Technical Note TN1197

QuickTime 4.1.1/4.1.2

CONTENTS

QuickTime 4.1.1/4.1.2 can be described as a minimal-changes releases that provide a number of miscellaneous bug fixes.

This Technote describes the changes made between the release of QuickTime 4.1 and the update releases of QuickTime 4.1.1 and 4.1.2.

Updated: [May 01 2000]

QuickTime 4.1.1 Changes

Listed below are many of the important changes included in the QuickTime 4.1.1 release:

- Fixed an AppleScript bug for QuickTime Player where the frontmost window was not getting targeted by default.
- Fixed an AppleScript bug for QuickTime Player where the "close every window" action would fail with an error.
- Added Windows 2000 support.
- Improved the Sequence Grabber to cope better when compressing to a different-sized format. This means that using DV video as an input to Sorenson Broadcaster is now viable (we still recommend broadcasting Built-In audio rather than DV audio).
- If you use the `QDesign` codec for streaming, you should know there are significant improvements in the transmission of both live and stored (video on demand) streams. These improvements are backwards-compatible with earlier clients. In particular, those using `QDesign` for live broadcasting should update as soon as possible, and make sure they are using the `QDesign` (not QuickTime) packetizer. Those with hinted movies may find some improvement in the data rate, after re-hinting the movies.

Note:
Those using the QuickTime packetizer for *any* stream at this point are urged to reconsider. This packing has no loss resilience and poor use of network bandwidth. It was designed as a catch-all for codecs without a good packing. In particular, its use for PureVoice is almost certainly a mistake. If you are using it for PureVoice because you are using a sampling rate other than 8kHz, you should know that the codec was designed at this rate and may not work as effectively at other rates, and the packing assumes this is the rate, thus forcing the QuickTime packetizer for any other sampling rate.

- Improved out-of-order packet handling when watching streaming content.
- Added audio/mpegurl and audio/x-mpegurl mime types to the QuickTime Plug-In to handle M3U files (lists of MP3's).
- Added Streaming Transport to the QuickTime Player Edit/Preferences Menu.
- Added SMIL "qt:chapter-mode" tag, which was documented, but missing in QT 4.1.
- Exporting movies with no audio or video track now works correctly.
- Export of Flash to DV bug fix.
- URL links in a web page for QuickTime VR movies that rely on the HOTSPOT parameter of an <EMBED> tag now work correctly.
- Fixed bug where `SoundConverterGetBufferSizes` would occasionally return the wrong destination number of bytes.
- System no longer hangs when opening picture files on a Kodak Picture CD (multi-session ISO-9660).
- "Check for QuickTime Updates" option in QuickTime Player now works while playing a movie.
- Incorporated Sound Manager audio/video synchronization improvements.
- Image sequence reference movies now get temporally compressed.
- Sequence Grabber DV preview with bottleneck procedure now works correctly.
- Better support for streaming through various firewalls/proxies.
- Added support for .sml as a file extension for SMIL, and removed the QuickTime Exchange .smi conflict with self-mounting images on Mac.
- Movies made by importing a SMIL file now play correctly no matter what platform they were made on or viewed on, provided the file was originally imported by QuickTime 4.1.1 (not QuickTime 4.1).
- Overall improvements to SMIL import.
- Added `SGGetDataOutputStorageSpaceRemaining64` API, which was missing in QT 4.1.
- Added support for a couple of TIFF variants we weren't handling:
  1) uncompressed single-strip TIFF files with missing StripByteCounts fields and
  2) 8-bit RGB palette TIFFs that have an 8-bit alpha value associated with every pixel.

Back to top

QuickTime 4.1.2 Changes

Listed below are many of the important changes included in the QuickTime 4.1.2 release:

- Fixed parsing of "mailto:"-style URLs, and URLs containing characters such as "?".
  (**Note:** The above fix was made by simply backing out a QT 4.1.1 change that fixed "going to relative URLs from sprite actions in QTPlayer." This means that "going to relative URLs from sprite actions in QTPlayer" is now back to its previous (pre-4.1.1) state: it doesn't work. We'll take a better shot at this next time.)
- Fixed a QT 4.1.1 bug where overrides in an <EMBED> tag (such as "CONTROLLER=FALSE") were being reset to the movie's authored value during a QTNEXT.
- Avoid a QT 4.1.1 crash that could occur when certain applications passed garbage GWorlds to the `DisposeGWorld` function.
- Fixed a QT 4.1.1 sequence grabber bug where some USB YUV cameras would display a bad preview image when capturing if the preview window was clipped.

Back to top

Detecting the Presence/Version of QuickTime from Web Browsers Windows

QuickTimeCheck.ocx

Included in QuickTime for Windows 4.1.1 is the file QuickTimeCheck.ocx, a Windows-only DLL that implements an "automation object", which is essentially a scriptable object for embedding in a Windows application like Internet Explorer. This particular object gives HTML authors in Internet Explorer for Windows the ability to detect the presence of QuickTime and determine its version.

Unlike ActiveX controls, this kind of object doesn't have any user interface. Instead, a web developer using VB Script (or

JScript in Internet Explorer) will create an instance of the object and make direct requests to the object. A web developer must use a VB Script embedded in a web page to create and to communicate with the object.

Checking for QuickTime

To detect the presence of QuickTime, first create the object, check if the object creation succeeded and check if the QuickTime extension (the file QuickTime.qts) is available. The following VB Script shows how this is done:

```
Dim theObject

On Error Resume Next 'Do not report runtime error if object itself was
not installed
Set theControl = CreateObject("QuickTimeCheckObject.QuickTimeCheck.1")
On Error Goto 0      'Allow runtime errors

If IsObject(theObject) Then
If theObject.IsQuickTimeAvailable(0) Then
MsgBox "QuickTime is available!"
End If
End If
```

There's another flavor of the `IsQuickTimeAvailable` method which can also be used. Passing the value 1 instead of 0 to the `IsQuickTimeAvailable` method will fully initialized QuickTime instead of just checking for the QuickTime.qts file in the Windows system directory. The disadvantage to passing the value 1 to the `IsQuickTimeAvailable` method is it can take a long time to complete. For this reason, we strongly encourage developers to pass 0 instead.

Shown below is HTML code with an embedded VB Script which makes use of the QuickTimeCheck.ocx object to check for the presence of QuickTime. If QuickTime is detected, the code simply displays a message stating QuickTime was found. Similarly, if QuickTime is not detected, the code displays a message stating QuickTime could not be found. However, if you find users that don't have QuickTime, you'll probably want to suggest they install it in order to get the best possible user experience. You can use a web badge on your web site to provide an active visual link to Apple's QuickTime site. See Apple's web badges page for more information on web badge usage.

```
<HEAD>
<TITLE>Test for QT</TITLE>
<SCRIPT LANGUAGE="Javascript">
<!-- hide from pre-script browsers
  var haveqt = false;
//-->
</SCRIPT>
<SCRIPT LANGUAGE="VBScript">
<!-- hide from pre-script browsers
    On Error  Resume Next
       Set theObject = CreateObject("QuickTimeCheckObject.QuickTimeCheck.1")
    On Error  goto 0

    If IsObject(theObject) Then
      If theObject.IsQuickTimeAvailable(0) Then   'Just check for file
        haveqt = true
      End If
    End If
//-->
</SCRIPT>
<SCRIPT LANGUAGE="Javascript">
<!-- hide from pre-script browsers
  if (navigator.plugins) {
    for (i=0; i < navigator.plugins.length; i++ ) {
      if (navigator.plugins[i].name.indexOf("QuickTime") >= 0)
        { haveqt = true; }
      }
    }
//-->
</SCRIPT>
</HEAD>
<BODY>
<SCRIPT LANGUAGE="Javascript">
<!-- hide from pre-script browsers
  if (haveqt)
    {document.write("You have QuickTime!!!");}
    else
    {document.write("You don't seem to have QuickTime");}
//-->
</SCRIPT>
<NOSCRIPT>
Man, you don't even have Scripting!
</NOSCRIPT>
<NOEMBED>
You aren't going to see QT if you don't allow EMBEDs!
</NOEMBED>

<H1>Check for QuickTime</H1>
</BODY>
</HTML>
```

Note:
The above HTML code has been carefully written to work on a variety of browsers and platforms. You are, of course, free to make your own changes to this code, but please understand any changes you make may cause the code to work incorrectly. Also, if you reference any media files in your code using an < EMBED > tag, you may want to include the parameter TYPE="video/quicktime" to insure the file is handled by the QuickTime plug-in.

In devising your QuickTime detection strategy, probably the best way to proceed is to first use a basic script like the one shown above at the HTML level, and alternately use QuickTime sprites, or other techniques, to do more sophisticated checking if necessary.

Finally, remember QuickTimeCheck.ocx is only installed with QuickTime 4.1.1. Therefore, the above code will report that QuickTime is not present if it is run on systems that do not have QuickTime 4.1.1.

Version Information

It's now possible to determine the particular version of QuickTime installed using the QuickTimeCheck.ocx object. The following line of code:

```
Set myQTVersion = theObject.QuickTimeVersion
```

will return in `myQTVersion` an integer value corresponding to the version of QuickTime installed (the same value you would get when calling the Macintosh Toolbox `Gestalt` function with the `gestaltQuickTime` selector - see Inside Macintosh: Operating System Utilities / Chapter 1 - Gestalt Manager for more information).

For example, a decimal return value of `68321280` would indicate a release version of QuickTime 4.1.2. Seen in hexadecimal, this number appears as `0x04128000`, which we can identify as version 4.1.2. The `80` in this hexadecimal value indicates this is a release version (again, refer to the Gestalt Manager documentation for more information).

Note:
The QuickTimeVersion method calls `IsQuickTimeAvailable(1)` internally, so it can also be expensive to make this call if `IsQuickTimeAvailable(1)` hasn't already been called.

The QuickTimeCheck.ocx object works on Windows 95/98, Windows NT 4, and Windows 2000. It is installed when a QuickTime basic installation is performed, which means it will always be available.

Back to top

Using Gestalt to get the QuickTime Version

As always, the standard way to determine which version of QuickTime is installed is to call the Macintosh Toolbox API `Gestalt` function. The listing below shows a code snippet demonstrating how to check the version of QuickTime.

```
{
     /* check the version of QuickTime installed */
     long version;
     OSErr result;

     result = Gestalt(gestaltQuickTime,&version);
     if ((result == noErr) && (version >= 0x04128000))
     {
          /* we have version 4.1.2! */
     }
}
```

Summary

You are encouraged to review this Technote and related information to help implement changes in your code to take advantage of the added features and bug fixes in QuickTime 4.1.1 and 4.1.2.

Back to top

References

Inside Macintosh: QuickTime

Inside Macintosh: Gestalt Manager

Back to top

Downloadables

 Acrobat version of this Note (K). Download

Back to top
