QT EPICS DEVELOPMENT FRAMEWORK*

A. Rhyder#, A. Owen, G Jackson. Australian Synchrotron

Abstract

QCa is a layered software framework based on Qt for accessing EPICS data using Channel Access on a range of platforms. It is used on several beamlines at the Australian Synchrotron. The QCa framework provides object oriented C++ access to control systems using EPICS (Experimental Physics and Industrial Control System). It is based on Qt, a widely used cross-platform application development framework. GUI or console based applications can be written that use QCa at several levels. QCa includes Qt plugin libraries, EPICS aware widgets, data formatting classes, and classes for accessing raw EPICS data in a Qt friendly way. QCa also includes an application for displaying forms produced by the Qt development tool 'Designer'. Using this application a complete EPICS GUI system can be generated without writing any code. A GUI system produced in this way can interact with existing EPICS display tools such as EDM. QCa handles much of the complexities of Channel Access including initiating and managing a channel. Applications using QCa can interact with Channel Access using Qt based classes and data types. Channel Access updates are delivered using Qt's signals and slots mechanism.

INTRODUCTION

Channel Access is described as 'one of the core components of an EPICS system. It is the software component that that allows a Channel Access client application to access control-system data which may be located on different hosts throughout a network' [1]

While CA is the default means to access EPICS data, its use is not trivial. A significant understanding of how CA works is required to execute the steps required to read or write data. The complexity of setting up and terminating CA requests leaves room for error. Also, CA uses a C programming interface and so does not make use of object oriented programming techniques.

Qt is a cross-platform application and UI framework. It includes a C++ class library and a cross-platform IDE.

The QCa framework provides a Qt based C++ framework for easy CA access to EPICS data.

It provides access to EPICS data at several levels from programmatic reading and writing of data, EPICS aware widgets for developing GUI based applications through to EPICS aware Qt plugins such as push buttons, sliders, and text widgets. When these plugins are used within the Qt form development tool 'designer' EPICS GUIs can be developed without the need for any code development.

QCA FRAMEWORK HIERARCHY OVERVIEW

The QCa framework is designed to allow access to CA data in the most appropriate form. The framework is based on a hierarchy of classes as shown in Table 1. This hierarchy is open at all levels to the developer. Appropriate use of the hierarchy is shown in Table 1.

Table 1: QCa framework hierarchy

Type of access to CA data.	**Functionality**	**Main classes**
C++ access to the CA library.	Provides convenient C++ access to the CA library.	CaObject
Qt based access to CA.	Hides CA specific functionality. Adds Qt functionality such as signals and slots.	QCaObject
Data type independent access.	Hides EPICS data types, providing read and write conversions where required.	QCaInteger QCaString QCaFloating
EPICS aware graphical widgets.	Implements graphical Qt based widgets that provide access to EPICS data.	QCaLabel QCaLineEdit QCaPushButton QCaShape QCaSlider QCaSpinBox QCaComboBox QCaPlot
EPICS aware graphical Qt plugins.	Adds Qt plugin interfaces to EPICS aware widgets.	QCaLabelPlugin QCaLineEditPlugin QCaPushButtonPlugin QCaShapePlugin QCaSliderPlugin QCaSpinBoxPlugin QCaComboBoxPlugin QCaPlotPlugin
GUI support widgets	Implements Qt based widgets that support control system GUIs. These widgets do not access the CA library.	AsGuiForm GuiPushButton CmdPushButton Link

*Work supported by the Australian Synchrotron
#andrew.rhyder@synchrotron.org.au

C++ ACCESS TO THE CA LIBRARY

The CaObject base class provides a C++ wrapper around the CA library. While available to the developer, it was written mainly to provide a level of abstraction within the Qt based QCaObject class. It is recommended to be used where a Qt framework is not available.

QT BASED ACCESS TO CA

The QcaObject class provides full access to EPICS data while hiding most CA specific functionality such as link status, connections and channels.

The QcaObject class adds Qt functionality. Data can be written using a Qt slot and Qt signals are available for data and status information as required. Qt data types are used to represent all EPICS data.

The data in the update signals may be of any type and is represented by a Qt variant.

DATA TYPE INDEPENDENT ACCESS

The classes QCaInteger, QCaString, and QCaFloating are based on QCaObject and interpret all data as integers, strings, and floating point numbers respectively. They are used to provide access to EPICS data in a known format regardless of the actual data type of the EPICS data. For example, string data is always required for a text label regardless of the underlying EPICS data type. While some conversions are unlikely to be of much practical use, all conversions are permitted.

EPICS AWARE GRAPHICAL WIDGETS

The classes QCaLabel, QCaLineEdit, QCaPushButton, QCaShape, QCaSlider, QCaSpinBox, QCaComboBox, and QCaPlot allow an application to add graphical objects to a user interface that are EPICS aware. That is, they interact directly with EPICS data. The application sets up the EPICS process variable name and other parameters that define how the widget interacts with EPICS data. The application does not have to handle EPICS data or any aspect of the CA interface.

The application may supply the EPICS aware widgets with an object that the widgets can send Qt signals to, including error and status messages signals.

EPICS AWARE GRAPHICAL QT PLUGINS

The classes QCaLabelPlugin, QCaLineEditPlugin, QCaPushButtonPlugin, QCaShapePlugin, QCaSliderPlugin, QCaSpinBoxPlugin, QCaComboBoxPlugin, and QCaPlotPlugin are EPICS aware widgets with a Qt plugin interface.

These plugins can be used by any Qt application that can load plugins.

They are loaded into the Qt GUI design tool 'Designer' which can be used to generate GUI description files that include EPICS aware widgets. These files can be loaded at run time by any application code, or used as source for any application. One application that loads these files at run time is AsGui, an MEDM/EDM replacement. A feature of these plugins is that they are active at design time.

GUI SUPPORT WIDGETS

The classes AsGuiForm, GuiPushButton, CmdPushButton and Link implement Qt based widgets that support the development of EPICS control system GUIs. They are not EPICS aware widgets.

The AsGuiForm class can contain any Qt based widgets, including the QCa framework's widgets. It is used as the scroll area in the AsGui application and can be used to create sub forms when developing control system GUIs in 'designer'.

The GuiPushButton class is used to launch new GUIs.

The CmdPushButton class is used to execute any command. Typically it would be used within a GUI to perform an action on the local machine, such as launch another application, or interact with an MEDM session.

The Link class provides a generic mechanism for configuring how widgets in a GUI interact. For example, the value in one widget can control the visibility of another. Examples of the GUI support for Qt plugins are shown in Figure 1.

QCA BASED APPLICATIONS

The QCa framework currently includes a couple of applications. The main application is AsGui.

AsGui is a graphical control system user interface. It displays EPICS aware GUIs based on user interface files created using 'Designer' as shown in Figure 1.

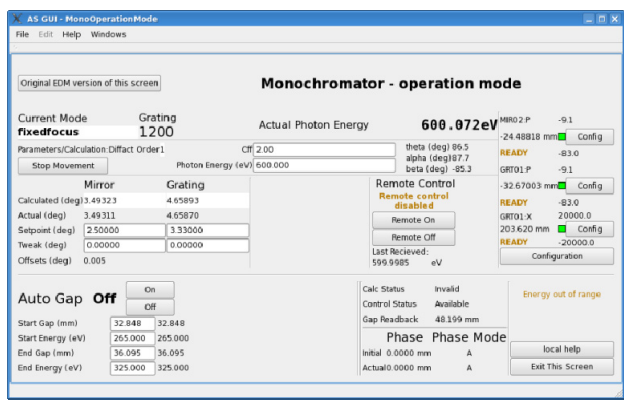

Figure 1: A sample GUI created in designer using EPICS aware plugins and GUI support plugins

QCaMonitor is a console application that takes a list of EPICS process variable names as an argument and monitors changes to the data specified by the names. It will perform the same task as the standard EPICS application caget. It is an example of using QCaString objects to generate a stream of textual based updates.

CLASS USAGE

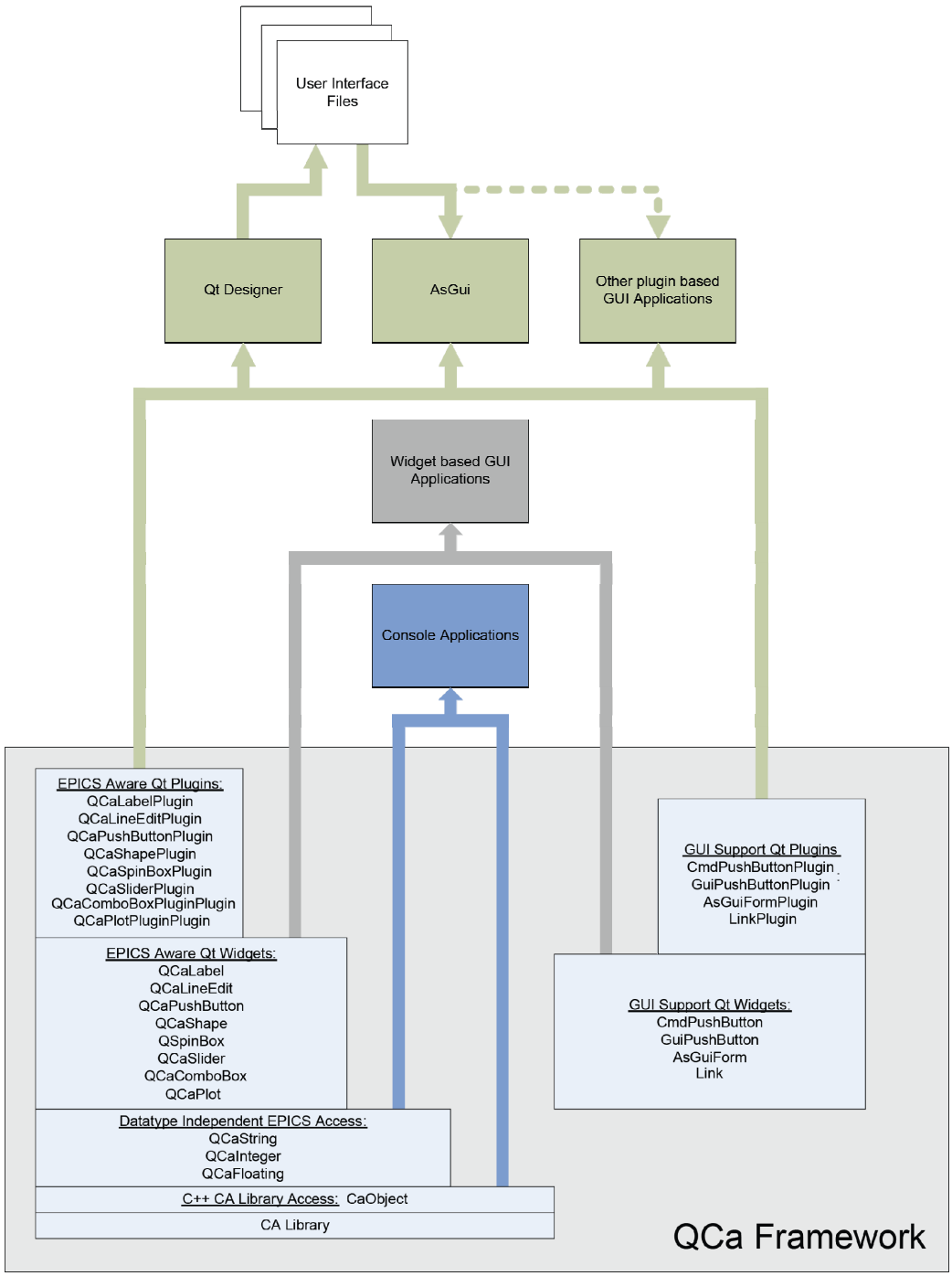

Figure 2: Typical QCa class usage

REFERENCES

[1] Philip Stanley Channel Access Client Library Tutorial. Los Alamos National Laboratory. http://lansce.lanl.gov/EPICSdata/ca/client/caX5Ftutor-1.html