DATABASE-BACKED CONFIGURATION SERVICE

J. Mader, J. Johnson, K. Tsubota, W. M. Keck Observatory, Kamuela, HI 96743, USA

Abstract

Keck Observatory is in the midst of a major telescope control system upgrade [1]. This upgrade will include a new database-backed configuration service which will be used to manage the many aspects of the telescope that need to be configured for its control software. These parameters will need to remain persistent between IOC restarts (e.g. site parameters, control tuning, limit values). This paper will discuss this new configuration service, including its database schema, iocsh API, user interface and other provided features. The solution provides automatic time-stamping, a history of all database changes, the ability to snapshot and load different configurations and triggers to manage the integrity of the data collections. Configuration is based on a simple concept of controllers, components and their associated mapping. The solution also provides a failsafe mode that allows client IOCs to function if there is a problem with the database server. It will also discuss why this new service is preferred over the file-based configuration tools that have been used at Keck up to now.

INTRODUCTION

The current telescope control system at W. M. Keck Observatory is configured using a series of parameter files that are individually loaded into the control system. The files contain a listing of EPICS process variables (PV), values and, if required, unit conversion information. These files are read using a program called *pvload* (EPICS **p**rocess **v**ariable **load**er). *pvload* reads the parameter file and updates each PV in order as listed in the file.

A *pvsave* program also exists that allows for parameter files to be updated. A backup of the original file is created when *pvsave* is used. The original files can also be updated by hand, bypassing *pvsave*.

There is a lack of system context when using the parameter files. A given system can include a number of files, but those files can only be strung together by looking at the VxWorks startup files. This is compounded by the fact that there can be multiple backup versions for each parameter file. The backup versions for each file will be different making it difficult to recreate a given telescope configuration.

The configuration files are located on Solaris file systems while *pvload* runs in VxWorks. Files are first transferred across the network, read by *pvload* and then uploaded into the IOC. This results in the *pvload* utility being very slow.

In addition, periodic VxWorks flexscan errors have been experienced that force a reboot, possibly multiple times, to successfully load the configuration.

The new configuration service will provide the same functionality as the file-based system and will also have a number of improvements. The new service will allow

- Database-backed configuration
- Automatic time-stamping and provide a history of all database changes
- Sharing of common configuration items
- The ability to snapshot configurations
- Fail-safe operations
- A simple iocsh API and record support
- A user interface
- A systems level application view

THE CONFIGURATION SERVICE

Database

Configuration data will be maintained within a PostgreSQL database. Although only a single database is required to configure both of the Keck telescopes, two databases will be used to keep them as separate systems. The database will be comprised of

- Controller information describing the main telescope control subsystems (e.g. axe, pnt)
- Component information describing the breakdown within each controller (e.g. pntSlow, timDat)
- Information to describe how each component is mapped to a controller
- Information describing each channel (e.g. name, value) within each component
- Database change history

Figure 1 shows the configuration database design.

Channel names will use macros for the telescope and controller, which will allow a component to be shared between multiple controllers. These macros will be set

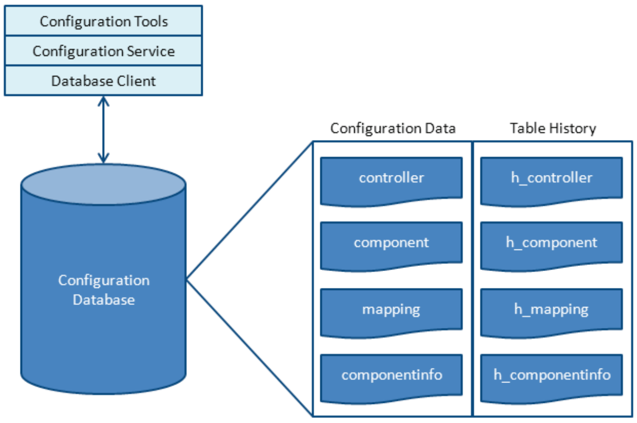
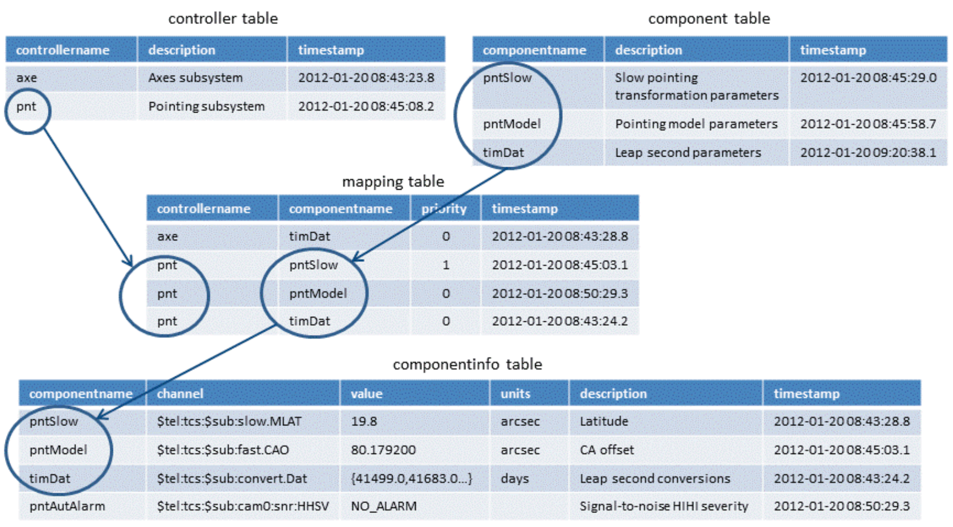


Figure 1: Telescope configuration solution.

ISBN 978-3-95450-139-7



controller table

controllername	description	timestamp
axe	Axes subsystem	2012-01-20 08:43:23.8
pnt	Pointing subsystem	2012-01-20 08:45:08.2

component table

componentname	description	timestamp
pntSlow	Slow pointing transformation parameters	2012-01-20 08:45:29.0
pntModel	Pointing model parameters	2012-01-20 08:45:58.7
timDat	Leap second parameters	2012-01-20 09:20:38.1

mapping table

controllername	componentname	priority	timestamp
axe	timDat	0	2012-01-20 08:43:28.8
pnt	pntSlow	1	2012-01-20 08:45:03.1
pnt	pntModel	0	2012-01-20 08:50:29.3
pnt	timDat	0	2012-01-20 08:43:24.2

componentinfo table

componentname	channel	value	units	description	timestamp
pntSlow	$tel:tcs:$sub:slow.MLAT	19.8	arcsec	Latitude	2012-01-20 08:43:28.8
pntModel	$tel:tcs:$sub:fast.CAO	80.179200	arcsec	CA offset	2012-01-20 08:45:03.1
timDat	$tel:tcs:$sub:convert.Dat	{41499.0,41683.0...}	days	Leap second conversions	2012-01-20 08:43:24.2
pntAutAlarm	$tel:tcs:$sub:cam0:snr:HHSV	NO_ALARM		Signal-to-noise HIHI severity	2012-01-20 08:50:29.3

Figure 2: Example database schema.

when the configuration is loaded by the controller. Channel values can be scalars, strings or arrays. Array values will be imported into the database using the format {val1, val2,…,valN}. Unit conversion will also be supported. For example, a value might be stored in the database in millimetres or arc-seconds and the service will convert those to meters or radians, respectively.

Figure 2 shows an example of the database schema. This example shows how the *pnt* controller consists of three components and how those components are mapped to their individual channels. Component sharing is also detailed, showing how the *timDat* component is shared between both the *axe* and *pnt* controllers.

Triggers

A trigger is a specification that the database should automatically execute a particular function whenever a certain type of operation is performed. The configuration service will use triggers to update all time-stamps and store a history of all database changes. In addition, there are triggers to keep the database from storing "unwanted" configuration data. For example, if all channels are deleted from a particular component, then the component and all mapping entries are also deleted.

Snapshots

A snapshot, or duplicate, of the database can be saved at any time. It will consist of exactly the same database schema and configuration data at the time the snapshot was taken. In addition to the tables being copied, all rules and triggers will also be copied. Any snapshot can be configured to be the active database.

The configuration service will use a simple database to configure itself, as shown in Fig. 3.

Fail-Safe Operations

There may be times at which a controller needs to load a configuration, but the database/database server is not up and running or there is a network glitch. In order for the controller to still be able to load the configuration, a private local copy of the configuration data will be stored on the local controller as text files. These files are created the first time an IOC makes a connection with the database and are updated automatically by the configuration service each time the IOC is started. In the case where a database connection is not made, the configuration items will be read and loaded from these local files. Figure 4 shows how these local files are used.

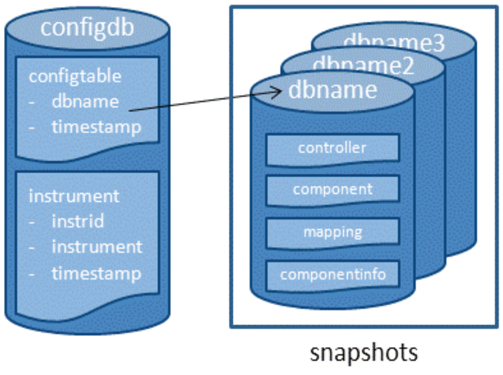


Figure 3: Snapshot usage and active database selection.

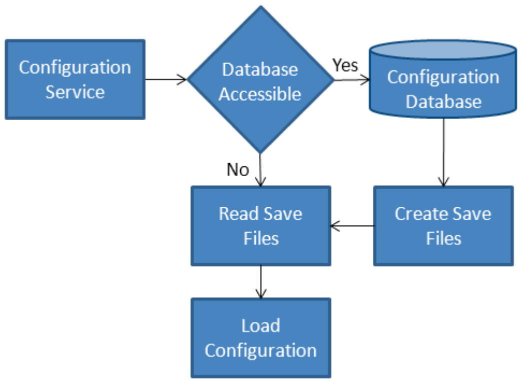
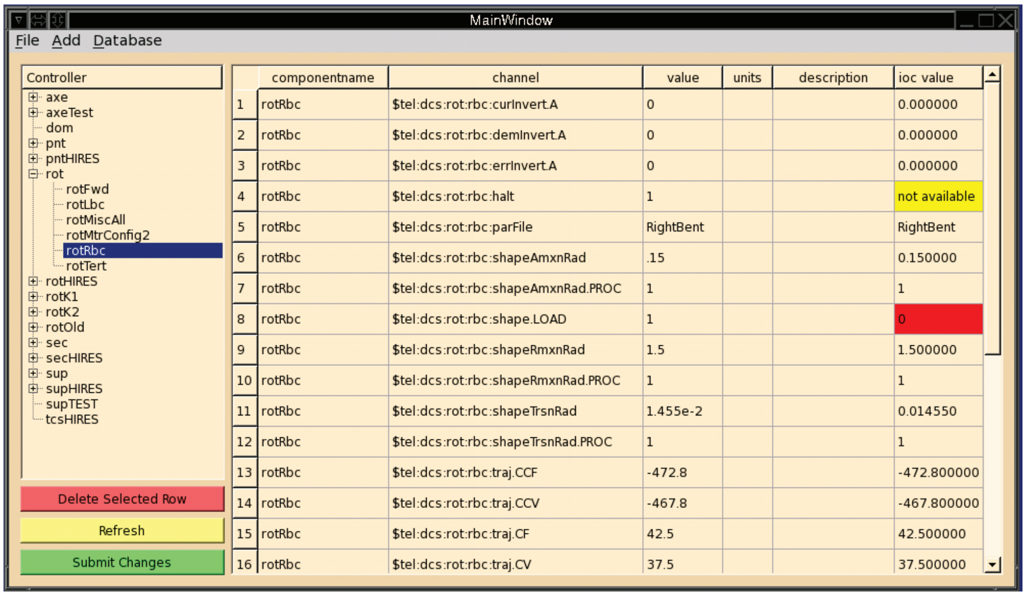


Figure 4: Database backup file usage.



	componentname	channel	value	units	description	ioc value
1	rotRbc	$tel:dcs:rot:rbc:curInvert.A	0			0.000000
2	rotRbc	$tel:dcs:rot:rbc:demInvert.A	0			0.000000
3	rotRbc	$tel:dcs:rot:rbc:errInvert.A	0			0.000000
4	rotRbc	$tel:dcs:rot:rbc:halt	1			not available
5	rotRbc	$tel:dcs:rot:rbc:parFile	RightBent			RightBent
6	rotRbc	$tel:dcs:rot:rbc:shapeAmxnRad	.15			0.150000
7	rotRbc	$tel:dcs:rot:rbc:shapeAmxnRad.PROC	1			1
8	rotRbc	$tel:dcs:rot:rbc:shape.LOAD	1			0
9	rotRbc	$tel:dcs:rot:rbc:shapeRmxnRad	1.5			1.500000
10	rotRbc	$tel:dcs:rot:rbc:shapeRmxnRad.PROC	1			1
11	rotRbc	$tel:dcs:rot:rbc:shapeTrsnRad	1.455e-2			0.014550
12	rotRbc	$tel:dcs:rot:rbc:shapeTrsnRad.PROC	1			1
13	rotRbc	$tel:dcs:rot:rbc:traj.CCF	-472.8			-472.800000
14	rotRbc	$tel:dcs:rot:rbc:traj.CCV	-467.8			-467.800000
15	rotRbc	$tel:dcs:rot:rbc:traj.CF	42.5			42.500000
16	rotRbc	$tel:dcs:rot:rbc:traj.CV	37.5			37.500000

Figure 5: Proto-type configuration UI.

User-Interface

A user interface will be provided to view and manipulate the configuration data, as shown in Fig. 5. The UI can be used to add, update or delete configuration items. In addition to each item's configuration data, the UI will also display the item's current value from the running controller IOC. When unit conversion is required, the IOC value will be converted back to the unit stored in the database. The UI will color code the cells when the IOC value differs from the database. A method to save an IOC value to the database will also be available.

IOC Commands

The configuration supports calls from within the IOC shell and dynamically during operations.

The IOC can load a single component or all components associated with its controller via the commands

- loadComponent("k#", "controller", "component")
- loadController("k#", "controller")

where "k#" is the telescope (k0, k1, k2). When used in PV names, the macros $tel and $sub will be substituted with the telescope and controller values. The configuration items also have a priority level which allows the items to be ordered when uploaded to the IOC.

There will be times when a configuration needs to be loaded dynamically during operations. A library is provided with a subroutine that can access the configuration service API. This subroutine can be called from within an *aSub* record. Each subsystem controller will have an aSub record that will be configured to update configuration items. A use case is when the instrument is changed and instrument specific parameters like the focal station, focus and pointing origin need to be updated. The *aSub* record will have three inputs

- The telescope (e.g. k1)
- The controller name (e.g. axe)
- The new instrument name

The *aSub* record monitors the instrument name and, when a change is detected, will call the configuration service's *loadInstrument* routine. The configuration service will then retrieve that instrument's configuration from the database.

CONCLUSION

The new configuration service is currently being tested at the Observatory with new development associated with the control system upgrade. The service has been running well and has been shown to be much faster and easier to use than the file-based system.

ACKNOWLEDGMENT

The W. M. Keck Observatory is operated as a scientific partnership among the California Institute of Technology, the University of California, and the National Aeronautics and Space Administration. The Observatory was made possible by the generous financial support of the W. M. Keck Foundation.

REFERENCES

[1] J. Johnson, K. Tsubota and J. Mader "Keck Telescope Control System Upgrade Project Status," MOCOAAB05, these proceedings.