MCT: A Command Scheduling Application for Mission Operation and Control (MOC) of the McMaster PRESET CubeSat

Austin Liu (liu461@mcmaster.ca)
PRESET Systems Team Lead
McMaster Interdisciplinary Satellite Team

Revision A
Sep 18, 2023

Scope of Document

This document describes a project proposal for MCT, a web application to assist the operation of the McMaster PRESET CubeSat. The first section gives an overview of the proposed project. The second section outlines the requirements specification.

Project Overview

Background

McMaster Interdisciplinary Satellite Team (MIST) launched its first-ever satellite, the NEUDOSE CubeSat, in March 2023. To communicate and command the satellite, a permanent ground station (GS) has been set up on the campus of McMaster University (43.2585° N, 79.9201° W). For radio frequency (RF) signals to reach each other, the satellite must be orbiting above the GS horizon, or 0° elevation. Higher elevation means shorter distances, stronger signals, and less noise. Although NEUDOSE will orbit the earth up to 16 times per day, it will only pass over the Hamilton sky 3-5 times during that day. Often, these overpasses are not ideal because their maximum elevations are far from the center of the sky, where the signals are the strongest. Operators use orbital prediction software such as Gpredict[1], N2YO[2], GMAT[3], and PyOrbital[4] to find overpasses in the future. For each suitable pass, an operator will wait until just before the Acquisition of Signal (AOS, i.e. going above 0° elevation), log into the mission control computer, launch software for RF communications, send the satellite commands, wait for responses, and finally closeout operation by the Loss of Signal (LOS, i.e. going under 0° elevation).

[1] http://gpredict.oz9aec.net/
[2] https://www.n2yo.com/passes/?s=56315
[3] https://software.nasa.gov/software/GSC-18094-1
[4] https://github.com/pytroll/pyorbital

During the Launch and Early Operations (LEOP) phase of NEUDOSE, this approach proved to be problematic. Operators were not available during suitable passes, which can happen at midnight, early morning, or the middle of the day. Launching software and entering commands were mundane and error-prone for a human operator. Command history was not saved under Configuration Management (CM), making it difficult to trace the current system state. The Flight Model (FM) and the Engineering Model (EM) had separated software interfaces, resulting in inconsistent system verification and operator training. Finally, access control was difficult to manage with a single password-protected computer.

Work is underway at MIST to develop a second CubeSat named PRESET. Deployed to a sun-synchronous orbit (SSO), PRESET will study the dynamics of electrons in the magnetosphere. The mission is currently in the Concept Design (MCR) phase and will be ready for launch in 2025. The amount of data generated by the science instrument will increase, while the number of access windows will decrease due to the change in orbit. For all the above reasons, it will be crucial to have mission operation software that is efficient, robust, flexible, and easy to use.

Proposal

Develop MCT (Mission Control Terminal), a new web application with the following features and goals:

- Enable scheduling commands using built-in orbit prediction[5]
- Unify control of the Engineering and Flight satellite models
- Facilitate operator training and hardware in the loop (HIL) testing
- Ensure storage and logs of commands sent to the satellite
- Manage operator accounts and access control
- Improve ease of use and accessibility of mission control software

Non-Goals

- It is not a goal to design specific commands to operate the satellite.
- It is not a goal to store or visualize housekeeping or science data collected by the satellite.
- It is not a goal to scale the application beyond a single server.

[5] See SatNOGS (https://network.satnogs.org/observations/?norad=99172) for an example of a scheduler to receive beacons, but not send commands.

Requirements Specification

Mandatory Functional Requirements

ID	Name	Description
REQ-MCT-001	Service type	The MCT shall be a web service hosted on a Linux-based server, consisting of a graphical user interface accessible through a web browser ("web interface") and a TCP port to communicate commands with satellite systems ("command port").
REQ-MCT-002	User management and roles	The MCT shall manage a list of users and their associated data, including roles, as defined by the application administrator.
REQ-MCT-003	User creation and authentication	The MCT shall facilitate user creation and authentication through the OpenID Connect protocol.
REQ-MCT-004	Command port encoding	The MCT command port shall use the UTF-8 character encoding and exchange commands and data in plain text.
REQ-MCT-005	Command port user interface	The MCT shall present a console-like graphical interface allowing users to enter, select, and execute satellite commands while receiving text output from each command.
REQ-MCT-006	Multi-target operations	The MCT shall enable simultaneous planning and operation of multiple satellite systems ("targets"), including but not limited to the flight model and engineering qualification model.
REQ-MCT-007	Target identification	The MCT shall accept a target identification message from a client on the command port and associate all subsequent traffic on this connection to the target identified.
REQ-MCT-008	Command logging	The MCT shall log all commands sent to and received from each target with timestamps.
REQ-MCT-009	Command sequences	The MCT shall present a graphical user interface to create, edit, and delete automated command sequences.
REQ-MCT-010	Sequence timeout	Automated command sequences shall have a configurable timeout.
REQ-MCT-011	Sequence execution	The MCT shall be able to execute automated command sequences in the integrated command console.
REQ-MCT-012	Delayed commands	The MCT shall allow time delays, specified in milliseconds, to be placed between two commands in an automated sequence.
REQ-MCT-013	Sequence cancellation	The MCT shall have the option to cancel an automated command sequence during execution.
REQ-MCT-014	Permission list for target	The MCT shall store a list of allowed commands and automated sequences for each target, specified using regular expressions.
REQ-MCT-015	Permission list for user	The MCT shall store a list of allowed commands and automated sequences for each user, specified using regular expressions.
REQ-MCT-016	PL initial state	A permission list for a target or user shall start empty.
REQ-MCT-017	Permission list editing	An application administrator shall be able to edit the permission list for each target and user.

REQ-MCT-018	Command rejection	The MCT shall reject a user from executing or scheduling a command or sequence unless that command matches the permission list criteria for the originating user and the command target.
REQ-MCT-019	Command scheduling	The MCT shall be able to schedule commands and automated command sequences for future execution.
REQ-MCT-020	Scheduler interface	The MCT shall present a graphical user interface to create, edit, and delete scheduled commands and sequences.
REQ-MCT-021	Command timeline	The MCT shall present a graphical timeline to show past, present, and future commands for each satellite target, including scheduled and manually triggered commands.
REQ-MCT-022	Orbital prediction	The MCT shall use TLE (two-line element set) data periodically obtained from the internet to predict satellite overpasses and solar illumination cycles.
REQ-MCT-023	Satellites of interest	The MCT shall present a graphical user interface to select and edit the satellites of interest.
REQ-MCT-024	Current orbital state	The MCT shall display the current orbital state for satellites of interest, including azimuth, elevation, and solar illumination.
REQ-MCT-025	List of future overpasses	The MCT shall display a chronological list of future overpasses for satellites of interest over user-specified ground station coordinates.
REQ-MCT-026	Future pass detailed display	A detailed future overpass prediction shall include a polar plot, times for AOS and LOS, start and end azimuth, and max elevation.
REQ-MCT-027	Overpass orbit scheduling	The MCT shall be able to schedule commands when a satellite enters a user-specified elevation threshold over the ground station coordinates.
REQ-MCT-028	Solar orbit scheduling	The MCT shall be able to schedule commands when a satellite enters or exits solar illumination.

Optional Functional Requirements

ID	Name	Description
OPT-MCT-001	Failure criteria	The MCT may be able to define failure criteria for each command in an automated command sequence.
OPT-MCT-002	Early exit	The MCT may abort an automated command sequence if a step meets its failure criterion within the sequence.
OPT-MCT-003	Conditional execution	The MCT may be able to skip a step if the previous step meets its failure criterion within the automated command sequence.
OPT-MCT-004	Command reference	The MCT may have an editable integrated command reference to help users navigate satellite functionality.
OPT-MCT-005	Autocompletion	The MCT may have autocomplete suggestions in the integrated command console based on the command reference.
OPT-MCT-006	Search and filter	The MCT may have search and filter capabilities for the command timeline, scheduled commands, and satellites of interest.

OPT-MCT-007	Webhooks	The MCT may notify an external service of the results of scheduled commands via webhooks.
OPT-MCT-008	API	The MCT may expose an API to enable hardware in the loop testing of satellite software.

Non-Functional Requirements

ID	Name	Description
NFR-MCT-001	Commission date	The MCT should be ready to use by September 2024
NFR-MCT-002	Service lifetime	The MCT service should be operational until 2026, subject to extensions and improvements.
NFR-MCT-003	Containerization	The MCT should be a containerized application.
NFR-MCT-004	Databases	The MCT should use well-established database technologies to store application data and avoid text or plain files.
NFR-MCT-005	Backups	The MCT should allow an administrator to back up its data.
NFR-MCT-006	Application Logs	The MCT application should store logs to facilitate debugging.
NFR-MCT-007	Exceptions	The MCT application should handle internal exceptions.
NFR-MCT-008	Ease of use	The MCT web interface should be easy to use and accessible.
NFR-MCT-009	Ease of admin	The MCT application should be easy to administer.
NFR-MCT-010	Responsiveness	The MCT web interface should be responsive to user interactions.
NFR-MCT-011	Compatibility	The MCT web interface should be compatible with major browsers, operating systems and device types.
NFR-MCT-012	Look and feel	The MCT look and feel should be consistent with the branding guidelines of MIST and the PRESET mission.
NFR-MCT-013	Concurrent users	The MCT should allow multiple users to access the service simultaneously without incurring delays.
NFR-MCT-014	Service security	Network ports exposed by the MCT should be secured using TLS, SSH, or a similar cryptographic protocol.
NFR-MCT-015	User security	The MCT should implement session timeouts and other measures as necessary to protect the security of its user accounts.
NFR-MCT-016	Availability	The MCT web service should be available outside of scheduled maintenance and not be affected by the availability of any targets.
NFR-MCT-017	Resource usage	The MCT should not use excessive CPU, RAM, or disk space.
NFR-MCT-018	Version control	The MCT source code should be version-controlled and releases adhering to the Semantic Versioning 2.0 standard.
NFR-MCT-019	Code quality	The MCT source code should be well-organized, modular, and follow fundamental principles for software engineering
NFR-MCT-020	Code formatting	The MCT source code should use a code formatting tool.
NFR-MCT-021	Static analysis	The MCT source code should use a linter or other static code analysis tools to catch errors and faults before runtime.

NFR-MCT-022	Testing	The MCT source code should have test suites to identify regressions and verify compliance with requirements.
NFR-MCT-023	Documentation	The MCT project should contain documentation for its web interface and key components.
NFR-MCT-024	Staging environment	A staging environment should be available to verify functionality before deploying to the production environment.
NFR-MCT-025	Development environment	A local development environment for the MCT should be self-contained and easy to set up.
NFR-MCT-026	CI/CD process	The MCT should use a CI/CD process for software deployment and incorporate automated testing and static analysis.
NFR-MCT-027	Time accuracy	The MCT should ensure that the system date and time are accurate.
NFR-MCT-028	System updates	The MCT should be able to update its libraries and dependencies to improve security, performance, or functionality.