Katana 2.5v1: API Changes and Compatibility

This document gives an overview of the changes in Katana 2.5v1 that may be of interest to developers.

C++ Plug-in Apis: CMake Build System

We now provide explicit support for building C++ plug-ins using the CMake build system in the form of a CMake package file, located in `plugin_apis/cmake`. Importing this package file in your `CMakeLists.txt` will define logical CMake targets corresponding to the various Katana APIs, and these define the required dependency information to plug-in targets that consume them, including source files, include directories and preprocessor defines.

Use of the CMake package file removes the need to manually specify an ad-hoc set of Katana plug-in source files when compiling your plug-ins. For example, an Op's `CMakeLists.txt` might look like this:

```
cmake_minimum_required(VERSION 3.2)
find_package(Katana PATHS "${KATANA_HOME}/plugin_apis/cmake" REQUIRED)
add_library(MyOp MODULE src/op.cpp)
target_link_libraries(MyOp Katana::FnGeolibOpPlugin)
```

Katana's own example plug-ins (in `plugins/Src`) have been updated to use this new build system, and their `CMakeLists.txt` files serve as a useful reference for build your own Ops and other plug-ins.

Symbol Visibility

Declarations in Katana's plug-in API headers have been tagged with symbol visibility annotations. For example, functions and classes in the FnAttribute module are tagged with the `FNATTRIBUTE_API` preprocessor define, and those in the FnGeolib module are tagged with `FNGEOLIB_API`.

Linux customers will not see any change from this. At this time, Windows users will need to compile their plug-in sources with additional `FN<MODULE_NAME>_STATIC` preprocessor defines (e.g. `FNATTRIBUTE_STATIC`) for every Katana module they use, as plug-in headers currently assume that the symbols are being imported from a .dll. You'll likely encounter compiler or linker errors if this is not the case and these defines are missing. We plan to address this in a future release (TP 226219 - Simplify Katana plug-in visibility macros).

If building with CMake and the shipped CMake package file (see above), the required preprocessor defines are added automatically.

Plug-in APIs

FnAsset

The following additional Asset API functions have been exposed in `DefaultAssetPlugin` (C++) and `Asset` (OpScript):

- checkPermission()
- resolveAllAssets()
- resolveAssetVersion()
- getAssetDisplayName()
- getAssetVersions()
- getAssetFields()
- buildAssetId()
- getAssetIdForScope()
- getAssetAttributes()

Plug-ins using `DefaultAssetPlugin` will need to be recompiled.

FnRender, FnDisplayDriver

The implementation of the rendering APIs -- FnRender and FnDisplayDriver -- have been streamlined, with the details of the communication between the plug-in and the host application moved into Katana-internal libraries. FnDisplayDriver no longer depends on thirdparty Boost or ZMQ libraries.

As part of this effort, some internal headers have been deprecated on Linux, and others are no longer shipped. However, we expect that the revised headers will source-compatible with existing render plugins.

The changes include:

- FnRender/SocketConnection.h has been deprecated without replacement on Linux, and is not available on Windows.
- The FnRender routines responsible for parsing and serialising command-line arguments have been deprecated.
- The following internal headers from FnDisplayDriver have been removed: ContextSingleton.h, IsAliveMessage.h, MessageTypes.h, ZmqHelpers.h

FnRenderOutputUtils

The internal bootstrapGEOLIB function has been removed.

FnGeolib

Platform-specific headers in the util/ folder have been deprecated without replacement on Linux, and are not available on Windows. The headers are: Mutex.h, Platform.h, PowerNap.h, Semaphore.h, ThreadException.h,Timer.h.

FnPluginManager

- findPlugins() no longer takes cacheFilePath param

FnScenegraphIterator

- Suite version number bumped.
- Added: getOpTreeDescriptionFromFile()
- Added: getIteratorFromOpTreeDescription()

FnViewerModifier

In order to support batching of draw calls, the Viewer Modifier Plug-in API has been updated to version 2.0.

Katana no longer automatically hides looked-through custom Viewer Modifiers:

- The ViewerModifierInput class now provides an isLookedThrough() method that can be used to alter the GL representation of a location if it is currently being looked through. This method takes an optional boolean argument, includeChildren (defaulting to false), that allows descendants of looked-through locations (such as light filters) to do the same if they so wish.
- In order to replicate the behaviour of earlier versions of Katana, custom Viewer Modifier Plug-ins for locations of type 'camera' and 'light' should, in their draw function, call isLookedThrough() and return without drawing if the return value is true.

Compatibility

A VMP compiled with the old v1.0 API **can** be loaded in a version of Katana which supports the new v2.0 API. These v1.0 plug-ins won't make use of the draw-call batching callbacks feature.

Loading Order

- Plug-ins compiled with a later API version have a higher priority, regardless of the order in which the plug-ins are loaded/registered. A v2.0 plug-in will always take precedence over a v1.0 plug-in that is competing for the same location type.
- The default Viewer Modifiers shipped with Katana 2.2 and higher all make use of the new v2.0 plug-in API. In order to overload a shipped Viewer Modifier Plug-in (for example, the Viewer Modifier for "light" location types), your custom VMPs will need to be recompiled against the new v2.0 API.
- Viewer Modifier Plug-ins are loaded in **alphabetical order**, and the **last** plug-in loaded will override any previously loaded plug-ins of the same API version, for a given location type. This means that, in cases where multiple VMPs of the same API version are competing for the same location type, the VMP with a registered name that is **last** alphabetically will win. (This is unchanged from previous Katana versions.)
- The name of a VMP can currently be used to overload another VMP, but a plug-in compiled against older API version can't overload a VMP compiled against a newer API.

Notes for VMP creators

- In order for VMPs to work with older versions of Katana, VMPs must use API v1.0.
- In order to support batching of drawing calls, a VMP must use API v2.0. This makes the VMP incompatible with earlier versions of Katana.
- VMP creators may wish to provide two versions of their VMPs: one compiled against API v1.0, and one against v2.0. The appropriate path in **KATANA_RESOURCES** could then be manipulated so that earlier versions of Katana pointed to the directory containing the appropriate plug-in.

Pystring

- New constants: pystring::os::sep, pystring::os::pathsep that evaluate the appropriate directory separator ('/' or '\') or search path separator (':' or ';') for the target platform.
- Extended expandvars() to support different sources of environment variables.

Third-party Software Versions

Library Name	Version in Katana 2.1v2	Version in Katana 2.5v1
OpenImageIO	0.1	1.5.20
OpenColorIO	1.0.0	1.0.9
OpenEXR	2.0.1	2.2.0
Boost	1.46.0	1.55.0
yaml-cpp	0.3.0	(removed)
SIP	4.15.5	4.16
GLEW	1.5.8	1.13.0*
Freetype	2.4.4	2.5.0.1
Libpng	1.2.25	1.4.8
numpy	1.5.1	(removed†)

* Viewer modifier plug-ins no longer need to call `glewInit()`

† Present in Linux version, but scheduled for removal in Katana 2.6.