# PyQt for Autodesk Maya 2016 64bit

Written by Cyrille Fauvel – Autodesk Developer Network (April 2013)
Updated by Vijaya Prakash – Autodesk Developer Network (May 2015)

## Building SIP, and PyQt for Maya 2016

PyQt [http://www.riverbankcomputing.co.uk] is a python binding to the Qt library. Because Maya uses Qt internally, you can use the PyQt modules in Maya python scripts to create custom UI. PyQt does not have the same licensing as Maya, Qt, or Python. Please consult the PyQt website for information about licensing for PyQt.

Download PyQt: http://www.riverbankcomputing.com/software/pyqt/download

Download SIP: http://www.riverbankcomputing.com/software/sip/download

The following are instructions for building a copy of the PyQt modules that have been known to work with Maya.

Maya 2016 uses Qt4.8.6 which is binary compatible with the latest version of PyQt - 4.11.3 / SIP - 4.16.7 (at time of writing, May 2015).

Note that it's important to use the Maya modified version of the Qt source code. A copy of the customized Qt 4.8.6 source is available from Autodesk's Open Source web-site (http://www.autodesk.com/lgplsource) and includes text files describing how to configure, build and install Qt for each platform supported by Maya.

However, be aware that with Maya 2016, there is no more need to build PySide as it is coming by default in Maya, nor you have to rebuild Qt itself as the main Qt tools to build PyQt are now included in the Maya distributions (I.e. qmake, moc, …)

libxml, openSSL, OpenAL, python2.7, qt-4.8.6, and tbb are also coming by default in the Maya include and lib folders, so unless you have a very specific need, you would not need to rebuild any of those libraries like before. Note as well that there is a 'MAYA_LOCATION/support/opensource' folder now which contains some of the community source.

**Important:** Maya 2016 now ships without the devkit, include and mkspecs folders. You can get the Maya 2016 devkit from the Autodesk Apps Exchange Store here for Windows, OSX, and Linux. Download the devkit and unzip the files into your Maya root folder. Make sure to read the instructions to install the devkit, include and mkspecs folders properly on your system.

The scripts used in this document are now also posted on Github.

Download SIP and PyQt source from 'http://www.riverbankcomputing.co.uk' - here I downloaded 'sip-4.16.7' and 'PyQt-win-gpl-4.11.3'. Unzip them in one folder, then you should get something like this:

# Mac

*/Users/cyrille/Documents/_Maya2016Scripts/sip-4.16.7*
*/Users/cyrille/Documents/_Maya2016Scripts/PyQt-mac-gpl-4.11.3*
*'/Users/cyrille/Documents/_Maya2016Scripts' being my local folder. Now the instructions, and bash scripts to build that SIP and PyQt.*

Follow the instructions from the API docs to setup your environment (Developer Resources > API Guide > Setting up your build environment > Mac OS X environment, in the Maya Documentation)

Untar the /devkit/include/qt-4.8.6-include.tar.gz into /devkit/include/Qt

Copy /Resources/qt.conf into /bin/qt.conf and edit it like this:

```
[Paths]
Prefix=
Libraries=../MacOS
Binaries=../bin
Headers=../../../devkit/include/Qt
Data=..
Plugins=../qt-plugins
Translations=../qt-translations
```

Untar the qt-4.8.6-mkspecs.tar.gz into $MAYA_LOCATION/Maya.app/Contents/bin/mkspecs.
Make sure the qconfig.pri looks like this:

**qconfig.pri**

```
#configuration
CONFIG += release def_files_disabled exceptions no_mocdepend stl
x86_64 qt #qt_framework
QT_ARCH = macosx
QT_EDITION = OpenSource
QT_CONFIG += minimal-config small-config medium-config large-
config full-config no-pkg-config dwarf2 phonon phonon-backend
accessibility opengl reduce_exports ipv6 getaddrinfo ipv6ifname
getifaddrs png no-freetype system-zlib nis cups iconv openssl
corewlan concurrent xmlpatterns multimedia audio-backend svg
script scripttools declarative release x86_64 qt #qt_framework

#versioning QT_VERSION = 4.8.6
QT_MAJOR_VERSION = 4
QT_MINOR_VERSION = 8
QT_PATCH_VERSION = 6

#namespaces
QT_LIBINFIX =
QT_NAMESPACE =
QT_NAMESPACE_MAC_CRC =
```

You also need to create copy of the Qt lib files as fake .dylib files from the /MacOS directory. The script below will give you the commands to do that.

## Build & Install SIP

```
#!/usr/bin/env bash
MAYAPYQTBUILD="`dirname \"$0\"`" # Relative
export MAYAPYQTBUILD="`( cd \"$MAYAPYQTBUILD\" && pwd )`" #
Absolutized and
normalized
cd $MAYAPYQTBUILD

export SIPDIR=$MAYAPYQTBUILD/sip-4.16.7
export MAYA_LOCATION=/Applications/Autodesk/maya2016

cd $SIPDIR
$MAYA_LOCATION/Maya.app/Contents/bin/mayapy ./configure.py --
arch=x86_64
make
sudo make install
```

## Build & Install PyQt

```
#!/usr/bin/env bash

MAYAPYQTBUILD="`dirname \"$0\"`" # Relative
export MAYAPYQTBUILD="`( cd \"$MAYAPYQTBUILD\" && pwd )`" #
Absolutized and
normalized
cd $MAYAPYQTBUILD

export MAYA_LOCATION=/Applications/Autodesk/maya2016
export QTDIR=$MAYA_LOCATION/Maya.app/Contents
export QMAKESPEC=$QTDIR/mkspecs/macx-g++
export INCDIR_QT=$MAYA_LOCATION/devkit/include/Qt
export LIBDIR_QT=$QTDIR/MacOS

if [ ! -f $QMAKESPEC/qmake.conf ];
then
  echo "You need to install qt-4.8.6-mkspecs.tar.gz in
$QTDIR/mkspecs
!"
  exit
fi
if [ ! -f $INCDIR_QT/QtCore/qdir.h ];
then
echo "You need to uncompress $MAYA_LOCATION/devkit/include/qt-
4.8.6- include.tar.gz in $INCDIR_QT !"
  exit
fi
# qt.conf -
/Applications/Autodesk/maya2016/Maya.app/Contents/Resources
if [ ! -f $QTDIR/bin/qt.conf ];
then
  echo "You need to copy $QTDIR/Resources/qt.conf in $QTDIR/bin !"
  exit
fi

test=`grep "Data=../.." $QTDIR/bin/qt.conf`
```

```
if [ ! -z "$test" ];
then
  echo "You need to edit $QTDIR/bin/qt.conf to use 'Data=..'"
exit
fi
test=`grep "Headers=../../include" $QTDIR/bin/qt.conf`
if [ ! -z "$test" ];
then
echo "You need to edit $QTDIR/bin/qt.conf to use
'Headers=../../../devkit/include/Qt'"
exit
fi
test=`grep "Libraries=../lib" $QTDIR/bin/qt.conf`
if [ ! -z "$test" ];
then
  echo "You need to edit $QTDIR/bin/qt.conf to use
'Libraries=../MacOS'"
  exit
fi
test=`grep "Plugins = qt-plugins" $QTDIR/bin/qt.conf`
if [ ! -z "$test" ];
then
  echo "You need to edit $QTDIR/bin/qt.conf to use 'Plugins=../qt-
plugins'"
  exit
fi
test=`grep "Translations = qt-translations" $QTDIR/bin/qt.conf`
if [ ! -z "$test" ];
then
  echo "You need to edit $QTDIR/bin/qt.conf to use
'Translations=../qt-
translations'"
  exit
fi

for mod in Core Declarative Designer DesignerComponents Gui Help
Multimedia Network OpenGL Script ScriptTools Sql Svg WebKit Xml
XmlPatterns
do
  if [ ! -f $QTDIR/MacOS/libQt${mod}.dylib ];
  then
    echo "You need to copy a fake Qt$mod dylib - cp
$QTDIR/MacOS/Qt$mod
$QTDIR/MacOS/libQt${mod}.dylib !"
    #cp $QTDIR/MacOS/Qt$mod $QTDIR/MacOS/libQt${mod}.dylib
  exit
fi
done
if [ ! -f $QTDIR/MacOS/libphonon.dylib ];
then
  echo "You need to copy a fake phonon dylib - cp
$QTDIR/MacOS/phonon
$QTDIR/MacOS/libphonon.dylib !"
  #cp $QTDIR/MacOS/phonon $QTDIR/MacOS/libphonon.dylib
  exit
fi
```

```
export DYLD_LIBRARY_PATH=$QTDIR/MacOS
export DYLD_FRAMEWORK_PATH=$QTDIR/Frameworks
export SIPDIR=$MAYAPYQTBUILD/sip-4.16.7
export PYQTDIR=$MAYAPYQTBUILD/PyQt-mac-gpl-4.11.3

cd $PYQTDIR
export PATH=$QTDIR/bin:$PATH
$QTDIR/bin/mayapy ./configure.py –use-arch x86_64
LIBDIR_QT=$LIBDIR_QT
INCDIR_QT=$INCDIR_QT MOC=$QTDIR/bin/moc -w --no-designer-plugin -g
make -j 8
sudo make install
```

You're done! go to the testing paragraph at the end of the article.

**Note:** In the shipping version of Maya 2016, PyQT will fail to execute because of a hidden phonon symbol. There is a QTBUG reference at https://bugreports.qt.io/browse/QTBUG-37209 which would support the change in compiler - exporting symbol issue. There is a Qt source change suggested for the CLANG build to export symbols. Visit the blog 'Around the corner' to get the fix or wait for the next service pack which may contain the fixed version.

# Linux

*/home/cyrille/Documents/_Maya2016Scripts/sip-4.16.7*
*/home/cyrille/Documents/_Maya2016Scripts/PyQt-mac-gpl-4.11.3*

*'/home/cyrille/Documents/_Maya2016Scripts' being my local folder. Now the instructions, and bash scripts to build SIP and PyQt.*

Follow the instructions from the API docs to setup your environment (Developer Resources > API Guide > Setting up your build environment > Linux environments (64 bit), in the Maya Documentation).

Edit your qt.conf file (/usr/autodesk/maya2016/bin) like below

```
[Paths]
Prefix=
Libraries=../lib
Binaries=../bin
Headers=../include/Qt
Data=../
Plugins=../qt-plugins
Translations=../qt-translations
```

Untar the /include/qt-4.8.6-include.tar.gz into /include/Qt

Untar the /mkspecs/qt-4.8.6-mkspecs.tar.gz into /mkspecs

Make qmake, moc executables from the Maya bin directory

```
sudo chmod aog+x /usr/autodesk/maya2016/bin/moc
sudo chmod aog+x /usr/autodesk/maya2016/bin/qmake
```

## Build & Install SIP

```
#!/usr/bin/env bash
MAYAPYQTBUILD="`dirname \"$0\"`" # Relative
export MAYAPYQTBUILD="`( cd \"$MAYAPYQTBUILD\" && pwd )`" #
Absolutized and normalized
cd $MAYAPYQTBUILD

export SIPDIR=$MAYAPYQTBUILD/sip-4.16.7
export MAYA_LOCATION=/usr/autodesk/maya2016

cd $SIPDIR
$MAYA_LOCATION/bin/mayapy ./configure.py
make
sudo make install
```

## Build & Install PyQt

```
#!/usr/bin/env bash
MAYAPYQTBUILD="`dirname \"$0\"`" # Relative
export MAYAPYQTBUILD="`( cd \"$MAYAPYQTBUILD\" && pwd )`" #
Absolutized and normalized
cd $MAYAPYQTBUILD

export MAYA_LOCATION=/usr/autodesk/maya2016
export QTDIR=$MAYA_LOCATION
export QMAKESPEC=$QTDIR/mkspecs/linux-g++-64
export INCDIR_QT=$MAYA_LOCATION/include/Qt
export LIBDIR_QT=$QTDIR/lib

if [ ! -f $QMAKESPEC/qmake.conf ];
then
  echo "You need to install qt-4.8.6-mkspecs.tar.gz in
$QTDIR/mkspecs !"
  exit
fi
if [ ! -f $INCDIR_QT/QtCore/qdir.h ];
then
  echo "You need to uncompress $MAYA_LOCATION/include/qt-4.8.6-
include.tar.gz in $INCDIR_QT !"
  exit
fi
# qt.conf -
/Applications/Autodesk/maya2016/Maya.app/Contents/Resources
if [ ! -f $QTDIR/bin/qt.conf ];
then
  echo "You need to copy $QTDIR/Resources/qt.conf in $QTDIR/bin !"
  exit
fi

test=`grep "Headers=../include/Qt" $QTDIR/bin/qt.conf`
if [ -z "$test" ];
then
  echo "You need to edit $QTDIR/bin/qt.conf to use
'Headers=../include/Qt'"
  exit
fi
```

```
export SIPDIR=$MAYAPYQTBUILD/sip-4.16.7
export PYQTDIR=$MAYAPYQTBUILD/PyQt-x11-gpl-4.11.3

cd $PYQTDIR
export PATH=$QTDIR/bin:$PATH
$QTDIR/bin/mayapy ./configure.py LIBDIR_QT=$LIBDIR_QT
INCDIR_QT=$INCDIR_QT MOC=$QTDIR/bin/moc -w --no-designer-plugin -g
make -j 8
sudo make install
```

You're done! go to the testing paragraph at the end of the article.

# Windows

*D:__sdkext_Maya2016 Scripts\sip-4.16.7*
*D:__sdkext_Maya2016 Scripts\PyQt-win-gpl-4.11.3*

*'D:__sdkext_Maya2016 Scripts' being my local folder. Now the instructions and scripts to build SIP and PyQt.*

Follow the instructions from the API docs to setup your environment (Developer Resources > API Guide > Setting up your build environment > Windows environment (64-bit), in the Maya Documentation)

Edit your qt.conf file (C:\Program Files\Autodesk\Maya2016\bin) like below

```
[Paths]
Prefix=
Libraries=../lib
Binaries=../bin
Headers=../include/Qt
Data=../
Plugins=../qt-plugins
Translations=../qt-translations
```

Unzip the /include/qt-4.8.6-include.tar.gz into /include/Qt

Unzip the /mkspecs/qt-4.8.6-mkspecs.tar.gz into /mkspecs

## Build & Install SIP

```
@echo off

set MAYAPYQTBUILD=%~dp0
set MAYAPYQTBUILD=%MAYAPYQTBUILD:~0,-1%
if exist v:\nul subst v: /d
subst v: "%MAYAPYQTBUILD%"

set SIPDIR=v:\sip-4.16.7
set MSVC_DIR=C:\Program Files (x86)\Microsoft Visual Studio 11.0
if ["%LIBPATH%"]==[""] call "%MSVC_DIR%\VC\vcvarsall" amd64

set MAYA_LOCATION=C:\Program Files\Autodesk\Maya2016
```

```
set INCLUDE=%INCLUDE%;%MAYA_LOCATION%\include\python2.7
set LIB=%LIB%;%MAYA_LOCATION%\lib

pushd %SIPDIR%
"%MAYA_LOCATION%\bin\mayapy" configure.py
nmake
nmake install
popd
```

## Build & Install PyQt

Note that the pyQT generated makefile are using the make macro <? while building 'pylupdate'. Unfortunately, it will hang forever on Windows, and will complain about an undefined macro <? ... This macro means to use the first build dependency file as argument, but nmake does not seems to like it. The script below will bypass the makefile rule, so it should work just fine.

```
@echo off

set MAYAPYQTBUILD=%~dp0
set MAYAPYQTBUILD=%MAYAPYQTBUILD:~0,-1%
if exist v:\nul subst v: /d subst v: "%MAYAPYQTBUILD%"
v:

set MAYA_LOCATION=C:\Program Files\Autodesk\Maya2016
if exist m:\nul subst m: /d
subst m: "%MAYA_LOCATION%"
set MAYA_LOCATION=m:

set QTDIR=%MAYA_LOCATION%
set MSVC_VERSION=2012
set QMAKESPEC=%QTDIR%\mkspecs\win32-msvc%MSVC_VERSION%
set _QMAKESPEC_=win32-msvc%MSVC_VERSION%
if not exist "%QMAKESPEC%\qmake.conf" (
  echo "You need to uncompress %MAYA_LOCATION%\mkspecs\qt-4.8.6-
mkspecs.tar.gz !"
  goto end
)
if not exist "%MAYA_LOCATION%\include\Qt\QtCore\qdir.h" (
  echo "You need to uncompress %MAYA_LOCATION%\include\qt-4.8.6-
include.tar.gz in %MAYA_LOCATION%\include\Qt !"
  goto end
)
findstr /L /C:"Headers=../include/Qt" "%MAYA_LOCATION%\bin\qt.conf"
>nul 2>&1
if ERRORLEVEL 1 (
  echo "You need to edit %MAYA_LOCATION%\bin\qt.conf to use
'Headers=../include/Qt'"
  goto end
)

set SIPDIR=v:\sip-4.16.7
set PYQTDIR=v:\PyQt-win-gpl-4.11.3

set MSVC_DIR=C:\Program Files (x86)\Microsoft Visual Studio 11.0
```

```
if ["%LIBPATH%"]==[""] call "%MSVC_DIR%\VC\vcvarsall" amd64

set INCLUDE=%INCLUDE%;%MAYA_LOCATION%\include\python2.7
set LIB=%LIB%;%MAYA_LOCATION%\lib

pushd %PYQTDIR%

pushd pylupdate
del moc_translator.cpp 2> nul
del moc_translator.obj 2> nul
%QTDIR%\bin\moc.exe -o moc_translator.cpp translator.h
popd

set PATH=%QTDIR%\bin;%PATH%
"%MAYA_LOCATION%\bin\mayapy" configure.py -p %QMAKESPEC%
LIBDIR_QT="%QTDIR%\lib" INCDIR_QT="%QTDIR%\include\Qt"
MOC="%QTDIR%\bin\moc.exe" -w --no-designer-plugin
nmake
nmake install
popd

:end
```

You're done! go to the testing paragraph at the end of the article.

## Testing

Copy and paste this example in the Maya Script Editor (in a Python tab), and execute the code:

```
import sys
from PyQt4 import QtGui

class Example(QtGui.QWidget):
  def __init__ (self):
    super(Example, self).__init__ ()
    self.initUI()

  def initUI(self):
    self.btn = QtGui.QPushButton('Dialog', self)
    self.btn.move(20, 20)
    self.btn.clicked.connect(self.showDialog)

    self.le = QtGui.QLineEdit(self)
    self.le.move(130, 22)

    self.setGeometry(300, 300, 290, 150)
    self.setWindowTitle('Input dialog')
    self.show()

  def showDialog(self):
    text, ok = QtGui.QInputDialog.getText(self, 'Input Dialog',
'Enter your name:')
    if ok:
      self.le.setText(str(text))

ex = Example()
```

If you see the dialog is showing, you are all set.