Status Server High-Level Overview

Tom Vermeulen

28 January 2016

This document is available on the Web at: http://software.cfht.hawaii.edu/statserv/HighLevelOverview/

Contents

1 Purpose

The purpose of this document is to provide a high-level overview of the Status Server.

2 Overview and Current Use

The Status Server serves as an open repository of status and state information available to any client within the CFHT network. Clients are able to view, create, update, remove or monitor information within the Status Server. In some respects, the Status Server could be thought of as a shared memory pool for multiple clients.

The Status Server is currently used at both CFHT and Pan-STARRS. TMT is currently evaluating HornetQ and Redis as Status Server type systems to handle event and telemetry information.

Information within the Status Server is stored as a hierarchy of name-value pairs. Only current status information is stored. At the time this document is written there are over 42,000 unique name-value pair status entries contained within the Status Server at CFHT. Each element in the Status Server is limited to a maximum data size of 255 characters.

Figure 1 shows a diagram of the systems within CFHT that produce status information as well as the clients that utilize this information. Before the Status Server existed at CFHT, status information was largely contained within each individual subsystem. If a client was interested in access to status it would be necessary to retrieve this information using customized solutions for each system. In 2002 it became clear that a general solution was necessary to consolidate status data into a central repository and the Status Server was designed and developed.

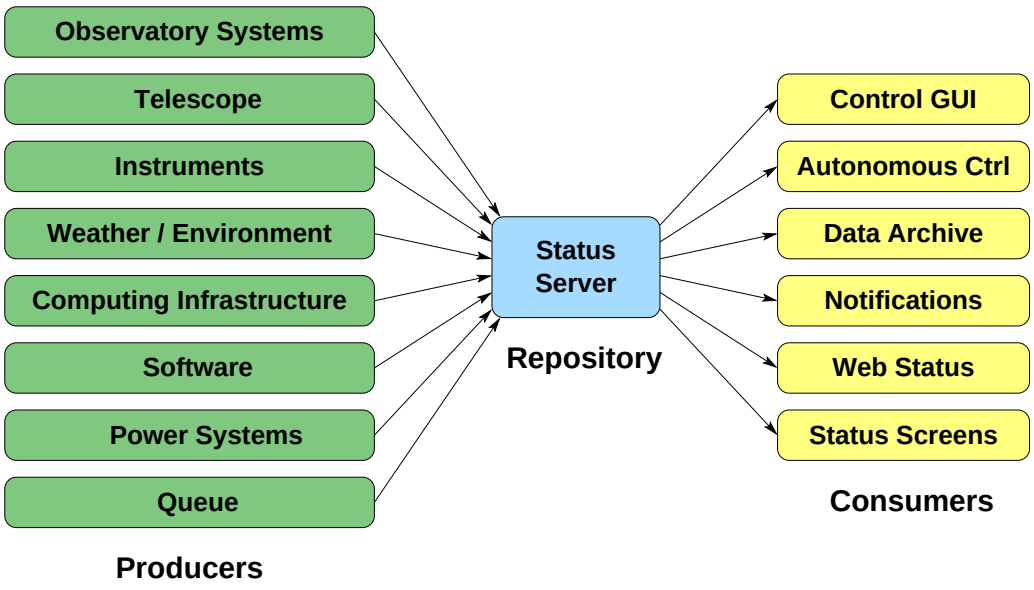

Figure 1: Status Server and Software Component Interactions

At this point virtually all the critical systems at CFHT have telemetry and status information published in the Status Server. Figure 1 shows the list of status producers. This can be event further broken down by category.

- **Observatory Systems** - Building Exhaust, Dome Drive, Dome Louvers, Dome Shutter, Dome Vents, Doors, Dry Air System, F8 Secondary, Fire Alarm System (summit & Waimea), Floor Cooling, Glycol System (instrument & building), Helium Compressors, Humidifier, Mirror Covers, Mirror Support, Particle Counter, Remote Lights, Water Supply, Windscreen.

- **Telescope** - Configuration, Dome Control, Ephemeris, Focus, Guiding, Physical Position, Power, Sky Pointing, Laser Traffic
- **Instruments** - All the relevant status and telemetry information for hardware associated with MegaPrime, WIRCam, ESPaDOnS, GRACES, and SITELLE.
- **Weather / Environment** - ASIVA, DataLogger, DIMM, MKAM, SkyProbe, Sonic Anemometer, Weather Tower, Webcams, and a host of additional environmental sensors.
- **Computing Infrastructure** - Host/Device Status, Computer Room Temperature/Humidity (summit & Waimea), Data Archive, Databases.
- **Software** - Configuration and status of active processes.
- **Power System** - HELCO, Generator, RPCs, UPS.
- **Queue**- Configuration and status.

Each of the producers noted earlier either directly publish information to the Status Server or helper applications were developed to query and publish to the Status Server on a periodic basis. An example of such a utility is the software program “abss”. The “abss” program is used to query Allen-Bradley PLCs and publish status information from registers contained within the PLCs to the Status Server.

Information is usually published to the Status Server at a pre-defined polling interval, depending on the source system, or it is published whenever the information is changed. In order to ensure that status information remains valid, it is possible to assign a lifetime to each piece of status. For example, the dome shutter status information is retrieved from the PLC at a 1 Hz frequency. Each of dome shutter status items have a lifetime of 60 seconds. A consumer of dome shutter status will receive a status value of “EXPIRED” for any object that hasn’t been updated in over 60 seconds.

On the other side of the Status Server in Figure 1 are the consumers of status information.

- **Autonomous Control** - With remote observing it is necessary to close the dome and put the observatory in a safe condition based on previously defined business rules. This process places monitors on the status items that make up the business rules for autonomous operations. If the value of any status items change, this process receives the new values and re-evaluates the business rules in order to determine whether any autonomous actions need to be taken.
- **Data Archive** - As noted previously, the Status Server only contains the last status of any published status objects. Items in the Status Server can be tagged for archiving. This process will place a monitor on any tagged items and store changes to those values in an archive.
- **Notifications** - Most of the cell phone and email notifications at CFHT are generated as a result of business rules that are evaluated based on status items within the Status Server. This process works similarly to the Autonomous Control process noted earlier.
- **OAP Control GUI** - The OAP Control GUI places monitors on a wide range of Status Server objects and the interface visualization is almost completely driven by the value of each Status Server object that is monitored.
- **Status Screens** - A number of different status screens have been developed that extract data from the Status Server.
- **Web Status** - The primary web portal into CFHT status and telemetry (http://statserv.cfht.hawaii.edu).

2.1 Directory Structure

Objects within the Status Server are grouped together in a “tree-like” fashion patterned after the UNIX file system. As a result, it is possible for a client to traverse and manipulate objects within the Status Server much like traversing a directory tree and manipulating files in a file system. Objects within the Status Server can be referenced either via a fully qualified directory path/object name combination, or a relative path-name combination. In order to manage relative path references, a current path is maintained for each client connection. Rules to determine whether a path-name combination is expressed as an absolute path or relative path are applied the same way they are in a UNIX file system. A visual example of the type of structure used to hold Status Server information is shown in figure 2. This example is a screen shot taken from the Status Server browsing client “gss”.

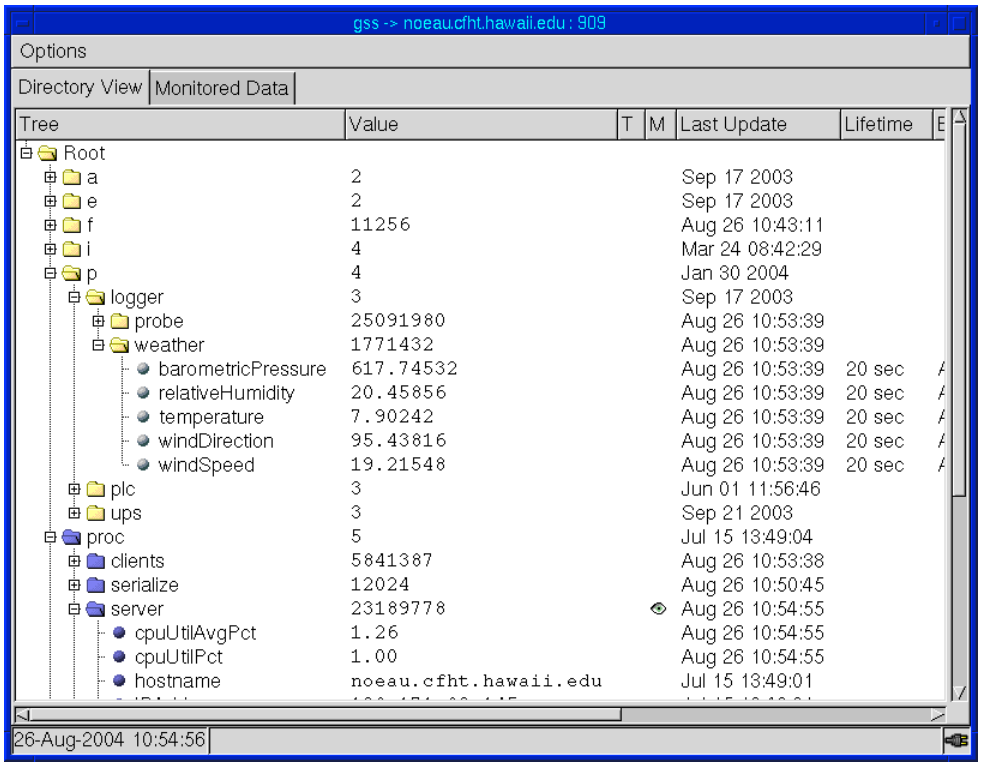

Figure 2: gss - Directory Hierarchy Screen shot

2.2 Performance

The Status Server is currently running on an older Linux machine and easily handles on average over 2,000 requests per second. Benchmark testing indicates that the Status Server could handle upward of 100,000 transactions per second on newer hardware. The Status Server is a single-threaded process so this performance is isolated to a single core within a CPU on the host computer.

2.3 Design Philosophy

The Status Server was designed to be simple and only focus on supporting the publishing, retrieval and monitoring of name-value pair information. Notifications, archiving, and other features are implemented within clients of the Status Server. As a result, it has been possible to create a small, robust, high-performance solution. The software itself hasn’t been modified since 2011 and has only been restarted when it has been necessary to reboot the host computer. This is important since the Status Server is an integral part of the operations and observing environment.

3 Document Change Log

Version	Date	Comments
1.0	Jan 28, 2016	First Release.