

3.3.7 C7 - Primary Controller and Network Interface Interaction

Connection Type Internal Connection

The primary controller (section 3.2.2) will interact with the network interface class to receive schedule data, transmit the ground station health, and provide raw socket access to the radio for satellite operators.

3.3.8 C8 - Telemetry Buffer

Connection Type Internal Connection

The primary controller will record all received telemetry and beacon data into an internal buffer (most likely just a file dump). This data will be associated with a scheduled pass and timestamp. It will also be streamed through the telemetry socket to the satellite operator as it is received from the satellite. Because telemetry data is often very large per pass (on the order of gigabytes), it will be rotated every few days. This allows ground station operators to retrieve a copy of the data if requested by the satellite operator (in the case of a transmission or receive failure during the initial pass).

3.3.9 C9 - Radio Driver Interface

Connection Type Internal Connection

The primary controller will connect to each radio driver using a standard class interface. This will provide the mechanism to allow the ground station to configure the radio settings, as defined by the pass schedule. It must also be able to provide a socket to the radio command and telemetry feeds. The specifics of this interface will be defined at a later time.