**Archiver Appliance Support for EPICS Qt**

**Introduction**

[EPICS Archiver Appliance](https://slacmshankar.github.io/epicsarchiver_docs/) (AA) has been around for a while now and it seems to be becoming a successor of EPICS Channel Access Archive. As the Australian Synchrotron is also on the path of switching to AA, a decision to add support for AA in QE Framework has been made.

In this page you will find some information about some design decisions, how to prepare build environment and how to build QE Framework with AA.

**Design Decisions**

### AA Version

During the development of AA support for QE Framework a requirements for some extra AA features have arisen. Developers of AA have been kind enough to add those features to the project; this however means that we have introduced a dependency on a specific version of AA.

AA support for QE Framework has been developed with **Fall 2017 release (tag** **v0.0.1\_SNAPSHOT\_27-November-2017)** of AA. Most probably it will work with newer versions; it definitely won’t work with older ones.

### Google Protocol Buffers

Archived data can be retrieved from AA in many different formats, however Google Protocol Buffers (PB) are AA’s native serializing tool and provide fast and efficient serialization. Speed was the main reason to go with PB, consequently this means that PB libraries, headers and executables need to be present on the machine at build time and run time. On most Linux distributions these are readily available to be installed using distribution’s package manager.

### archapplDataSup

All PB related code has been separated into a new application, namely archapplDataSup (located in the QE framework top directory). Building of this application is dependent on how environment variables are set up. Built library has only one entry point and one data structure. The entry point is a function that takes raw PB input received from AA and saves parsed data into a vector whose reference has been passed to the function.

This means that there are no dependencies on PB in the code of QE Framework Qt project which allows the project to be built in exactly the same way as before, if no additional environment variables are defined.

This also means that archapplDataSup can be used as a standalone library for any other cases were parsing of AA’s PB data is needed.

### Qt 5.0 +

In Qt 5.0 support for JSON has been introduced as well as support for easy URL query composition. Since Qt 5.0 has been around for a while, those libraries have been used. This means that in order to build QE Framework with AA support, Qt 5.0 or higher must be used.

### Obsolete functions

Some of the functions have become obsolete and are only there so that any of references to them from external applications don’t become broken. These functions are:

* QEArchiveAccess::initialize() - Not needed anymore with new modifications.
* QEArchiveAccess::initialize(const QString& archives, const QString& pattern) – This functionality has now been removed. Archives and pattern can now only be defined using QE\_ARCHIVE\_LIST and QE\_ARCHIVE\_PATTERN environment variables respectively.

**Getting Ready**

### Have AA Deployed

There’s an extensive guide on how to do that on [AA’s website](https://slacmshankar.github.io/epicsarchiver_docs/). As mentioned before, AA support for QE Framework requires **Fall 2017 release (tag** **v0.0.1\_SNAPSHOT\_27-November-2017)** or newer.

### Install Google Protocol Buffers

AA support for QE Framework has been tested with versions 2.5.0 and 3.5.0 and 3.5.1

**On Linux**

This should be fairly painless. All major distributions should have PB software available via their package managers. So in case of Fedora and other RedHat flavours you’d run:

yum install –y protobuf

At least in the case of CentOS 7, this will install a fairly old version of PB (2.5.0). For the purpose of AA support in QE Framework, this is good enough, however you can get the latest version of PB from their [github releases website](https://github.com/google/protobuf/releases/latest).

**On Windows**

As expected, here is where it gets a bit trickier. So far the building of PB has only been tested with MinGW on Windows 7 and Windows 10, both x64.

* CMake needs to be installed. It can be found [here](https://cmake.org/download/)
  + Make sure that the directory containing cmake.exe is on the PATH
* MinGW needs to be installed, I simply used the one that comes with Qt installation
  + Make sure you add <MinGW\_DIR>\bin and <MinGW\_DIR>\lib to the PATH
  + If you haven’t already, rename <MinGW\_DIR>\bin\mingw32-make to make for convenience
* Download the source for C++ code from [github releases website](https://github.com/google/protobuf/releases/latest). The package name should be in the format of “protobuf-cpp-x.x.x.zip”. Extract it.
* Open command prompt
* Navigate to the directory where you have previously extracted PB source code to (PB\_DIR).
* Navigate to <PB\_DIR>/cmake, create directories <PB\_DIR>/cmake/build/release and move into newly created directory.
* Run the following:

cmake -G "MinGW Makefiles" ^

-DCMAKE\_BUILD\_TYPE=Release ^

-DCMAKE\_CXX\_STANDARD=11 ^

-Dprotobuf\_BUILD\_SHARED\_LIBS=ON ^

-Dprotobuf\_BUILD\_TESTS=OFF ^

-DCMAKE\_INSTALL\_PREFIX=< MinGW\_DIR> ../..

This will generate Makefiles for MinGW, which will be used to build Release build, have default C++ version set to C++11, build dynamic libraries, avoid building tests and install generated files in <MinGW\_DIR>/lib, <MinGW\_DIR>/bin and <MinGW\_DIR>/include

* Run:

make –j3 # j3 is optional but it makes building a lot faster

make install

* Check that everything has been installed successfully by running:

protoc

This should print the help for this command.

### Set Environment Variables

Three new optional environment variables have been introduced with AA support for QE Framework:

* QE\_ARCHAPPL\_SUPPORT – This is a build time variable. If set to YES, QE Framework will be built with support for AA. If not defined, support for AA will not be built.
* QE\_ARCHIVE\_TYPE – This is a run time variable. It can take of two values (CA, ARCHAPPL) where CA defines that we’re connecting to EPICS Channel Access Archive and ARCHAPPL defines we’re connecting to AA. If not defined, QT Framework will try to connect to EPICS Channel Access Archive.
* PROTOBUF\_LIBS\_DIR – This is a build time variable. In case PB libraries are not on your PATH (Windows) or LD\_LIBRARY\_PATH (Linux), this variable is used while building archapplDataApp to point to PB libraries directory.

**Building**

Once PB is installed and variables set, QE Framework can be normally built in the same way as it has been built before, by executing “make” in the QE Framework’s TOP directory.

The build with AA support enabled will result in two new libraries archapplData.\* in QE Framework’s libraries directory and one new header archapplData.h in framework’s include directory.

**Running**

The only thing that’s left to do now is to define the URL in QE\_ARCHIVE\_LIST to the AA. The URL should point to AA’s management BPL so something like <http://archiver01/mgmt/bpl> (protocol and port are not required; default is http and port 80).