Qt® Now Available Under LGPL

Implications for Commercial and Government Users

By Mark Hatch, COO of Integrated Computer Solutions, Inc.
January 14, 2009 (revised 1/20/09)

Integrated Computer Solutions Incorporated

54 B Middlesex Turnpike • Bedford, MA • 01730 • www.ics.com • 617.621.0060

Qt® Now Available Under LGPL
Implications for Commercial and Government Users

Table of Contents

Qt Now Available Under LGPL
Implications for Commercial and Government Users

Introduction

On the 14th of January, 2009, Qt Software announced that they added the Lesser GNU Public License v2.1 (LGPL) as an additional licensing option for users of its Qt® cross-platform development framework for C++. This means that you can now develop Qt applications under any of three licenses: Qt Commercial, GNU Public License (GPL), and LGPL. The addition of this license option has exciting implications for commercial and government users of Qt.

For all users, the availability of the LGPL version of Qt will prove to be a significant benefit resulting in more Qt extensions/add-ons, additional platform ports, and increased numbers of skilled Qt developers. For many, the LGPL version also provides a significant financial advantage by eliminating the developer and deployment license fees associated with the commercial version. However, for some users, the commercial version will remain the correct decision. This whitepaper analyzes the implications of the availability of the LGPL for Qt and attempts to provide some practical observations that commercial and government users can utilize to select the best license for their needs.

There are two disclaimers that need to be made up front:

- **I am not a lawyer, and I don't pretend to be one.**
  If you need help writing Qt software, I can help. If you want legal advice, ask a lawyer.
- **I work for ICS.**
  This whitepaper is based upon the information that I have seen, but I do not work for Nokia, or represent Nokia, and I do not pretend to be making any promises for Nokia.

I welcome comments of all sorts and will incorporate them in future revisions of this whitepaper. Please email them to me at hatchm@ics.com and check back from time to time at http://www.ics.com/ to get the latest update of this document.

What Is Qt?

The development of applications that run on multiple operating systems (e.g., Windows, Linux, MacOS, etc.) can be expensive. The Qt framework was developed by Trolltech to simplify the development of cross-platform applications by providing a single API that is consistent and highly functional with native look-and-feel and native performance on all platforms. By using Qt as the underlying foundation, a developer can write his application once and re-compile it to run on any major operating system. In addition, an application that was originally developed as a desktop device could be delivered on an embedded device with only minor changes. Qt completely revolutionized the way cross platform applications are developed and is the most economical solution for many

companies. This has led to Qt being adopted as a standard by some of the largest companies in the world. Historically, Trolltech has been a strong supporter of the open source community. The leading desktop for Linux, KDE, was written using Qt.

Trolltech was acquired by Nokia in 2008. The business unit is now known as Qt Software.

What Was Announced…

Qt Software announced that starting with Qt version 4.5, scheduled to be released in March 2009, users can choose to use Qt under the LGPL V2.1 license[1]. Developers also have the option to continue licensing Qt under either the commercial Qt license or the GPL. The LGPL option covers not only all operating system platforms, but also covers both desktop and embedded environments. Qt Software will continue to offer support for commercial Qt licenses. As part of the announcement, they also stated that they will be offering support for users of the LGPL version too.

This option is not being made retroactive. Prior versions of Qt (i.e., Qt 4.4.x, 4.3.x, etc.) will only be available under the commercial Qt and GPL licenses.

The official press release is available at: http://www.qtsoftware.com/. There is more information there on the goals and objectives behind Nokia's decision. I would recommend you read and process this information yourself.

After the Nokia acquisition of Trolltech last year, many in the Qt community were concerned about the implications for developers that use Qt. I believe that this bold step by Nokia is going to accelerate the adoption of Qt for both open source as well as proprietary applications. In short, the best toolkit around just got even better.

LGPL vs. GPL?

Most commercial and government development organizations are already well acquainted with the cost of using code released under the GPL – although the software costs $0, you may be required to share your source code with the world, and more importantly, your competitors. The potential loss of the intellectual property to competitors often dwarfs the cost of a commercial license. This is the reason why many companies have purchased commercial Qt licenses even with the prior availability of the GPL version.

Although derived from common principles and sharing much of the license boilerplate, the goals of the LGPL are significantly different from that of the GPL. The LGPL preamble does an adequate job of explaining the different goals:

[1] A 3.0 version of LGPL is also available. However, at the time that this whitepaper was written, Qt is only being made available under the 2.1 version.

> "…When a program is linked with a library, whether statically or using a shared library, the combination of the two is legally speaking a combined work, a derivative of the original library. The ordinary General Public License therefore permits such linking only if the entire combination fits its criteria of freedom. The Lesser General Public License permits more lax criteria for linking other code with the library…."

And then continues…

> "…For example, on rare occasions, there may be a special need to encourage the widest possible use of a certain library, so that it becomes a de-facto standard. To achieve this, non-free programs must be allowed to use the library…"

So the explicit goal of the LGPL is to allow "non-free" (e.g., proprietary) applications to use a piece of software without being forced to release the source code of the proprietary portions of the application to others. This is the fundamental difference between the two open source licenses.

The LGPL and Your Source Code

Although this whitepaper is not intended to be a detailed legal analysis of the LGPL, it is worth understanding that the LGPL carefully delineates between two concepts:

- Work *based* on the Library[2]
- Work that *uses* the Library

If you were to modify one of the Qt libraries (or for that matter Qt Designer), you would be creating a Work that was *based* on the Library. Conversely, if you were writing an application that used Qt for its intended purpose of simplifying the porting of applications, you are creating an application that *uses* the Library.

It is important to determine whether you are creating a work *based* on or a work that *uses* the Library. If you create a work *based* on the Library, you must release your source code and make it possible for others to rebuild your modified Library. However, if you are developing a work that *uses* the Library, then you need to make it possible for someone to replace the Library with a newer version or a modified version of the Library.

So what about using C++ subclassing and templates to extend or tailor the facilities of Qt? This is one area where the LGPL V2.1 shows its age – it was initially released in 1999. These issues were directly addressed in LGPL V3.0 while V2.1 rather obliquely addresses it in section 5 where it says:

[2] Note: Historically the LGPL was originally named the Library GNU Public License as it was designed for use by programming libraries like the GNU C library. However, it was extended and repurposed in version 2.1 to be the "Lesser GNU Public License" because it was being used for software other than programming libraries. For example, OpenOffice is released under the LGPL. So when the license talks about "Library," do the mental translation and understand that in this context Library means the software that is licensed under the LGPL.

> "When a "work that uses the Library" uses material from a header file that is part of the Library, the object code for the work may be a derivative work of the Library even though the source code is not."

Since C++ subclassing and the expansion of templates are accomplished through the use of a Library's header files, the LGPL would seem to suggest that the typical use of those facilities do not make your source code a "work *based* on the Library." Public discussions on this topic on the Internet generally agree that subclassing is safe. However, there has been some spirited debate on whether the use of C++ template facilities (which Qt does provide for its container objects) pollutes the application source code and so makes it susceptible to the LGPL license. Some suggest that to be safe, a copyright owner needs to add a special exception to their LGPL license. Others assert that there is no need to be concerned. This could be one area where you might want to review the final LGPL license used for Qt – especially the "special exceptions area" – and check with your legal counsel.

Living in the LGPL World

While it looks like a long list, the LGPL is a very easy license with which to comply. You are probably already using libraries that are licensed under the LGPL (e.g. the GNU C Library, glibc, is licensed under the LGPL). If you are deploying applications on Linux, you are almost certainly using libraries licensed under the LGPL. But you should be aware that the privilege of using LGPL software comes with several basic obligations:

1. You must tell people of their rights. Specifically, Section 6 of the LGPL license says:

   "You must give prominent notice with each copy of the work that the Library is used in it and that the Library and its use are covered by this License. You must supply a copy of this License. If the work during execution displays copyright notices, you must include the copyright notice for the Library among them, as well as a reference directing the user to the copy of this License."

2. The modified work (i.e., what you make the Library into) must in itself be a Library. That's an easy one to comply with too.

3. In any files that you have changed, you must include prominent notices identifying yourself and documenting the date of change. This is just good programming practice, and so this should be a pretty easy clause to satisfy.

4. If you add a reference to a function that is external to the Library, you must make a good faith effort to ensure the Library functions correctly if this function is not available. For example, suppose you needed to add a high accuracy square root calculation to Qt. Then if that function was not available (suppose it was a proprietary math function), you have an obligation to ensure that the modified Qt library you provided would continue to function (perhaps using a slower, less accurate generic

function). If you want to modify Qt to use a special proprietary communication library, you may want to re-think your architecture. This requirement is easy to satisfy with advanced planning.

5. You cannot change the license (although you could convert your modified library to a GPL version – see section 3 of the LGPL) and you cannot charge for the source code to the Library.

6. If the Library is not normally distributed with the operating system, or you have made changes to the Library, or if you bundle the Library into your software distribution, you must make the source code to the Library available – either by including it with your software or by offering access to the source code from a designated place (e.g., download, toll-free line to order a CD, etc.)

7. Your application must be "re-linkable" against a modified version of the Library. In practical terms this means the application must dynamically link with the Library. If you statically link you are required to provide the object modules to your application and any re-linking tools needed to enable substitution of a modified Library[3]. Since most applications use dynamic linking, your application is probably already compliant with this requirement.

These basic obligations are easily fulfilled and not too onerous. However, it is very easy to forget to include a copy of the LGPL license, or not display it with other copyrights at runtime. But probably the most common failure for commercial firms is forgetting to make the library source code available when a product is shipped externally. Since this is an unusual part of a product's release cycle, it is often overlooked in getting a new product to market.

So Can I Use the LGPL Version of Qt?

For most companies, the answer is YES. Remember, the LGPL licensed version of Qt is the same software as the commercial version, you can purchase support for it and you probably already use other software licensed under the LGPL without any problems.

When Does the LGPL Version of Qt Not Fit?

The LGPL might not be a good fit for either business or engineering reasons. If you have a business issue using the LGPL, I suspect you know it. You have probably already had long and frustrating discussions with your business/legal folks on topics like "warranties," "infringements" and the general use of any open software.

[3] In some of Qt Software's briefing materials that I have seen prior to the announcement, they directly say that to use the LGPL library, you must dynamically link it. In my research on this issue, a common interpretation of the LGPL is that you can statically link your application with an LGPL library as long as you make it possible for the user to re-link with a different (hopefully improved) version of the Library. In the simplest case this is making the ".o" files of your application available along with any tools needed to re-create the executable. However, remember the "I am not a lawyer" disclaimer? This is an area where you might want to do further research or ask Nokia for clarification. While you are looking at this section of the LGPL, there is also a "reverse engineering" clause that is worth thinking about.

There are three engineering reasons for staying with the commercial license:

1) Your value is in modifications to Qt. This includes the case where you have extended Qt to use a proprietary library. If you use the LGPL version, you will have to share your intellectual property with others.

2) You are delivering a statically linked application, cannot provide a re-link tool, and cannot move to dynamic linking. One can imagine this situation in certain embedded or real time applications.

3) Your application uses an older version of Qt and it is difficult to migrate the software to Qt 4.5.

Predictions and Last Words

Based on my business experience with Qt, I have several predictions that I would like to share:

1. **Bad News for Competing Toolkits**
   Technically, Qt is the best toolkit around. However, it has also been expensive. Toolkits like GTK+ and wxWidgets have been competitive because they are free (in terms of price). Native API's are low cost, but single platform. Qt's ease of use, and cross-platform support combined with the new license will make Qt the winner in this competition.

2. **Qt – the King of Embedded GUIs**
   Qt is already successful in embedded systems. However, the runtime royalties of Qt have slowed its adoption. With the LGPL version, embedded systems now have a quality runtime royalty free solution.

3. **Existing Qt Applications Will Migrate Quickly to Qt 4.5**
   Anybody still using Qt 3, is already paying a support premium for the continued use of a "legacy" version. Although it previously might have been a bargain to stay on stable and familiar Qt 3, the effective elimination of developer fees is going to tip the balance.

4. **Qt Experience Will Remain in Short Supply**
   Items 1-3 above translate directly into increase demand for Qt developers. As the leading Qt Consulting firm in North America, we already know the difficulty in finding Qt experts. By the fall 2009, the lack of experienced Qt developers will become the inhibitor to the start of new Qt based projects. If you are an experienced Qt engineer, this is good news in today's economy. For the rest of you, start reading those Qt books and watch some ICSNetwork webcasts at http://www.ics.com/icsnetwork/

5. **Keep an Eye on WebKit**
   There is more here than has been effectively communicated to the community. Given Nokia's focus in the mobile market, I personally wonder when, not if, Qt will really bridge the gap between desktop, embedded, mobile and the web. To learn more, check out our Qt WebKit webcast at http://www.ics.com/icsnetwork/.

6. **Independent Support Options Will Emerge**
   As the number of Qt developers doubles or triples over the next two years, Qt Software will be challenged to maintain its high quality of support. Under the LGPL, updates/new releases are effectively free – you give it to one, then everybody can request it. This provides the opportunity for independent organizations to offer premium services that blend support with consulting. ICS has already sold several premium support contracts that provide a higher service level and address the question: "How do I do this the right way?"

Why This Announcement is Important

The move from the GPL to the LGPL is a gift from Nokia to the open source community, and more generally to the software development community. They may indeed reach their goal of creating a popular standard way of creating applications, and that may lead to more mobile applications, and a better mobile experience for their users. But regardless of their future success, they have given a valuable gift. By making Qt effectively free, they are making software development better and cleaner. Qt makes single platform applications a thing of the past, and I believe that Qt is now destined to become a standard. Nokia is forgoing license fees which were 10s of millions of dollars a year, and I for one would like to say "thank you".

About the Author

Mark Hatch is the Chief Operating Officer of Integrated Computer Solutions, Inc. (ICS), the largest supplier of Qt consulting and training services in North America. A 30+ year industry veteran, his career has spanned from the very early days of UNIX at the University of California, Berkeley in the 1970s to the exciting open source revolution of the present. His use of Linux dates back to 1993 where he had to insert over 20 floppies to install Linux Release 0.99 p13.

Mark joined ICS in 1995 as VP of Marketing and today manages the Qt business, both products and consulting services, at ICS. Previously, Mark was VP of Marketing for a networking middleware company as well as Manager of Software Marketing for Apollo Computers. In this latter position, he played a key role in the formation of a number of industry groups including the X Consortium and the Open Software Foundation. The X Window System and the Distributed Computing Environment (DCE) achieved their current position of industry standards in part because of the efforts of Mark and his marketing team at Apollo. Mark has a Masters in Electrical Engineering and Computer Science from the University of California, Berkeley. He also has a Masters of Business Administration from Boston University.

About ICS

Integrated Computer Solutions, Inc. (ICS), of Bedford, MA, is the largest independent supplier of add-on products, training, and professional services for the Qt® cross-platform framework. ICS is a Qt Software preferred Qt training and consulting partner in North America. ICS has been providing professional services to its customers since its inception in 1988 and provides training services to individuals, small groups, and over half of the Fortune 500, including many whose technology defines their core business. In addition, ICS sponsors the ICSNetwork, an online training center where users can learn advanced techniques for developing with Qt. More information can be found at www.ics.com.