APECS - The Atacama Pathfinder Experiment Control System

D. Muders¹, H. Hafok¹, F. Wyrowski¹, E. Polehampton¹,², A. Belloche¹, C. König¹, R. Schaaf¹,³, F. Schuller¹, J. Hatchell¹,⁴, and F. v.d.Tak¹,⁵

¹ Max-Planck-Institut für Radioastronomie, Auf dem Hügel 69, D-53121 Bonn, Germany
² Space Science and Technology Department, Rutherford Appleton Laboratory, Chilton, Didcot, Oxfordshire, OX11 0QX, UK
³ Argelander-Institut für Astronomie, Universität Bonn, Auf dem Hügel 71, D-53121 Bonn, Germany
⁴ School of Physics, University of Exeter, Stocker Road, Exeter EX4 3QL, UK
⁵ SRON National Institute for Space Research, Landleven 12, 9747 AD Groningen, The Netherlands

Received <date>/ Accepted <date>

ABSTRACT

APECS is the distributed control system of the new Atacama Pathfinder Experiment (APEX) telescope located on the Llano de Chajnantor at an altitude of 5100m in the Atacama desert in northern Chile. APECS is based on Atacama Large Millimeter Array (ALMA) software and employs a modern, object-oriented design using the Common Object Request Broker Architecture (CORBA) as the middleware. New generic device interfaces simplify adding instruments to the control system. The Python based observer command scripting language allows using many existing software libraries and facilitates creating more complex observing modes. A new self-descriptive raw data format (Multi-Beam FITS or MBFITS) has been defined to store the multi-beam, multi-frequency data. APECS provides an online pipeline for initial calibration, observer feedback and a quick-look display. APECS is being used for regular science observations in local and remote mode since August 2005.

Key words. Telescopes – Methods: data analysis – Methods: numerical – Astronomical data bases: miscellaneous

1. Introduction

APECS (Muders 2005) is the distributed telescope control system of the new APEX¹ submillimeter telescope (Güsten et al. this volume) in the Chilean Atacama desert. APECS is based on the framework of the ALMA Common Software (ACS, Raffi, Chiozzi & Glendenning 2001) and the ALMA Test Interferometer Control Software (TICS, Glendenning et al. 2001).

ACS uses the Common Object Request Broker Architecture (CORBA, OMG 1999) to provide the middleware communication layer to interface hardware devices and software components. The container-component model allows for automatic component property monitoring. ACS uses a special configuration database to store object naming hierarchies. TICS provides the basic antenna control including astronomical coordinate system handling and observing pattern elements. A mechanism to store the monitoring data into a database is included in TICS.

APECS itself was developed at the Max-Planck-Institut für Radioastronomie. We began the development by defining the generic instrument and device interfaces (Muders et al. 2002, Muders 2006) and their CORBA object code (cf. section 3). In parallel, the new Multi-Beam FITS or MBFITS raw data format was defined (cf. section 4). Subsequently, we developed the observer level which provides a Python (Rossum & Drake Jr. 2001) based scripting language for observing, the central Observing Engine (cf. section 5) to coordinate all devices and processes, the MBFITS Raw Data Writer (cf. section 6) and the Data Calibrator (cf. section 7) to automatically perform the atmospheric corrections and provide CLASS (The GILDAS software) data with a $T_A^*$ temperature scale.

We also developed generic graphical monitoring tools to view any system property and its alarm states. The corresponding Graphical User Interfaces (GUIs) can be constructed by simply drawing widgets using an existing GUI construction tool and calling them according to the monitor point names. All observations are automatically logged including important scan details and current observing and environmental parameters. In addition, the observers may enter their own comments.

Overall, APECS now provides a fully featured single-dish telescope observing system that supports the standard observing modes like calibration, pointing, focusing, on-offs, pointed rasters and on-the-fly mapping. Special modes for bolometers (e.g. spiral patterns) and heterodyne array receivers (e.g. array derotation) will be added when those instruments are delivered to APEX.

Send offprint requests to: D. Muders, dmuders@mpifr-bonn.mpg.de

¹ APEX is a collaboration between the Max-Planck-Institut für Radioastronomie, the European Southern Observatory, and the Onsala Space Observatory
2. APECS Design

APECS is designed as an object-oriented pipeline system (see fig. 1). Observations are set up using so-called Scan Objects which contain the full description of the next observation, i.e. the instrument setup details, target coordinate information and the desired observing patterns. The Scan Objects are passed to the Observing Engine which sets up all necessary devices and controls the data acquisition. It passes the Scan Object on to the raw data writer which extracts necessary information for the primary data products.

On the high level almost all APECS applications are written in Python since the complex bookkeeping can be easily mapped into a flexible object structure. Heavy duty numerical calculations are performed in compiled libraries using Python wrapper interfaces.

3. Generic Instrument Interfaces

One of the most important initial steps in a software development project is to define the structure of packages and their interfaces. In a telescope control system there is an additional need for interfaces to all the hardware devices that are being used for the observations. We therefore began to collect information about typical setups at other radio observatories to eventually define a set of common device properties and methods (Muders et al. 2002).

The important design decision was to require that instruments of the same kind (e.g. heterodyne receivers, spectral backends, etc.) must all use the same high-level interface. This simplifies the setup for the high-level observing software enormously because one merely adds a new instrument name without having to worry about adding new features at that level.

Using the collection of instrument properties, we defined a set of CORBA Interface Definition Language (IDL) files which are the building blocks out of which one can construct software representations of even very complex devices (Muders 2006). The actual detailed setup of, for example, a receiver differs, of course, from one instrument to another. Some sub-devices exist for some instruments but not for others. This is represented by a naming hierarchy of CORBA Distributed Objects (DOs) for the hardware components in the receiver. However, the top-level interface is identical for all receivers.

We reuse the same IDL and even the same DO library many times. We do not even have to compile any new code unless one adds something that is not covered by the existing IDLs. The CORBA side of the interfaces is automatically generated from the CORBA IDL interface files using a code generator developed at the U Bochum (R. Lemke priv. comm.).

The communication between the CORBA components and the embedded hardware control systems is implemented via acknowledged clear text commands sent through UDP sockets to avoid known problems with broken TCP connections. The commands follow the SCPI (Standard Commands for Programmable Instrumentation, SCPI Consortium) standard (Hafok, Muders & Olberg 2006). The SCPI commands are automatically derived from the hierarchical CORBA device names and the property and method names (e.g.: APEX:HET460:L02:MULTI1:backShort2).

We have developed generic Python simulators which can replace the embedded systems to be able to run APECS for tests and developments without the need for real hardware. Together with the TICS telescope simulator one can run the full APEX simulator on a normal Linux PC or even a laptop.

4. MBFITS Raw Data Format

In addition to the hardware interfaces, one also needs to determine the data product interfaces early on. There was a lack of modern raw data data format descriptions when the APECS developments began. We therefore defined a new data format called MBFITS (Multi-Beam FITS, Muders, Polehampton & Hatchell 2005) to store the raw APEX data.

The MBFITS format was derived structurally from the ALMA-TI FITS (Lucas & Glendenning 2001) raw data format, although a number of changes had to be made to accommodate the special needs of the APEX and also the IRAM 30m and Effelsberg 100m telescopes where MBFITS is being used.

The MBFITS format uses the FITS standard (Wells, Greisen & Harten 1981) and the World Coordinate System (WCS, Greisen & Calabretta 2002) representation. MBFITS is based on the scan-subscan-integration scheme used by ALMA-TI fits and retains many of its keywords. However, due to the changes in structure and additional keywords needed to accommodate single-dish configurations, particularly multiple beam observing and multiple frontend/backend combinations, the MBFITS format can now be considered to be an independent format.

For each level of time granularity (scan, subscan, backend integration) there are FITS binary tables to store the corresponding data. A special monitoring table allows to record important instrument parameters in parallel to the backend data stream for later analysis.

For efficiency reasons, the MBFITS data are written to a file system directory structure where each binary table is stored.
in a separate file. The files are connected to each other via a grouping table using the FITS Hierarchical Group standard (Jennings et al. 1997).

5. Observing Engine

All APECS observations are triggered by sending Python Entity Objects, the so called Scan Objects, to the Observing Engine process. The Scan Objects can currently be constructed using a Command Line Interface (CLI). Observer commands have been grouped according to functionality areas into catalog, target, instrument, calibration, pattern and switch mode setups.

The command setup makes use of Python’s object-orientedness by re-using the same classes for all instruments of a given kind. User parameters are usually taken as next default to allow user environment customisation. We intentionally implemented first a CLI to facilitate user scripting which was a strong requirement. The future GUI will use the existing CLI commands.

The Scan Objects are then passed via CORBA to the Observing Engine which plays the central role in coordinating all devices to actually perform the observations. The Observing Engine interprets the Scan Objects and sets up all devices and controls the telescope movements, data taking and initial calibrations. For each step there are detailed logging messages that are sent to the observer using CORBA. Potential hardware problems can thus be easily detected.

6. Raw Data Writer

The core of the raw data writer’s object-oriented design is a system of multi-threaded, internal pipelines (see fig. 2) laid out according to the design pattern Pipes and Filters (Buschmann et al. 1996). Each backend that is selected for a scan is associated with a Backend Pipeline that receives the backend data via a TCP stream, processes it, and writes it to the corresponding binary tables. The binary table structure is stored in an XML template.

The Monitor Pipeline receives telescope (and in the future wobbler) position data from the corresponding CORBA notification channels, provides the data to the backend pipelines for interpolation, and writes it to the file. In addition, it collects and writes user-defined monitor points from different devices.

For the realization of the pipeline system, a framework was developed that allows for a consistent treatment of all aspects of the setup, operation, and error handling. In particular, it allows the reconfiguration of the pipelines at runtime. This is necessary since the backend setup may vary from one scan to another.

The data associated parameters are interpolated to the backend time stamps using trapezoid integrations in Cartesian coordinates. The final offsets are computed using the mean angular coordinates and applying the radio projection.

7. Data Calibrator

The APEX Data Calibrator (Polehampton 2005) provides initial reduction, calibration and display of data for both heterodyne and bolometer receivers. This includes feedback to the observing system for the basic pointing and focus observing modes via CORBA, as well as producing final calibrated spectra, which are written to disk in the CLASS (The GILDAS software) format.

The basic data structures used in the reduction process are Python Entity Objects which contain the data from a single subscan. They feature methods to perform standard operations on their contents including overloaded operators to allow arithmetic on the entire entity.

The reduction proceeds on a subscan-by-subscan basis, retaining entities that are required for further processing (e.g. references, calibrations). Heterodyne calibration is carried out using an extended version of the standard radio astronomy chopper wheel technique on sky, hot and cold loads.

Internal calculations with the ATM atmospheric model library (Pardo, Cernicharo & Serabyn 2001) are carried out using the full Planck equation and do not involve the Rayleigh-Jeans approximation. Bolometer data online reduction are carried out using libraries of The Bolometer Data Analysis Project (BoA, Bertoldi et al.).

The resulting data are stored in modified entities - containing the reduced data with additional calibration attributes. For heterodyne data, these are then written to the CLASS format.
Feedback to the telescope system is provided via a CORBA interface. This allows pointing and focus corrections to be passed back into the system and implemented on the APECS command line by the observer.

An offline command line interface is also provided for heterodyne reduction. It uses exactly the same methods as for the online system. Offline bolometer reduction is provided directly by BoA.

The APEX Data Calibrator uses a highly modular, object-oriented design, allowing the flexibility required to easily update the system for future developments in instrumentation, observing modes or calibration techniques.

8. Deployment

The APECS software is deployed at three main locations: the telescope itself, the control room at 5100m altitude on Chajnantor and the control room in the APEX base camp in Señior near San Pedro de Atacama which is connected to the mountain via a 32 Mbps microwave link.

Three main servers provide the CORBA services, the CORBA containers and their objects and the MBFITS writer and the calibrator. A number of client stations are used for local or remote observations from the high site or the base. Remote observing from the partner institutes in Europe is possible using either Virtual Network Computing (VNC) or Virtual Private Network (VPN) technologies.

9. Conclusion

APECS is a modern, object-oriented telescope control system based on the ALMA software framework using CORBA as the middleware. Its generic interface approach greatly simplifies adding new instruments. The automatic monitoring of instrument properties facilitates debugging hardware problems. The user-friendly, Python-based scripting language that is employed for observations and data calibration allows using many existing software libraries, thus saving much development time. New observing modes can be easily added at the scripting level. The new MBFITS raw data format provides a self-descriptive, self-contained way of storing all data that are necessary for further processing. The online data processing pipeline provides calibrated spectra and feedback for typical calibration scans. Overall, APECS is now a mature telescope control system with the potential for future extensions.

Acknowledgements. We would like to thank the ALMA software developers who helped us a lot to understand their software so that we could begin our own developments within that framework.

References


Buschmann, F. et al., 1996, Pattern-Oriented Software Architecture: A System of Patterns, Wiley
Güst et al., The Atacama Pathfinder EXperiment Telescope, 2006, this volume
Hafok, H., Muders, D. & Olberg, M., 2006, APEX SCPI Socket Command Syntax and Backend Data Stream Format, APEX Report APEX-MPI-IFD-0005, Rev. 1.0
Muders, D., 2006, APEX Instruments Generic CORBA IDL Interfaces, APEX Report APEX-MPI-IFD-0004, Rev. 1.5
Muders, D., Polehampton, E. & Hatchell, J., 2005, Multi-beam FITS Raw Data Format, APEX Report APEX-MPI-IFD-0002, Rev. 1.57
Raffi, G., Chiozzi, G. & Glendenning, B., 2001, The ALMA Common Software (ACS) as a basis for a distributed software development, ADASS XI
The GILDAS data reduction software, http://www.iram.fr/IRAMFR/GILDAS

\(^2\) Operating hard disks at such a high altitude is technologically challenging due to the low air pressure that can lead to head crashes. APECS uses specially selected SCSI disks.