cdev 1.5 Release Notes

1. Overview

The cdev (control device) C++ library is designed to provide a standard application programming interface (API) to one or more underlying packages, typically control system interfaces. The implementation is in 2 layers: the uppermost layer is used directly by an application, and provides an abstraction of the underlying package as well as many general purpose features. The second layer (service layer) provides the interface to one or more underlying packages, and is implemented as loadable libraries.

cdev was initially developed at CEBAF with input from the EPICS collaboration, and is now available for anyone to take and use or improve. We welcome all comments, suggestions, bug reports, and (especially) bug fixes, including fixes for new platforms. There are many planned enhancements, and would welcome volunteers to contribute to this project. See below for a summary of new and planned features.

You may join a cdev discussion list by sending the message

subscribe cdev

to the address mailserv@cebaf.gov. (Send the message HELP to the same address for information on how to use the mail server.) The cdev web page is

http://www.cebaf.gov/cdev/

cdev 1.4 is the third publically released version of cdev.

2. Documentation

The documentation for cdev includes the following:

- *Introduction to cdev* -- this is an update of *The cdev User's Guide*, complete with more examples. It covers all the basics, and is the best place to start.
- *A cdev Tutorial* -- this document goes through how to use each of the classes in cdev, including all public methods; extensive examples are given.
- *cdev Reference Manual* -- documents each class in cdev in alphabetical order
- *cdev Administrator's Guide* -- this is for system administrators, and those intending to port cdev to new platforms
- *Service Developer's Guide* -- this documents a couple of additional classes needed to create a custom service, and goes through a lengthly example demoService.
- *cdevCollection Device Requirements and Design* -- This document illustrates the structure and usage of the cdevCollection device.
- *cdev C Binding* -- documents the C wrapper routines for cdev

This documentation is also available on the web site.

3. Installation

Sources are available in compressed tar format, and contain makefiles to build the libraries and utilities. These have been tested on the HP, SunOS, and SGI platforms to execute correctly, and SLAC has ported this version to VAX/VMS. Fixes have also been incorporated from an earlier port to IBM/AIX (at CERN). As configured, cdev will build support for the EPICS channel access service.

The Makefiles have been extensively reworked since the 1.1 release, and should build successfully on more platforms.

For further information, see the README file in the topmost directory, and the *cdev Administrator's Guide* in the /doc sub-directory.

Support for site specific control systems must be added by the implementer. See the *Service Developer's Guide* for help in getting started interfacing cdev to a new control system. In addition, there are a number of example services included with this release.

4. Changes from 1.1 Version

A number of changes have been made to make the source code compile correctly on a larger variety of compilers, including VAX C++ and g++. In addition, cdev now produces more error and warning messages through its error logging routines. The following is a list of the more visible feature changes:

1. The C binding is included in the release (previously released as an add-on to 1.1).
2. The device definitions (ddl file) may now be stored and loaded in a binary format, which is about 20 times faster to load compared to the ascii format. A new utility program, cddl, compiles the ddl from ascii to binary. At run-time, the file format of the ddl file is detected automatically so that either ascii or binary may be used.
3. Shared objects are now loaded from a subdirectory of the directory pointed to by CDEVSHOBJ, where the subdirectory name is the name of the version of cdev against which the application is running. This makes it possible to run different applications against different versions of cdev. If the subdirectory does not contain the needed object, it is loaded from CDEVSHOBJ as before.
4. cdevFdChangedCallback added to cdevSystem, allowing an application to be notified when file descriptors are opened or closed by cdev services.
5. cdevData now supports a cast to char* as well as operator= overloading.
6. cdevGroup now supports a deferred execution / re-execution mode: if a group is in this mode, operations are remembered but not executed. Saved operations may later be executed as a group, and re-executed without re-learning. Only asynchronous operations are remembered.
7. A few bugs in the channel access service have been fixed. Monitoring of status without monitoring value now works correctly.
8. The cdev/src directory now contains an extensions sub-directory. Extensions include the tcl interface, and the ACE (Adaptive Communication Environment) based client/server package used for building new cdev servers and services capable of passing entire cdevData objects over the network.

9. The cdev/examples directory now contains 2 additional example services, one implementing compound cdev devices, and the other (demoService) is a trivial service useful as a starting point for read/write services.

5. Changes from 1.3 Version

Several minor changes have been introduced to the system in order to improve performance for existing operations.

1. The *operator* == and *operator !*= methods have been added to the cdevData class. These methods allow the caller to perform a deep comparison of the contents of two cdevData objects.
2. A *cdevTagTableCallback* class has been added to the cdevTagTable.h file. This class has a pure virtual callback method that (when registered with the *cdevData* class) will receive a callback each time a new tag is added to the cdev global tag table.
3. The *addTagCallback* method has been added to the *cdevData* class. This method is used by the caller to install a *cdevTagTableCallback* object that will be executed each time a new tag is added to the global tag table.
4. The *delTagCallback* method has been added to the *cdevData* class. This method removes a *cdevTagTableCallback* object that was previously installed using the *cdevData::addTagCallback* method.
5. The *readTagTable* method has been added to the *cdevData* class. This method allows the caller to retrieve a list of integer tags and their corresponding character strings that are stored in the global tag table.
6. The *CDEVTAGTABLE* environment variable is used to indicate the location of a file based list of tags that will be automatically loaded when a cdev application is started. If the variable is not specified the application will attempt to load the list from */usr/local/lib/cdevTagTable*. If this file does not exist, then the default list of tags will be used.
7. Channel Access Service (caService) modified to call the callback function immediately when a second monitor is installed on an channel that is already monitored.

6. Changes from 1.4 Version

Major additions have been made to the system to support new functionality.

1. The cdevCollection device and cdevCollectionRequest object have been added to the CDEV class library. These mechanisms provide the capability to send a single message to a group of devices that may span multiple services.
2. Modifications have been made to the cdevDevice, cdevRequestObject, and cdevService class to support the new cdevCollection hierarchy.
3. Because of binary incompatibilities between the different versions of the CDEV DDL file. The cdevSystem will now append the version number of the current version of CDEV (-1.5) to the name specified in th CDEVDDL environment variable before attempting to load the file.
4. Modifications have been made to the CDEV Generic Server Engine to allow it to manage user defined packet types using the same protocol definition.
5. ACE Version 4.0.32 is now being distributed with CDEV.

6. The *cdevRequestObject::attachRef(cdevDevice &, char *message)* method has been made protected and is no longer part of the public interface. This method was always intended to be used internally by cdev. Developers should use *cdevRequestObject::attachRef(char * device, char * message)* to obtain a cdevRequestObject reference.
7. The cdevCallback class has been expanded to provide a fireCallback method that has an additional parameter; partialTransaction. If the service is delivering a message that is one of many that might be generated by the request (such as monitorOn), the partialTransaction flag should be set to non-zero. If the reply is in response to a monotonic operation (such as “get” or “set”) or if it is the last response of a multi-part reply, the partialTransaction flag should be set to zero.
8. The isTransactionDone method has been added to the cdevCallback class to allow the callback function to identify if the message is a monotonic response or one of many.

7. Problems and Restrictions (general)

Restrictions:

- cdev is not yet thread safe, and may not be safe against instantiating more that one cdevSystem object.
- Device names can only be added to or removed from the cdevCollection device until it has been used ***once***. After a cdevCollectionRequest object has been generated from a specific cdevCollection device, the device list cannot be modified.
- cdevCollection devices do not support the messages “monitorOn” and “monitorOff”. Only monotonic operations are currently supported by the cdevCollection device.
- cdevCollection definitions must be the last things defined in the CDEV DDL file. This is because the constituent devices must have already been defined when the cdevCollection is defined.
- When the data from a cdevDevice operation is used to construct the output cdevData object for a cdevCollection operation, only data that is scalar in nature will be used. If a cdevDevice returns tagged data items that are arrays in its results, those data items will not be included in the result.
- cdevCollection objects must be comprised of regular cdevDevice objects only. Because the cdevCollection does not support cdevDevices that return arrays of data in their result, a cdevCollection can not contain another cdevCollection.
- If a cdevCollection consists of devices from multiple services, the collection does not guarantee the order in which the cdevDevice’s requests will be serviced. Because services perform at varying speeds, a faster service may process its devices requests sooner than another service.

8. Channel Access Service Features and Restrictions

EPICS / Channel Access version 3.12 is recommended. The following comments from the 1.1 release still apply:

Major features:

- Supports synchronous and asynchronous send’s.
- Fetches data in native type (type conversion done on client and not server).

- Provides a default name service so that if a device is not defined, then an EPICS channel access connection is attempted with the record name set equal to the device name, and the field name set equal to the attribute name.

Restrictions:

- Only supports set/get/monitorOn/monitorOff verbs and the pv and readonly service data tags. Support for arbitrary messages with default data has again been postponed.
- Discards all exception callbacks from channel access in this version.
- Channel access security is not supported in this release.
- sendCallback does not support the same timeout feature as send.
- caService fails if send is called from within a callback function (either asynchronous methods are OK: sendNoBlock and sendCallback)

Compilation options:

- _CA_SYNC_CONN = perform connections synchronously, waiting up to 4 seconds.
- _CDEV_DEBUG = print verbose messages
- _EPICS_3_12 = if not defined, use 3.11 calls instead.

9. Customizing for a New System

To customize cdev for a new control system, a new cdev service must be written. Two services are provided with the release: caService for EPICS channel access, and skService, a skeleton service. The ca (channel access) service is a complex example of how to interact with a control system which supports asychronous I/O. The skeleton service simply shows the minimal interface a service must provide, and can be used as a starting point for building a new service. A third example is contained in the Service Developer's Guide, and is intermediate in complexity.

Still Planned:

- a simple service for which you need only provide synchronous read and write routines. This will speed up integrating a simple synchronous RPC or memory image based system.

10. Planned Enhancements

There are many enhancements planned for cdev, and releases will probably come out every 3 months for a while. The following is a list of topics planned for summer of 1996. Work on all of these topics is underway.

1. Collection devices. Created dynamically or statically, each device would contain a list of other devices. Messages would be repeated to all contained devices, and read results would be collected together and returned to the caller as a vector. Implementation will support passing the list down to the service instead of iterating in the cdev layer so that list based services (such as SLAC's) can optimize performance.
2. Service + Server based upon passing cdevData objects between client and server. This allows complex query/reply type operations. A prototype is already in operation at CEBAF, although the protocol is likely to change somewhat.
3. Virtual device wrapper around the correlation measurement engine (see below).

The following list has been put on the back burner, but still of interest:

4. “Multiknob” devices. Write operations would be applied to a vector of devices with weighting factors which may be constant, or a simple functional form, or derived by a call to another process. Intended for tuning machines.
5. Indirect devices. The name of the actual device can be changed on the fly, and all connections to the old device will be automatically dropped and new connections established. Useful for changing generic displays from one device to another.
6. tcl devices. Devices implemented as tcl scripts, which may call other cdev devices (or anything else).
7. Connection to an object oriented database.

11. cdev Applications

The following applications are available as cdev extensions from CEBAF:

1. Correlation package, modeled after SLAC’s correlation plot utility. Consists of a correlation engine class library plus an application (XACT) which uses it.
2. Multiknob utility (like #4 above, with a GUI interface). This is based upon a prototype of multiknob devices, and is configured by ascii files.
3. ARTEMIS Model Server.
4. CEBAF Lock Server.

Finally, the following EPICS tools will be ported to cdev or re-written using cdev in the next 12 months (ports to be distributed with EPICS, not cdev, due to licensing constraints):

1. Alarm Handler (specialized GUI tool). Porting now underway at APS.
2. General operator interface (GUI), either dm or medm or both.
3. Archiver (new, to be distributed with cdev).
4. Save / Restore tool (ditto).

Other applications will soon become available from other sites, and the most current information will be available from the cdev web site, http://www.cebaf.gov/cdev.