Release Notes for Katana 1.4v1

Release Date

29 April 2013

System Requirements

- Katana 1.4v1 is tested and qualified on Linux 64-bit CentOS/RHEL 5.4
- A graphics card which supports OpenGL shader model 4.0
- A supported renderer (see below)

Supported Renderers

Katana 1.4v1 supports PRMan 17.0 and Arnold 4.0.11.0. The supplied renderer plug-ins are compiled and tested against these versions. Minor version increments of PRMan and Arnold may work, as long as they are API compliant with the supported versions.

To use a version of PRMan or Arnold other than those listed above, you may need to recompile the renderer plug-in.

To expose new features and portions, you may need to modify the renderer plug-in.

Using a version of PRMan or Arnold other than those listed above may produce unexpected behavior. Please note that we can only guarantee to respond to Katana bug reports when they are reproducible with the supplied versions of the renderer plug-in.

Important Changes for 1.4v1

- Support for PRMan 16.4 has been deprecated, and Katana is no longer shipped with a PRMan 16 renderer plug-in.
- The Katana 1.4v1 release addresses inconsistencies in the internal use of the Asset API, and adds asset import and export support for new use cases. In addition, the supplied PyMockAsset example plug-in reflects these updates and illustrates newly supported behavior. See New Features for 1.4v1 for more on the updates to PyMockAsset.

Nodes That Support Asset Management Import

The following nodes —and use cases— support asset management:

- Alembic_In
- ScenegraphXML_In
- AttributeFile_In
- LookFileAssign
- LookFileGlobalsAssign
- LookFileMaterialIn
- LiveGroup
- ImageRead
- Importomatic
- PrmanGlobalSettings
- PrmanObjectSettings
  Assets imported through the **ribInclude** parameter are supported.
  See Known Issues & Workarounds for more on this.
- ArnoldGlobalSettings
  Assets imported through the **assInclude** parameter are supported.
- PrimitiveCreate
  For the following asset types:
  - rib archive
  - coordinate system sphere
  - coordinate system plane
- Material
  Shaders can be accessed using the Asset API.

NOTE: Accessing materials through the Asset API now uses the **getRelatedAssetId()** function of the Asset plug-in to check for a related asset-managed Args file. If one is not found, it looks up the Args file in the usual fashion, relative on the **.so** file location.

- RenderProceduralArgs

NOTE: Accessing a render procedural through the Asset API now uses the **getRelatedAssetId()** function of the Asset plug-in to check for a related asset-managed Args file. If not, it will look for an Args file with the same name as the **.so** file, in the same directory. For example, an **.so** file under **/tmp/proc.so** expects an Args file at **/tmp/proc.so.args**.

Asset Publishing Behavior

Under the new asset publishing behavior, the Asset ID returned from **createAssetAndPath()** is used to create the fields passed to **postCreateAsset()**. The result from **postCreateAsset()** is used from that point onward (such as in the **File** > **Open Recent** menu or in any references to that Asset ID in the current scene):

```
assetFields1 = assetPlugin.getAssetFields(assetId, True)
id1 = assetPlugin.createAssetAndPath(..., assetFields1, ...)
[Write Katana scene file, etc.]
assetFields2 = assetPlugin.getAssetFields(id1, True)
id2 = assetPlugin.postCreateAsset(..., assetFields2, ...)
```

Import and Export Through the Katana UI

There are a number of UI locations where files are imported and exported interactively, typically as menu options. When exporting through the UI from any of the nodes listed below, the asset publishing procedure is as described in Asset Publishing Behavior.

The following use cases support asset management:

- Katana Scene Files

  Choose **File** > **Open**, **File** > **Save**, **File** > **Save As**, and **File** > **Export Selection** to export Katana Scene Files.

  Choose **File** > **Import** to import Katana Scene Files.
- Live Groups

  Choose **File** > **Import** to import a Live Group.
- Macros

  Choose a node's **Wrench** menu > **Save As Macro...** to export a macro.
- Gaffer

  Choose right-click > **Add** > **Import Rig...** in a Gaffer node's **Parameters** tab to import a Gaffer rig.

  Select a rig in a Gaffer node's **Parameters** tab, and choos right-click > **Export Rig...** to export a Gaffer rig.
- FCurves

  Choose right-click > **Export FCurve** on a suitable node parameter in the **Parameters** tab to export FCurves.

  Choose right-click > **Import FCurve** on a suitable node parameter in the **Parameters** tab to import FCurves.
- Catalog Images

  Choose **File** > **Import Image / Sequence** in the **Catalog** tab to import Catalog images.

  Choose **File** > **Export Catalog** > in the **Catalog** tab to export Catalog images.

- **LookFileManager Settings**

  Choose **Look FIles** > **Cog menu** > **Import / Export** > **Import Manager Settings...** in a LookFileManager node's **Parameters** tab, to import LookFileManager settings.

  Choose **Look FIles** > **Cog menu** > **Import / Export** > **Export Manager Settings...** in a LookFileManager node's **Parameters** tab, to export LookFileManager settings.
- **Scene Graph Bookmarks**

  Choose **Bookmark menu** > **Import Bookmarks...** in the Scene Graph to import Scene Graph Bookmarks.

  Choose **Bookmark menu** > **Export Bookmarks...** in the Scene Graph to export Scene Graph Bookmarks.

Nodes That Support Asset-Managed Export

The following nodes and use cases support asset management:

- Render
- ImageWrite

When performing a Hot Render from a Render node, or an ImageWrite node, the pre- and post-publishing steps are not performed. To manually perform those steps, go to the **Parameters** tab of a Render or ImageWrite node and choose **Action** > **Pre-Render Publish Asset** and **Action** > **Post-Render Publish Asset**.

- LookFileBake
- LookFileMultiBake

Asset Types, Contexts and Relation

The following asset types are available to the **AssetAPI** module:

- kAssetTypeKatanaScene
- kAssetTypeMacro
- kAssetTypeLiveGroup
- kAssetTypeImage
- kAssetTypeLookFile
- kAssetTypeLookFileMgrSettings
- kAssetTypeAlembic
- kAssetTypeScenegraphXml
- kAssetTypeCastingSheet
- kAssetTypeAttributeFile
- kAssetTypeFCurveFile

- kAssetTypeGafferRig
- kAssetTypeScenegraphBookmarks
- kAssetTypeShader

In addition, the following list of contexts is available inside the **AssetAPI** module, and passed as hints to the asset browser:

- kAssetContextKatanaScene
- kAssetContextMacro
- kAssetContextLiveGroup
- kAssetContextImage
- kAssetContextLookFile
- kAssetContextLookFileMgrSettings
- kAssetContextAlembic
- kAssetContextScenegraphXml
- kAssetContextCastingSheet
- kAssetContextAttributeFile
- kAssetContextFCurveFile
- kAssetContextGafferRig
- kAssetContextScenegraphBookmarks
- kAssetContextShader
- kAssetContextCatalog
- kAssetContextFarm

A constant to hold the relationship between assets has been added. This constant is used when the **getRelatedAssetId()** function is called:

- kAssetRelationArgsFile

Updated WriteToAsset Functions

The **WriteToAsset()** function on LookFileBake and LookFileMaterialsOut nodes now supports an optional **args** dictionary of strings which is passed to **createAssetAndPath()** as well as **postCreateAsset()**.

Previous method:

```
def WriteToAsset(self, frameTime, assetId, versionup=False,
                 publish=False, progressCallback=None)
```

New method:

```
def WriteToAsset(self, frameTime, assetId, args=None,
                 progressCallback=None)
```

NOTE: The previous way of calling this method using **publish** and **versionUp** arguments is no longer supported. If desired, the **publish** and **versionUp** flags can still be passed inside the **args** dictionary.

NOTE: The **comment** parameter on a LookFileBake node is no longer automatically passed to **createAssetAndPath()** via **args**.

New Features for 1.4v1

There are no new features in this release.

Feature Enhancements for 1.4v1

The PyMockAsset example plug-in has been extended to reflect the updates to the Asset API. Worth noting are the example implementation of **getRelatedAssetId()** (previously unimplemented), the use of the newly defined asset type constants in **createAssetAndPath()**, and improvements to the browser when selecting assets.

NOTE: The ${MOCK_DEBUG} environment variable can be set to print debugging information about calls to the Asset plug-in.

In addition, PyMockAssetWidgetDelegate shows how the asset browser can be disabled in **configureAssetBrowser()** for certain contexts. The PyMockAsset Asset example is included as:
`${KATANA_HOME}/plugins/Src/Resources/Examples/AssetPlugins/MockAsset.py`

The PyMockAssetWidgetDelegate example is included as:
`${KATANA_HOME}/plugins/Src/Resources/Examples/UIPlugins/PyMockAssetWidgetDelegate.py`

Bug Fixes

- BUG ID 35428 - After using **File** > **Save As...** to save a scene, Katana incorrectly indicated that the scene still had unsaved changes.
- BUG ID 35233 - Using a custom Asset plug-in and saving a scene incorrectly triggered the **Scene File Exists** override dialog.
- BUG ID 31910 - When the Asset plug-in was changed in the Project settings, Katana displayed an error if the current open scene was not compatible with the current Asset plug-in, even if the current scene was a valid file.
- BUG ID 31991 - Browsing for a LiveGroup asset from a node parameter widget used a different file filtering than the **Import LiveGroup...** menu item.
- BUG ID 32835 - The **Import LiveGroup** command resulted in a node with name set to **unknown**.
- BUG ID 34589 - **resolvePath()** was not consistently called for all render output types during Disk Renders.
- BUG ID 35155 - The import and export of LookFileManager settings failed. Import and export of LookFileManager settings now behave as expected, and support asset-managed import and export.

- BUG ID 35280 - The behavior Katana uses when searching for args files is now switchable. If the environment variable KATANA_LEGACY_ARGS is set, Katana searches for Args files using the technique used in Katana 1.1 (which is based on the shader filename instead of the shader name).

Known Issues & Workarounds

Asset API

- BUG ID 35334 - Katana gives an error loading a LiveGroup on Startup when using an Asset plug-in that was not previously selected.
- BUG ID 35321 - The PrmanObjectSettings node does not resolve an Asset ID in a **ribInclude** when using Debug Output.
- BUG ID 35318 - The OCIOFileTransform node is not asset-managed.
- BUG ID 32555 - The DefaultAsset, and FileSequence plug-ins must be set explicitly in a Custom Asset plug-in.

  If you want to always use your own default Asset plug-in, your plug-in must include a callback to set the Asset plug-in name, and file sequence name, every time you create a new scene:

```
from Katana import Callbacks
AssetPluginName = "PyMockAsset"
FileSequenceName = "PyMockFileSeq"
def setDefaultAsset(**kwargs):
   from Katana import AssetAPI
   from Katana import NodegraphAPI
   # Set the default Asset plug-in and file sequence
   # plug-in in the AssetAPI
   AssetAPI.SetDefaultAssetPluginName(AssetPluginName)
   AssetAPI.SetDefaultFileSequencePluginName(FileSequenceName)
   # Set the default Asset plug-in and file sequence
   # plug-in in the Katana file
   NodegraphAPI.GetRootNode().getParameter("plugins.asset").\
       setValue(AssetPluginName, 0)
   NodegraphAPI.GetRootNode().getParameter(
       "plugins.fileSequence").setValue(FileSequenceName, 0)
# Trigger this on startup and whenever a new scene is created
Callbacks.addCallback(Callbacks.Type.onStartup,
    setDefaultAsset)
Callbacks.addCallback(Callbacks.Type.onNewScene,
    setDefaultAsset)
```

  Place a variant of the above script in a directory called **Plugins** within a path listed in your KATANA_RESOURCES environment variable. Alternatively, you can add your script to an **init.py** placed in a directory named **Startup**, in the **.katana** directory found in your **home** directory.

Miscellaneous

- BUG ID 35447 - When performing an Interactive Render with Arnold with overscan, the overscan is incorrectly applied. Hot Renders are unaffected, and work as expected. To disable overscan for Interactive Renders only, add an InteractiveRenderFilter which sets overscan to zero.
- BUG ID 34950 - The RiOption **decimationrate** has changed to type **int**, but the PrmanGlobalSettings node still looks for a **float**.
- BUG ID 34669 - Currently **exrcopyrenderattrs()** fails to copy EXR header information when tile stitching.
- BUG ID 34367 - Division of integers is performed differently in the Python interpreter launched with the render process to the interpreter launched with Katana: integer (old) and floating point (new / future) respectively. This can cause discrepancies between values computed in the main Katana process, and those computed in the render process.

TIP: It is possible to force new style division on a per module basis using the following Python command:

```
from __future__ import division
```

- BUG ID 34289 - The Render Log output is incorrectly formatted, with superfluous newlines.
- BUG ID 33298 - When rerendering with PRMan, ending a rerendering session does not terminate the **prman** process.
- BUG ID 33242 - Interactive rerendering is not designed to work with rapid updates to region of interest (ROI), and attempting rapid ROI changes may cause unexpected behavior.
- BUG ID 32159 - The Alembic library does not support multiple process / thread access to an Alembic file. This means that modifying an Alembic file outside Katana, while it's loaded in an open Katana scene results in a Katana crash.

  To avoid this, either close any Katana scenes before modifying included Alembic files, or collapse the Scene Graph of any Katana scenes to **/root**, and click **Flush Caches** before attempting to update any modified Alembic files.

TIP: Right-click on **/root** in the **Scene Graph** tab and choose **Close All**.

- BUG ID 31432 - The **HotRender Deps before Interactive Render** toggled menu item does not currently apply to renders triggered using the **Render** > **Rerender Previous** command, or to renders triggered by going to the **Rerender Control** tab and clicking **Restart Rerendering**.
- Opening a PDF document from Katana, using the document viewer Evince, can fail under certain system configurations. Katana ships with Zlib version 1.2.5, which may be incompatible with your OS-packaged Evince. If you experience problems opening PDF files from Katana, make sure your LD_LIBRARY_PATH variable is configured to resolve a suitable version of Zlib.
- When rerendering in Arnold, motion blur does not behave as expected under the following circumstances:
  - If **makeInteractive=Yes**, and the Viewer is set to look through the camera.
  - When rendering through a Camera with an interactive transform applied. For example, a CameraCreate node with transforms, or a Camera with a Transform3D applied.

NOTE: For setups that don't support motion blur, the first frame rendered may show motion blur, although subsequent updates wont.

TIP: An example setup that displays correct motion blur behavior when rerendering in Arnold is rendering through a camera from an Alembic cache with animation, but no extra interactive transforms from within the scene.