Release notes

What's new in GCF 3 ?

GCF 3 is the successor of GCF 2.x series. It is designed to address new problems and those that were already being addressed by GCF 2.x in a reliable way. This was achieved by a complete re-architecturing of the entire library, without compromising on its identity.

This document captures what is new in GCF 3, and is divided into different sections to portray the differences in the relevant aspects of a GCF application.

1. Application

- GCF 2.x mandated that all the GCF applications have to be GUI applications. With GCF 3 however, that is not the case.
- GCF 3 has three different application configuration options and you could build your application based on any one of these. The available configurations are
    - **GCF::Application** - to create non-gui applications
    - **GCF::GuiApplication** - to create Qt widgets based applications
    - **GCF::QmlApplication** - to create Qt quick based applications
- GCF 3 doesn't impose any particular method to load a list of components, as opposed to the **Application XML** mandated in GCF 2.x. Now it is up to the developer to create the prefered mechanism to identify the components to be loaded, and request loading of the same.
- All GCF 3 applications are capable of parsing command line arguments in **key:value** format, which can then be accessed using the **key** specified from the global GCF application services object.

2. Components

- Components in GCF 3 are not singleton. You can create as many instances of a component as you want.
- In GCF 3 there are three different component configurations available. Depending on the domain of problems they address, your components could choose to inherit from any one of the available configurations.
- The available configurations are
    - **GCF::Component** - can be loaded by any GCF application
    - **GCF::GuiComponent** - can be loaded by only GCF::GuiApplication
    - **GCF::QmlComponent** - can be loaded by only GCF::QmlApplication
- GCF 3 uses **QSettings** to save / load the configuration options of components. Components can configure the file to be used by QSettings.
- GCF 3 uses custom **events** to notify each component about different stages of their loading. Developers can install event filters to hook into the loading process of a component and track it.

- GCF 3 uses a much more simpler **content file** system to configure the exposed objects of a component.

4. IPC

- GCF 3 comes with a much improved interprocess communication mechanism.
- With GCF 3, developers can create several instances if GCF::IpcServer (as opposed to a single instance of GCF::AppAccessPoint in GCF 2). This makes it possible for components to define their own access points.
- Marshalling and unmarshalling of data values make use of QVariant's streaming capabilities - instead of mandating a custom mechanism. This results in simpler and more robust code.
- GCF 3 offers a new class for making asynchronous IPC calls without having to create a GCF::IpcRemoteObject.

5. Tools

- GCF 3 comes with three new powerful tools.
- The new tools in GCF 3 are
    - **GCF Investigator** - A simple to use GUI testing suite for GCF applications.
    - **GCF GDriveLite component** - A collection of mechanisms that makes integration of a GDrive file system to your GCF application, extremely simple.
    - **GCF Fiber** - A simple mechanism to host the functionalities in your components like a web service, which can then be accessed from clients written in any technology.

6. Other utilities

- **ObjectMap** and **MapToObject** classes to ensure that when you need to have QObjects as the key or value in a map, only valid entries remain in the map.
- A comprehensive logging mechanism
- A context aware error dump mechanism.
- **SignalSpy** mechanism that doesn't require your application to link against testlib from Qt.

Known issues

The following sections capture the known issues at the time of release of GCF 3. We are already working on the reported issues, but, unfortunately couldn't bundle the fixes within the release of the software. We would be releasing fixes for these issues via a minor version release, pretty soon.

1.Build system

- Right now we don't have a mechanism in GCF that enables building the library in both debug and release modes against several Qt versions in one shot
- We have not tested static building with GCF

- Loading a GUI component from Core app crashes the application. (Same is the case with other combinations as well)

2.Core

- Object merging doesn't work with addContentObject() method

3.IPC

- Invoking methods across 32-bit and 64-bit machines doesn't work.

4.GDrive

- Refresh button in the QML example doesn't work

5.Rover App

- Spiral view is not polished and complete