Poking at iTunes

A developer's guide to the iTunes platform

Chris J. Karr

One comment: *No wireless. Less space than a nomad. Lame.* Rob "CmdrTaco" Malda introduced the iPod to the Slashdot crowd with a statement rivalled only by Bill Gates' quip "640 KB should be enough for anybody".

Since that post in 2001, Apple's iPod quickly became one of the most successful products in consumer electronics history. While its success largely derives from its "hip" factor and stylish design, the iPod's integration with the iTunes music application and the iTunes Music Store has made the device a favorite among music listeners.

The iPod is a single component of Apple's larger music platform.

The hub of the platform is the iTunes desktop digital music player. The iTunes application is only one of many in a long series of digital music library applications. iTunes began existence as a classic MacOS application called SoundJam. When Apple began developing and promoting their digital hub strategy, Apple purchased SoundJam and rechristened it iTunes. Prior to the introduction of the iPod, iTunes was simply another Mac MP3 program.

In 2001, Apple released the iPod. The device differed from other portable digital music players because of the tight integration with iTunes. Since other portable digital music players connected to applications other than those provided by the devices' vendors, no other player was as tightly coupled with a desktop application as the iPod. While this close integration created a disadvantage where the iPod depends fully upon iTunes, Apple used the opportunity to craft the

Fig. 1: Both the music store and music sharing functionality are visible in this instance of iTunes

iPod as an extension to desktop-bound audio collections. Users can create static and dynamic playlists in iTunes, and the iPod synchronizes itself with those lists upon connection to the host computer.

While the iPod and iTunes originated as Mac-only products, in 2003, Apple expanded its strategy and ported iTunes to Windows and added USB functionality to the iPod. This opened the product's market to Windows users. In short order, the iPod and iTunes dominated the market for portable hardware players and digital music applications.

Later in 2003, Apple expanded the iTunes platform by introducing the iTunes Music Store. Previous online music

stores had struggled with balancing copyright protection and user convenience. But Apple made the iTunes Music Store a success by striking new deals with copyright holders and leveraging the tight integration with iTunes. Furthermore, the iTunes Music Store is only accessible from within iTunes and iTunes (via Quicktime) is the only desktop application legally allowed to play protected tracks purchased from Apple's online store. While critics had initially predicted the failure of the online music store, due to the difficulties inherent with Digital Rights Management (DRM) technology, the store succeeded and flourished due to the fine balance that Apple was able to maintain, between copyright holders and users. Apple is capable of maintaining this balance because it controls all levels of the technology platform used to acquire and play the protected tracks.

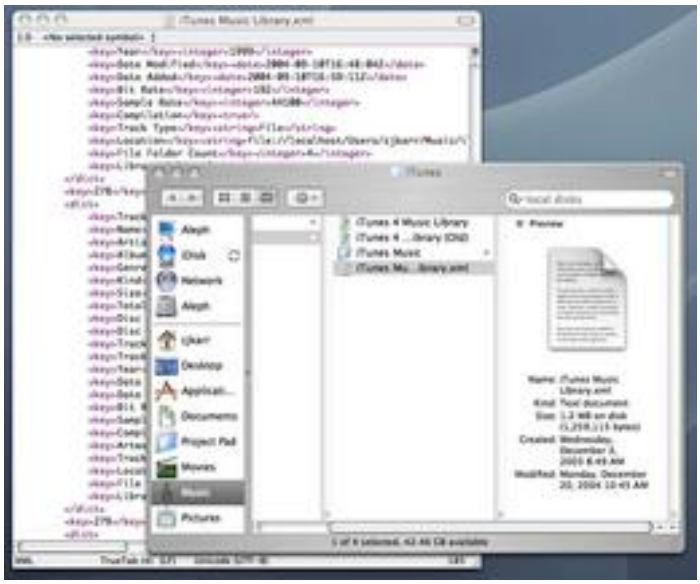
The iPod is a single component of Apple's larger music platform

In 2004, Apple further expanded the iTunes music platform with the introduction of AirPort Express base stations. The Airport Express device is a smaller version of Apple's Airport wireless access hardware. The Airport Express had also improved upon the older designs by including a dedicated USB hub, for printers and audio output ports, allowing connection between standard stereo equipment and wireless networks. To communicate with the Airport Express audio features, iTunes was given new functionality that sent audio to local Airport Express units. With this feature, iTunes users can use connected stereo devices to listen to their protected audio tracks. In addition to the analog stereo output port, Apple also integrated a digital 5.1 surround sound output into the device. (However there are currently no applications or content that can use the 5.1 surround sound feature as Apple has restricted access to third-party developers'.)

While Apple continued to expand the iTunes platform every year since its release, numerous third-party developers also created applications that integrate and interoperate with iTunes. Creative examples include music quiz applications that use the users' local iTunes libraries as a source for questions, and applications that assist users creating and refining metadata by providing connectivity to new data sources.

While Apple continues to restrict developer access to the Airport Express peripherals, it documented iTunes XML and AppleScript dictionaries for third-party developers to use when extending the iTunes platform. Furthermore, due to the efforts of persistent developers, undocumented portions of the platform (notably, music sharing protocols) are now better understood. Developers use this knowledge, official and otherwise, to create new applications that take advantage of the iTunes platform.

Fig. 2: The location and contents of the iTunes XML file



Programming with iTunes – XML, AppleScript, and Windows

Out of all the components in the Apple music platform, the iTunes application offers the most options for interested developers.

An XML file provides access to most of the iTunes metadata. Other applications can control iTunes with AppleScript (on the Mac) or the iTunes COM SDK (on Windows). Client and server applications can communicate with iTunes using the Digital Audio Access Protocol (DAAP). Finally, protected AAC files purchased from the online music store can be played using Apple's Quicktime API.

Writing applications that can read the iTunes library XML file provides the easiest route for adding new functionality. This file is located at the root of a user's music collection and is named "iTunes Music Library.xml". This file uses an Apple subset of XML. It begins with metadata about the iTunes application and continues with a list of track entries

and playlist definitions. The individual track entries contain metadata like creators, albums titles, and genres. The file also contains automatically generated metadata such as the last play date and local file locations. With the exception of album cover art, metadata encoded in individual tracks is also part of the XML file and is accessible to applications that read that file. Following the song entries, the file contains XML descriptions of the local playlists.

In addition to making its metadata available via XML, iTunes have also provided a rich AppleScript dictionary. MacOS X applications use AppleScript to automate common tasks or to control other applications. Users write AppleScripts in an English-like syntax that is designed for non-programmers.

AppleScript also provides access to frameworks such as Cocoa, allowing users to write standard GUI applications entirely in AppleScript. Furthermore, AppleScripts can be invoked from the command line or developed interactively with a bundled AppleScript interpreter called Script Editor. This application is a specialized environment for developing and testing AppleScript code. Developers access locally installed applications' AppleScript dictionaries from Script Editor. Using the iTunes dictionary, developers can discover the functions that the iTunes application provides via AppleScript.

Apple's approach to the DAAP developer community probably reflects a general uncertainty within the company about releasing the technical details of their popular applications

A script that instructs iTunes to start playing can be written in a single line:

```
tell application "iTunes" to play
```

With more code, finer control of the application is available:

```
tell application "iTunes"
       play track 13 of user playlist
         "Sparkle and Fade"
end tell
```

The AppleScript language also includes control statements:

```
tell application "iTunes"
       if player state is stopped then play
end tell
```

In addition to playback, iTunes also provides access to its encoding functionality:

```
tell application "iTunes"
       set preferred_encoder to current encoder
       set available_encoders to (name of encoders)
       repeat with anEnc in available_encoders
             if format of encoder anEnc is "AAC" then
                   set current encoder to encoder anEnc
                   exit repeat
             end if
       end repeat

       -- (Comment) do stuff

       set current encoder to preferred_encoder
end tell
```

Numerous resources are available online that describe in more detail the mechanics of iTunes and AppleScript.

In addition to Script Editor, AppleScript developers can run and compile their AppleScripts from the command line using the `osascript` and `osacompile` utilities.

These allow developers to use AppleScript in the same manner as traditional Unix scripting languages.

Finally, developers can use AppleScript within their own applications. Apple provides convenient access to AppleScript via the Cocoa API. In Objective-C, code for executing an AppleScript follows the form:

```
// asCode is a NSString object containing AppleScript
// instructions.

appleScript = [[NSAppleScript alloc]
                initWithSource:asCode];
[appleScript executeAndReturnError:nil];
```

Using these API calls, applications use AppleScript without the use of external script files.

When using AppleScript, there are many ways third-party applications can work with iTunes. A CD cataloging application can retrieve album data from iTunes and use that data to initialize a collection. Applications like iPhoto and iMovie use iTunes as a media resource for audio clips when creating home movies and DVDs.

Helpers to chat clients that query iTunes for the current track are common. Computer-based telephony applications can

Fig. 3: The iTunes AppleScript dictionary as seen from the Script Editor application

pause iTunes when an incoming call is answered and resume playing at the completion of a call. A particularly innovative application might use Apple's iSight web camera to determine whether a user is present and play some favorite music when a user is present.

Unfortunately, while iTunes on the Mac is accessible from an AppleScript standpoint, its Windows counterpart does not contain the same level of integration. This is due to the lack of a system-wide scripting architecture on the Windows platform. However, Apple provides a Windows-based Software Development Kit (SDK) for iTunes.

Unfortunately, while iTunes on the Mac is accessible from an AppleScript standpoint, its Windows counterpart does not contain the same level of integration

This allows Windows applications to interoperate with iTunes. The examples provided by the Windows SDK are Visual Studio Jscript files. The syntax of Jscript differs significantly from AppleScript:

```
// This code deletes tracks with empty locations.

var iTunesApp =
  WScript.CreateObject("iTunes.Application");
var mainLibrary = iTunesApp.LibraryPlaylist;
var tracks = mainLibrary.Tracks;
var numTracks = tracks.Count;

while (numTracks != 0)
{
        var currTrack = tracks.Item(numTracks);


        if (currTrack.Location == "")
        {
                // yes, delete it
                currTrack.Delete();
               deletedTracks++;
        }

        numTracks--;

}
```

While AppleScript remains the more popular tool, this may change in time given the growing number of Windows iTunes users and developers writing new applications using the iTunes COM SDK

Sharing and streaming with DAAP and iTunes

With the release of iTunes 4.0, Apple included music sharing between running iTunes instances on local networks. The original implementation allowed users to find shared libraries on local subnets and access remote libraries by entering the IP addresses of remote servers. In a later release, Apple removed the remote sharing functionality because of copyright concerns, but left the local sharing intact.

When the application is configured to enable sharing, iTunes announces the availability of the music service via Rendezvous. Other local clients receive this announcement and update their interfaces to display the newly shared library. When clients connect to the sharing hosts, the communication between the client and server uses the Digital Audio Access Protocol (DAAP).

DAAP is a protocol similar to HTTP that has been reverse engineered by Mac enthusiasts. While some of the programmers involved in documenting the protocol have received communications from Apple employees promising official DAAP specifications, presently no official documents exist. Despite the lack of documented specifications, the protocol is sufficiently understood that developers have successfully created servers and clients that interoperate with iTunes.

Developers desiring DAAP functionality within their applications can use a number of free software options. The `libdaap` library is used to create both clients and servers. The `daapd` server is used for sharing content. OpenDAAP

is another library for writing clients and servers, while the mt-daapd project produces a DAAP server that is multi-threaded and is less dependent upon external libraries.

For the developers wishing to work directly with the DAAP protocol, knowledge of HTTP is essential.

Not only does DAAP use a request format similar to HTTP, functions like user authentication use the same types of protocols and algorithms.

A sample DAAP request:

```
GET /server-info HTTP/1.1
Host: daap.example.com
Accept: */*
User-Agent: iTunes/4.0 (Macintosh; N; PPC)
Client-DAAP-Version: 1.0 Accept-Encoding: gzip
```

And the response:

```
HTTP/1.1 200 OK
Date: Thu, 01 May 2003 18:00:28 GMT
DAAP-Server: iTunes/4.0 (Mac OS X)
Content-Type: application/x-dmap-tagged
Content-Length: 122
Content-Encoding: gzip

[Byte stream continues\ldots{}]
```

While providing full DAAP documentation appears to be in Apple's best interest, the company has not yet committed to providing the full details of the protocol. This problem manifested itself during April 2004, when DAAP application authors discovered that Apple tweaked the authentication protocol from an MD5-based algorithm to a custom hash table algorithm. This broke numerous DAAP implementations and developers cried foul. Later, developers discovered that DAAP is a subset of a larger protocol called Digital Media Access Protocol (DMAP). DMAP provides portions of the photo-sharing capabilities in iPhoto 4 in addition to portions of the iTunes music sharing.

Apple's approach to the DAAP developer community probably reflects a general uncertainty within the company about releasing the technical details of their popular applications. Long criticized for its "not invented here" syndrome, Apple made progress establishing itself as a good-faith participant in the open standards and source community with technology like Darwin and Rendezvous. While reasonable people can be sympathetic to the company for withholding details in order to reserve a measure of freedom for later innovations, iTunes is a mature product and developers wish to write applications that extend the platform. If Apple decided to publish the details of DMAP and DAAP and then committed to making those standards open and public, companies such as Tivo who build complementary products to iTunes and iPhoto would be able to use existing protocols and standards rather than reinventing their own. (Tivo's sharing functionality is currently implemented using another program that utilizes another non-standard HTTP-based sharing protocol.) If third party product developers and consumers were confident that these standards were documented and stable, then software developers and equipment manufacturers could create extensions to the iTunes platform that would enhance its overall value.

In addition to Script Editor, AppleScript developers can run and compile their AppleScripts from the command line using the osascript and osacompile utilities

On platforms where market demand is not sufficient for motivating the creation of commercial products, free software developers would be free to participate and also contribute to the platform's aggregate functionality. Why Apple continues to refuse to document DAAP in light of these considerations remains a mystery. It may be an indicator that Apple plans to further change the DAAP protocol and continue to innovate, but it may also be a matter of simple institutional dysfunction lingering from its "not invented here" days.

Quicktime and FairPlay

While Apple's music store has been a success, it has not been without controversy. In order to reach distribution agreements with copyright holders, Apple is required to include digital rights management technology with every track sold.

The tracks sold on the iTunes Music Store are standard AAC audio files with embedded FairPlay protection technology. While Apple's DRM is considered less oppressive than competing alternatives, it still places restrictions upon the legal and reasonable use of the purchased tracks. For

example, while users can burn the tracks to compact disc, they are limited in the number of times a particular order of songs can be burned to multiple discs. Furthermore, while tracks can be shared among computers, users are limited to five computers that can be authorized to play the tracks.

AppleScript also provides access to frameworks such as Cocoa, allowing users to write standard GUI applications entirely in AppleScript

In response to these restrictions, a number of developers have created applications capable of removing the DRM components of the files.

This allows users to transcode the files in other formats and removes the limitations on playback. While this type of reverse engineering is illegal in countries such as the United States, these projects have continued underground by constantly changing the location of hosting servers and by distributing the software anonymously. For users in countries that prohibit the breaking or removal of DRM technology, applications can still be written that permit the legal playback of protected tracks. The drawback is that applications are required to use the Quicktime APIs and frameworks. Platforms without official Quicktime or iTunes implementations are excluded. This places a severe limitation on the users of free software operating systems, who wish to respect the copyright holders' rights while listening to their purchased music on alternative platforms.

Before Quicktime can play the protected tracks, the user must authorize the computer for playback using iTunes. After the computer has been authorized, Mac users can use the standard Cocoa and Carbon APIs to play protected files. To play a protected track using Cocoa, variations on the code below are used:

```
NSURL * url =
  [NSURL URLWithString:@"file:///path/to/file.m4p"];

NSSound * s =
  [[NSSound alloc] initWithContentsOfURL:url
byReference:NO];

[s play];
```

Similar functionality is available on Windows using the Quicktime SDK.

Fig. 4: The DAAPD home page (`http://www.deleet.de/projekte/daap/`)

While these methods provide playback functionality, they do not allow developers to access the raw audio sampling data and it's doubtful that this policy will change in the current copyright environment.

"Programming" the iPod

While developers can write applications that interoperate and use iTunes metadata, options for programming the iPod are much more limited.

While developers can write applications that interoperate and use iTunes metadata, options for programming the iPod are much more limited

While Apple does not offer a way to develop applications for the device, some functionality can be added to the device using its hypertext notes.

iPod notes use a subset of HTML tags and are limited to a size of four kilobytes. Titles can be set using a `<title>` tag, while line and paragraph breaks are created using the `<br>` and `<p>` tags. Links to files are created using standard `<a href="...">` tags. Note files that end in the `.link` prefix contain solitary links that automatically redirect to a particular note or song. Files ending in the `.linx`

extension appear to the iPod user as standard folders filled with links. This is not unlike directory pages generated by web servers when index files are missing.

In addition to the note files, the iPod also supports a museum mode that forces the device to boot into the note reader and prohibits the user from exiting the note reader application. This is often used to limit the user's access to other files and audio tracks on the iPod while presenting the audio in a proper context. This is useful in settings where the iPod is used as a supplemental educational tool.

For the developers wishing to work directly with the DAAP protocol, knowledge of HTTP is essential

While this is a simple format for text presentation that lacks many of the features normally associated with true application development, creative developers can produce the illusion of applications. This can be accomplished by using real desktop applications to generate a set of linked iPod notes that simulate a real application. For example, a developer can write a desktop application that reads weather data from an online source, generates a linked note set, and then copies those files to the iPod. To the iPod user, this would appear to be a genuine weather application. In another case, a read-only cookbook program can be constructed that generates a variety of subject and index notes that link to individual recipes. While this form of programming is much closer to static web development than real application programming, it does present creative developers with an opportunity to expand the iPod's basic functionality.

Conclusion

There are a number of contradictions for developers, within the iTunes platform. The application metadata and track information is easily accessible via XML. Audio sharing remains officially undocumented and developers find themselves at the mercy of future protocol changes. Robust AppleScript support is provided for fine-grained application control. However, Apple severely limits the ways and environments within which, protected audio tracks may be accessed. In some parts of the platform, Apple makes it simple to engineer compatibility and interoperability, while in other parts development is a black art fraught with peril.

On the basis of its advanced functionality and growing user base, the iTunes platform presents a tempting target for developers. Apple should be applauded for its efforts to encourage independent developers to embrace the platform, while at the same time it should be criticized for its lack of transparency and refusal to provide the necessary code and documentation that allow developers to create new players and network clients on platforms that are continually neglected by commercial desktop software vendors.

Bibliography

[1] Adams, Doug; Doug's AppleScripts for iTunes (http://www.malcolmadams.com/itunes/itinfo/info01.php), 2004

[2] Apple Computer; iTunes COM for Windows SDK (http://developer.apple.com/sdk/itunescomsdk.html); Apple Computer, 2004

[3] Apple Computer; iPod Note Reader User Guide (http://developer.apple.com/hardware/ipod/ipodnotereader.pdf); Apple Computer, 2004

[4] McArthur, Sandy; DAAP (http://sandy.mcarthur.org/DAAP); 2004

Copyright information

About the author

Chris Karr is the lead developer of the Books (http://books.aetherial.net/), a free software MacOS X application