

Managing Your iPod With Smartlists

April 2009 Edition



Charles Hannum Jr
2004-2009

Introduction

What this document does

This document describes a system of linking smartlists that allow you to create self-updating music mixes according to weights and rules you decide. The particular examples used are derived from my own creations, but, if you understand what I've done, you can easily tailor things to suit your tastes.

Why you might care

iTunes' smartlist system is what truly separates the iPod from the rest of the market. Forget the style, forget the simplicity of use (which largely arises from the iTunes software anyhow), and think about taking greater control of what you listen to in a way that takes very little effort on your part. Once you understand how to link smartlists to create automatic music mixes, you can craft self-generating playlists that turn your iPod into a series of cyber DJs that play music tailored to you without very little to no further effort on your part.

In addition to the "cyber DJ" aspect of smartlists, they can also serve an important function for those who have more music than fits on their iPod. If this describes you, then you know you will need some sort of system to rotate music on and off your iPod. Sure, you could just manually manage everything, but wouldn't it be nicer if most or even all of this rotation was automated?

Damn, this is a lot of reading; why not just use the "Shuffle Songs" feature of the iPod?

The "Shuffle Songs" feature, though idiot proof, is relatively weak. Although there have been some improvements, notably the ability to set a "Skip while shuffling" flag for files you don't want shuffled, there are still shortcomings to the "Shuffle Songs" feature that aren't eliminated simply because you can now tell the iPod not to shuffle your comedy albums and sound effect clips.

1. Shuffling is blind, deaf, and dumb. When you shuffle songs, the iPod simply goes through all of the audio files eligible for shuffling in a more or less random manner. Some file types and media kinds are automatically excluded from shuffling, but these are hard coded by Apple. All other exclusions from shuffling must be manually flagged in iTunes. This binary yes/no to shuffling is the beginning and end of your primary control of the music mix. If you want to hear more Eminem than Mozart, too bad, what gets shuffled is at the mercy of Apple's shuffling algorithm.

Now, you can take control what gets shuffled by carefully limiting what's available in the first place either by only shuffling within a user created playlist or carefully controlling the ratio of what gets put on your iPod in the first place (e.g. 20% Jazz, 40% Classic Rock, 10% 1980s Pop, 30% Hip Hop & Rap). Those aren't terrible solutions, but they are laborious to set up and maintain. The beauty of this document is it explains how to automate such a process and polish it far beyond anything you can do manually.

2. Shuffling is cognitively impaired. Shuffling is not purely random and has some sort of weighting and/or kludges in the algorithms for how it chooses music. This might be fine except that it lacks a persistent memory state: it resets its shuffle state back to zero after every reset or sync. The result is that you hear some songs repeatedly while never hearing others. You can wind up hearing mostly the same 400 songs in slightly different arrangements again and again while never hearing much of the other 4000 songs on the iPod.

iTunes is the solution, not the problem

Your iPod was *designed* to work with iTunes, and iTunes was *designed* to work with the iPod. There are alternative programs out there, some free, some not, that purport to do a better job than iTunes of managing your iPod. Some are arguably pretty good, notably MediaMonkey Gold (\$19.95) that replicates the power of smartlists with its autoplaylists. Most of them, however, are based around the idea of providing the “simpler” management of drag and drop from the operating system’s GUI (*e.g.* Windows Explorer). To me, there is nothing efficient or desirable about drag and drop management, particularly if you have a large music collection and a smaller iPod. Routinely clicking through upwards of thousands of folders and tens of thousands of files is not my idea of “better than iTunes”.

This document is for those who are trying to get the most out of iTunes, not those trying to get away from iTunes. If your hang up is that you have to use iTunes to accomplish anything really useful with the iPod, stop reading now. The very reason I bought an iPod and continue to buy iPods is iTunes. Yes, it sucks for CD ripping, volume normalization, and is even a bit clunky for music playback, but for music and player organization, it can’t be touched in my opinion.

There is very little iTunes forces you to do in regards to managing your music beyond allowing iTunes to create an index of it. A nice feature of the iPod/iTunes marriage is that Apple caters to three types of control schemes. The total control freak that only wants to move files on and off of the iPod manually can do that - and drag and drop is fully supported from within iTunes if that’s your thing. At the same time, iTunes fully supports the “don’t think about it, just plug it in and go” crowd. Last, iTunes caters to those in the middle, that’s where this document comes in.

The simplest explanation of what you are going to do is set your iPod to sync to a certain set of playlists, some smart, some regular, and then all you will have to do is plug the iPod in, let it update, and, voilà, music will be moved on and off of your iPod according to the rules you choose.

Disclaimer

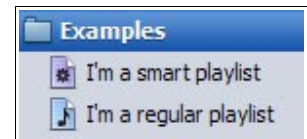
There are differences in exactly how smart playlists behave on the different iPods. Smart playlists behave slightly differently on each generation of iPods and sometimes between firmware versions for the same model. There are even some differences in current iPods between clickwheel based iPods compared to iPhone OS based iPods. I have not owned any iPhone OS based iPods and a fourth generation nano is my current main iPod (The 4G nano's software seems to be a modified version of the sixth generation iPod software). I have owned, but do not currently use, fourth and fifth generation clickwheel based iPods. It is simply not possible for me to be aware of every single difference between all the different iPods and iPod software versions. As such, although I have tried to point out known differences, you may find that not every single thing in this document is true or works as stated for your particular iPod.

Additionally, changes in the iTunes software can alter smartlist behavior and functionality. As of the publication of this document, the most current version of iTunes is 8.1. If you're using a later (or earlier) version of iTunes, you may find that not everything in this document works as stated.

Chapter 1.1 Playlist Basics

An iTunes playlist is a customized list of media files that are indexed in your iTunes library. At the simplest, it is a list of files to be played, but playlists are also important to managing the contents of your iPod. Playlists are displayed under the “Playlist” heading in the left hand source panel of iTunes. You can create new playlists three different ways: under the **File** menu there are options for creating regular or smart playlists, these menu items tell the keyboard shortcuts for your operating system, and at the bottom left of iTunes is a button with a '+' icon that says, “Create Playlist”, when you mouse over it (if you hold down the Shift key, the icon changes to a '⚙' and the mouse over says, “Create Smart Playlist”).

As alluded to, there are two types of playlists, regular and smart. A regular playlist is just a list of songs that you manage directly by dragging or copying genres, artists, albums, or songs over to the playlist. These are the lists with the sheet and musical note icons in the left hand panel of iTunes. In list view, you can change the viewable columns, sort by column, or manually select and drag items into a custom order. You can delete any contents by selecting them in the playlist and hitting the Delete key.



A smart playlist (smartlist for short) is a rules-based list whose contents is dynamically generated by iTunes and your iPod based upon rules that you set when the list is created or later edited (right click on a smartlist to edit it). These are the lists with the blue sheet and asterisk icon in the left hand panel of iTunes. Although you can sort smartlists by any available column criteria, you do not have the option to rearrange items into custom order.

There are two uses for regular playlists in managing your iPod. The first is as a list of songs you directly manage that you want on your iPod. This can be a special mix playlist, *e.g.* “Jane’s Birthday Party”, or it can be an album that you want intact on your iPod. You can use as many or as few non-smart playlists as you want for this aspect of iPod management. The second use is as a list of files that a smartlist will pull from (more on this later).

Smart playlists, however, are where it’s at – with a push of the '+' button you add a new rule. With the many drop down options you can filter for almost anything that iTunes keeps track of: play count, rating, artist, genre, composer, date added, when it was last played, bit rate, etc. You create the conditions and iTunes instantly finds every single song in your library that matches them. Further, by chaining smart playlists, you can perform some fairly sophisticated selection of music.

Both kinds of playlists allow you to display their contents in iTunes any way you can browse library content in iTunes, be it some form of list, grid, or cover flow. List view is the most important for purposes of using playlists to play and organize media on the iPod. This is because the sort order in list view determines the sort order of that playlist's content on the iPod. If you go to all the trouble to create a mix playlist, but have it sorted by artist at the time of syncing, oops, it’s all in artist order on your iPod. You have to take care to make sure that playlists are sorted or arranged exactly how you want them to be before any syncing operations. For example, in the case of an auto-generated mix, you should have it sorted according to the list track number (the next to left most column between the album art and name columns).

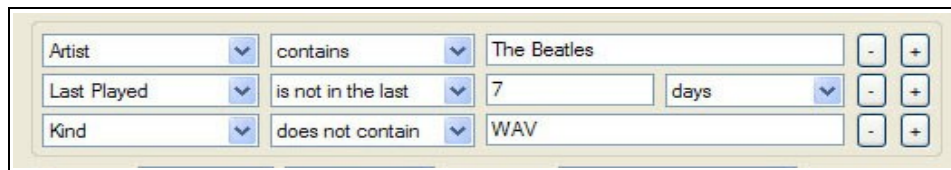
Chapter 1.2 Anatomy of a Smartlist

This is an example of what a smartlist looks like when you create or later edit it:



Starting at the top is the global “Match” criteria. This is a Boolean logic concept. Boolean logic is the rules that define how a set of logical statements linked by conditional statements (and, or, equal, not equal, etc.) are evaluated. Fortunately, the smart playlist system is simplistic: Every smart playlist has a setting, “MATCH ALL” or “MATCH ANY” at the top of it. “All” means to think of each rule in that smartlist as linked by a Boolean AND conditional and “Any” means to think of each rule in that smartlist as linked by a Boolean OR conditional. The simplest explanation is that if you have a smart playlist set to “MATCH ALL” then every rule you define must be true for every audio track or it won’t be pulled in by that list. If you have a smartlist set to “MATCH ANY” then, logically enough, if any one of the rules is true for any track, it will be pulled in by the smartlist. You can read more on Boolean logic here: http://en.wikipedia.org/wiki/Boolean_logic

Note: The check box to the left of “Match” in the above picture is a newly introduced bug (present as of iTunes version 8.1). If you uncheck it, you cannot edit the smartlist rules and conditions, nor can you select “OK” to save the smartlist. Leave it alone.



Below the “Match” criteria, is a series of line items enclosed in a light gray border. These are the rules of the smartlist. You add a new rule by clicking the '+' button at the end of the line (the new rule will be inserted below the line you clicked), and you remove a particular rule by clicking its '-' button. The left most drop down list allows you to select the category you want to filter; once you select a category, one or more category specific drop down lists or boxes become available. I'm not going to spend much time explaining these in detail (see Appendix A if you want more information). Suffice it to say that just about any id3 tag field or iTunes specific metadata criteria can be filtered via these rules. Most categories are self explanatory and a little experimentation on your part will teach you all you need to know.

Tip: If you shift-click the '+' button (option-click on a Mac) in the editing window for a smartlist, the rule category of the line you clicked, as opposed to the default of “Artist”, will automatically be selected for the new rule.

Concept: Although each individual smartlist is either a series of all AND or all OR statements due to the MATCH ALL or MATCH ANY setting global to each smartlist (with each rule line being one statement), you can also think of each playlist as a set of Boolean statements enclosed by parentheses. As such, there is no practical limit to the complexity of filtering possible through linking playlists, just limitations from what criteria and conditionals you can use in the first place (and the limitations of your patience).



Below the rules are three selection criteria check boxes. The “Limit To” is going to be the most important in general. If you leave it unchecked, the smartlist will find every matching track in your library with no upper limit. If checked, this allows you to specify the upper limit for how much music a given smartlist is supposed to find. You can limit it by number of items, total duration, or total file size. The right most drop down for the “Limit to...” function allows you to determine how matching items are pulled into the smartlist, be it at random, in album order, in name order, according to date added, the least often played of the matching results, etc.. Like suggested with the rules above, go take a look at a smartlist in iTunes and experiment to gain a better understanding of your choices (more information in Appendix B).

“Match only checked items” is self-explanatory. If this is checked then only items that have the check box (found next to the track name in the iTunes library) checked will be eligible for selection by this smartlist. If it's unchecked, any item in the library is eligible for selection.

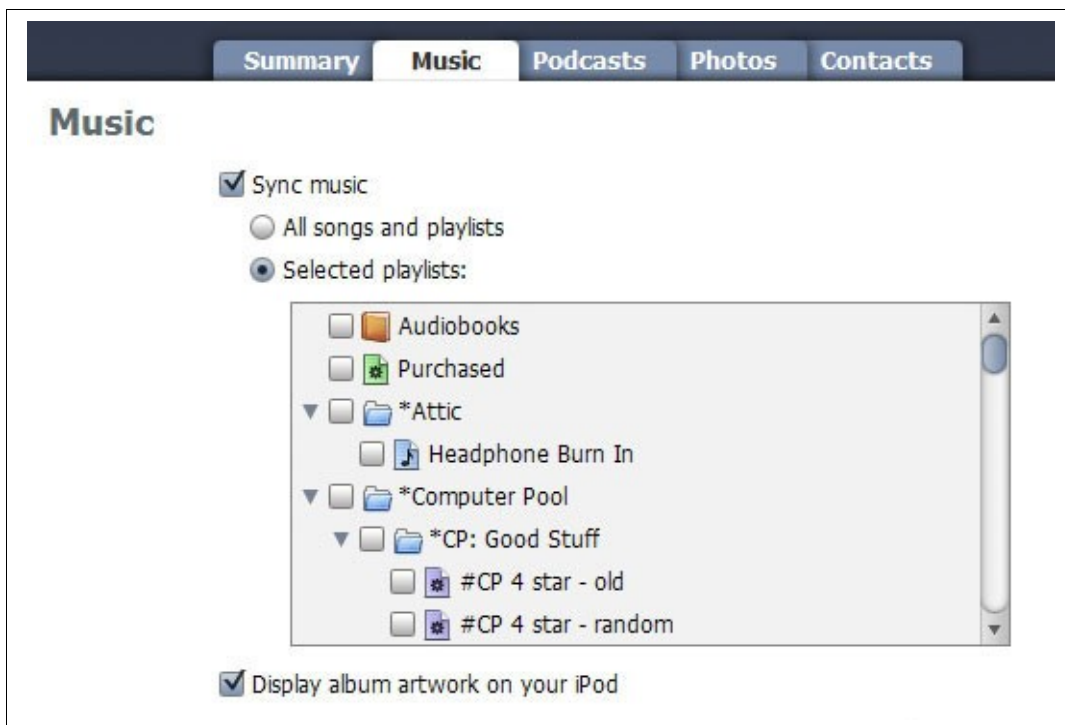
I will go into more detail about “Live updating” in a bit. For now, the basic concept is that if you have this checked then the smartlist will continue add or delete tracks from its results as filtered criteria change. For example, if you have a smartlist that pulls in all of your music that is rated higher than 3 stars and you give a new song a 4 star rating then it will be added to the smartlist. If you later change your mind and rate it 3 stars, it will be removed from the smartlist. If you uncheck this, the smartlist will only fill itself once and then remain static. You can refresh a live updating list by selecting all of the playlist items and deleting them, the list will immediately refill.

Tip: Each smartlist must contain at least one rule, but you are not required to have a rule that actually filters much. For example, you could have a single rule {Bit Rate IS GREATER THAN 1} that matches everything in your library. This can be useful when you need a smartlist to filter according to one or more of these three selection criteria rather than any id3 tag or metadata criteria.

Chapter 1.3: The basics of managing an iPod using the smartlist functions of iTunes

Configuring the iPod

In order to manage an iPod with playlists you will need to set the options on the Music tab for the iPod to sync to selected playlists. This is considered a form of auto-syncing. Among the benefits of auto-syncing is that ratings and play data, an integral part of smartlist management, are transferred bidirectionally between the iPod and iTunes. What you do in iTunes is automatically reflected on the iPod and vice versa. Choosing to sync specific playlists allows you precise control over the contents of your iPod with minimal effort on your part since, whenever the iPod is synced to iTunes, content controlled by your smartlists will automatically move on and off the iPod according to the smartlist rules.



You then simply check off all playlists and/or playlist folders you want synced to your iPod in the playlist browser. Then select the “Apply” button in the lower right of iTunes to save these changes to the iPod.

Playlist folders – Smoke'em if you got'em

In iTunes, there is the option of organizing playlists into folders. Once you create more than a few playlists, the benefits of being able to do this should be obvious. Starting with the fall 2007 models, iPods began supporting playlist folders as well. This is one of the best improvements to the iPod user interface ever. Instead of a single flat list of dozens of playlists, you can now be as organized as you wish. You can have one folder of just mixed playlists, another folder for full albums, another folder for audiobooks, a folder that contains (“hides”) all of the messy organizational playlists this document concerns, etc..

Additionally, on those iPods supporting playlist folder, nesting of folders is supported. This allows even more control over content navigation via playlists, but also has one quirk you need to keep in mind: You have to keep whatever you want to be your iPod's top level folders and playlists at the top level in the iTunes source pane because the entire folder nest hierarchy is displayed on the iPod. For example, lets say you have a bunch of nested playlist folders and playlists in iTunes but there is only one playlist that you want synced to your iPod and it's five levels deep. You check just that playlist in the sync to playlists browser, but on the iPod you will still have to click through folders one, two, three, four, and five to get to your playlist. If you want that playlist to be visible immediately when you select the Playlist menu, you'll have to move it to the top level of the source pane.

If you do not have an iPod model that supports playlist folders, it is in your best interest to adopt some sort of naming convention for playlists to make it easier to navigate. Since the iPod displays playlists sorted alphabetically, you can achieve folder like organization through prefixes (e.g. **MIX** for mix lists and **AB** for audiobooks – you will find extensive use of these prefixes in this document) and special characters (e.g. #, \$, or *) at the beginning of playlist names for sort order control. With a bit of trial and error, you can come up with a naming convention that will let you keep things organized on your iPod no matter how many playlists you have.

Tip: The iPod does not necessarily sort (or even display) special characters the same way as iTunes, hence the bit about “trial and error” above. Keep in mind that special characters also work just fine for altering sort order of playlist folders.

Smartlist Updating – Quirks and unresolved issues

When you create a smartlist, one of the options you can set is whether or not to enable “Live updating”. A smartlist for which this option is turned off only fills itself up once and then behaves like a static regular playlist until you activate the live updating check box again (at which point, you can uncheck it again to “freeze” the song selection). Live updating, as mentioned earlier, makes the list automated and dynamic. A smartlist with live updating enabled *should* automatically add or drop tracks from that smartlist whenever something it keeps track of changes. As another simple example, if you have a smartlist for all songs that haven't been played in the last week and you add a new album to your library, its tracks should be added to that list. When you listen to that album later, its tracks should be removed from that list and won't be re-added unless a week passes without them being played again. Live updating is pretty simple conceptually, but not always so simple when it comes to iTunes or the iPod. Here are some of the major quirks involving live updating that you need to know to predict how your smartlists will work:

1. In iTunes, updating of smartlists is immediate. If you're listening to a smartlist that only includes those songs rated higher than 3 stars and you lower a song's rating to 3 stars it will immediately stop playing and be removed from the list. You'll then have to restart the list playing as well.

On the iPod, however, while the smartlist is modified in the background, an actively playing list won't change so long as you are in that playlist. This, to me, is the preferable behavior because you can change the rating on a song without everything immediately coming to a halt because the state of your smartlists has been changed. However, even that's not always obvious or clear. On older iPods, the playlist contents would remain static until you went to a level above them and re-entered that playlist. On newer iPods, though, going from the now playing screen to the playlist contents does display the updated list even though it will continue play the original playlist as though it has not been changed.

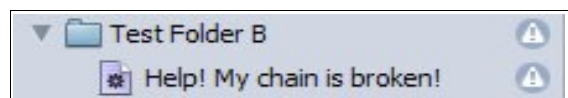
2. A smartlist will not update on the iPod unless all dependent playlists are present. Smartlists can and often do reference other playlists (or even playlist folders, see more below). However, on the iPod, for live updating to work for a particular smartlist there can be no broken links in the chain. For example, if you have a smartlist, **SMARTLIST**, that takes at least some of its content from a second smartlist, **SECOND SMARTLIST**, but only **SMARTLIST** is synced to your iPod, it won't live update on the iPod. This example is simple and obvious, but it becomes troublesome with the sort of chained systems this document is built around because if any smartlist in a chain isn't synced to the iPod, none of the smartlists involved in that chain will live update. Basically, there are three options:

A. Sync ALL your playlists involved in your management system to your iPod. This can be a bit unwieldy and has the secondary requirement that the total amount of music defined by the smartlists doesn't exceed the capacity of your iPod. Depending on what you're managing in iTunes and the particular smartlist system you are using, this may be impossible. For example, if you're using an exclusion list as a key part of your system, it may be nearly as large or larger than your iPod, and that's the stuff you don't want on the iPod! There are also some potential issues related to the processor capabilities of the iPod with attempting this that I'll cover in more detail later.

B. "Cheat" by manually copying the contents of some intermediate smartlist(s) onto regular list(s) to get around this behavior. This is how I managed music on a 60GB fifth generation iPod. I used a system of smartlists in iTunes to generate a customized music pool and periodically copied that onto a regular music pool playlist that was synced to the iPod. In turn, the on-iPod system of smartlists for creating various top mix lists all referenced this regular music pool playlist at the base of their chains. While not fully automated, such a strategy only involves the clearing and copying of contents between two lists periodically to achieve great results.

C. Just ignore live updating for on-the-go use of the iPod and rely on iTunes syncs for updating the smartlists on your iPod. If you design your system with this limitation in mind, it will be of little, if any, handicap. The first example later in this document for a playlist system uses this approach.

Note: if you have a broken smartlist chain, iTunes shows this with an exclamation point icon inside a triangle when you expand the iPod's contents in the source pane.



3. Not all of the things tracked by iTunes are supported on all iPods. Just because smartlists work with some particular criteria in iTunes is no guarantee they will function as expected on all iPod models. If you attempt to use a smartlist on your iPod that incorporates a rule involving one of these criteria, live updating for that list (as well as any lists that depend on it) may not work. Worse, iTunes won't inform you with the exclamation point icon like it does for most normal smartlist breaking errors. Some examples of known smartlist breaking elements are:

A. Referencing a folder. You can organize playlists using folders in the source pane and, even more conveniently, you can actually refer to an entire folder (even one containing nested folders) as if it were a giant playlist when using the "Playlist IS/IS NOT" rule. Unfortunately, unless you have a recent iPod that supports playlist folders, referencing a playlist folder with the playlist rule will break live updating. The solution is to not reference folders as playlists.

B. Podcast is True/False. There is a smartlist rule just for whether an item is a podcast. Newer iPod models support this, but on older ones it doesn't work right and any playlists that use this rule will not live update on the iPod. The solution is instead to use the {Genre IS/IS NOT Podcast} rule to include or exclude podcasts. With rare exceptions, all podcasts downloaded by iTunes will have this as their genre automatically anyhow.

C. Video Kind is/is not XXX. Like the Podcast flag, this is another potentially half-baked implementation that does not work right on older iPods. You'll want to use the general {Kind IS/IS NOT XXX} or {Genre IS/IS NOT XXX} rules for including or excluding video content on the iPod with smartlists to avoid this problem.

D. Last Skipped is/is not in the last.... A few years ago, iTunes began tracking if a track was skipped in the first 20 seconds. This can be a handy way of getting a track to drop off a smartlist temporarily if you're not in the mood for it just now but might be tomorrow. Similar to the Podcast and Video flags, the rule is incompatible rule with older iPods. The solution here is to not to use this condition unless you know your iPod supports it.

There are possibly more conditions that can break live updating on the iPod, but these are four that I know about. If a smartlist *should* live update on the iPod, as in you know it doesn't reference any playlists not on the iPod, live updating is checked, and it still fails to live update, it's probably because of an iPod incompatible rule. You will have to engage in a little trouble shooting to figure out the source of your problem.

Still no good way to use smartlists to select albums

One thing you'd think you could use smartlists to do is select whole albums, but you'd be wrong. The closest you can come is to use the select by album criteria to generate a large list of albums, and this almost works. The catch is that no matter how carefully you craft your selection criteria, you will still almost always wind up with at least one partial album, and there's simply no way at all to grab a single album. This is doubly puzzling as it has been a consistently requested feature from users for years and the ability is clearly already present in the functionality of most iPods ("Shuffle by Album" is an available setting on all non-iPhone OS iPods going back to monochromatic iPods).

Smartlist Updating - Resolved Issues

It used to be that there was an illogical processing of smartlists when you synced an iPod. Files were moved on and off the iPod based on the contents of the relevant smartlists in iTunes before the sync, then the updated play data from the iPod was processed by iTunes. Particularly for smartlists depending on iTunes for updating them (*i.e.* deliberately broken chain smartlists), this mandated a second sync after the first one completed to ensure that smartlist contents were current. iTunes now processes the play data from the iPod first and then moves files on or off as needed, a very nice change.

The big picture

The basic concept of this document is chaining smartlists to perform more complex filtering of music than you can do with a single playlist. Conceptually, I think of this as different tiers. There are base level playlists that perform primary filtering of content and do not reference any other playlist (with the possible exception of an **EXCLUSION** playlist as explained in later sections). These base level playlists

are combined or aggregated into a second tier playlist, which may or may not be aggregated with other base or second tier playlists to form a third tier music pool. The music pool functions as a randomized yet rules based and weighted subset of your whole library. From that pre-filtered and customized selection, you can then build top level mix lists that you use as your listening lists; these are the “cyber DJs” talked about in the introductory section.

Graphically, you could represent the sort of systems I'll explain in detail later on like this:



Chapter 2.1: Time for some examples of playlist systems

What we're going to learn about

This document will describe and discuss in some detail three systems of using smartlists for organizing music on your iPod:

1. The first system does not take advantage of live updating on the iPod and relies upon syncing with iTunes for all updates of the playlist contents. Conceptually, it is the easiest to understand so we'll start there.
2. The second system is more elegant and takes advantage of live updating on the iPod for generating dynamic playlists. However, it requires an intermediate regular playlist as a source. Its advantage is that it gets around the problems of having to sync every playlist involved in a smartlist chain to the iPod.
3. The third system is, I believe, the most elegant. Everything is automated with smartlists. The disadvantage is that it's the most complicated to set up properly and may not work well on large capacity iPods.

Ratings – They're not just a waste of time

While it is not necessary to use ratings, they are my preferred way of weighting mixes so that they always contain a certain percentage of my favored songs. Ratings are specific to each song and the iPod has a handy way to set and change song ratings on the device. Still, it is possible to create analogous weighting with genres, artists, groupings, etc., it all depends on what works for you and your particular library.

At any rate, all three example systems make extensive use of ratings for weighting music mixes. The rating scheme used in this document (which happens to be my personal rating scheme) is:

- 1 star = garbage, don't care if I ever hear it again.
- 2 star = doesn't work as a shuffled song but fine if listening to the whole album.
- 3 star = meat and potatoes song; good but not great.
- 4 star = really good song but not the best.
- 5 star = my absolute favorites.

This document's conventions for describing smartlists

Each smartlist can be described by its name, MATCH setting, rules, and then any selection criteria set for it. To keep things clear, most smartlists will be shown like this:

Playlist	Conditions	Selection Criteria
SAMPLE PLAYLIST	MATCH ALL {Bit rate IS GREATER THAN 1}	50 songs selected at random

Playlist names will always be in all caps, Arial font, and blue. Individual rules will be enclosed in brackets. Conditionals (e.g. GREATER THAN or IS NOT) will also be in all caps. Live updating is always on in these systems.

Chapter 2.2: The Beginner

This system is a very simple one that could be used to either manage a small capacity iPod (I used a similar system once for an iPod mini) or to supplement whatever management scheme you have for a larger capacity iPod. The basic idea is to generate a mixed pool of music that isn't just random but, instead, is weighted toward your favorite music through the use of multiple and chained smartlists.

Concept: iTunes is not truly random. If you've used the "Shuffle Songs" or any other randomizing feature you probably already figured that out. There are certainly random elements, but there is also some undocumented weighting and/or kludges going on in the choices due to the underlying algorithms. The upside is that it's not that bad at simulating the job a human does when going through a list of music and picking and choosing; there is a certain amount of seemingly logical clumping and concentration of artists within any particular generated list. The downside is that, without your help, the shuffling and random selection features of iTunes will do little more than shuffle through the same music you've already heard while managing to ignore most of your library. Fortunately, like most Apple products, although "out of the box" it's geared more towards trained chimps, power users can use iTunes to great effect if they'll put a little thought into things.

The mixed music pool is built around the following categories:

1. Highly rated music – who doesn't want to hear their favorite music?
2. A random selection from the whole library – this gets things mixed up.
3. Music not played in a while – makes sure that everything is rotated.
4. Music you're in the mood for now.

Parts 1, 2 & 3 are the general mix, part 4 is more specialized; it's that new album that just came out, or that artist you have a hankering to immerse yourself in. I'll discuss how to generate the general part of the mix first.

First thing you need to do is define what can be shuffled and what can't. We're going to create a smart playlist called **EXCLUSION**. Set it to MATCH ANY and set conditions for anything you don't want shuffled. For example, {Genre IS Audiobook} and {Genre IS Comedy} are some of the rules I used in my own system. If you use ratings (as this example does), also set it to match any ratings you don't want shuffled (for me, that's the 1 & 2 star rated songs). The end result is a list of all the files you don't want to be shuffled by iTunes. The way you use this list is that whenever you generate another smart playlist you can make {Playlist IS NOT **EXCLUSION**} one of the mandatory requirements and it will never pull in anything you don't want. Once you've done that, you're ready to move on.

Concept: One thing mentioned above and that you need to keep in mind is that iTunes is positively awful at hitting all your music without you forcing it to do so. If you do not create the conditions that force it to play all your music then the sun might very well be cold by the time iTunes does so left on its own. I've seen some people recommend forcing full rotation by always having playlists select according to least recently played. However, that's a very artificial way of listening to your music as you only hear things once per library rotation, bleh. There are ways to force this full library rotation without resorting to pure brute force, and I'm going to tell you about them.

Concept: Before I get to that, the key thing about these content generating smartlists is that they have to cycle music on and off the playlist (and, by extension, your iPod) in the first place. When you mess

around with smartlists, you will see that there are a huge variety of temporal and quantitative methods to limit music. You can have iTunes only select those songs played less than a certain amount of times, you can have iTunes only select those songs not played in the last 7 days (or 7 weeks), you can have iTunes only select music added in the past month, and so on. These are the keys to shuffling music through these playlists. If you have a smartlist with the limitation that music must not have been played in the past week and you do play it, poof, it drops off the list and something else is pulled in to replace it once the list is updated.

Addendum: When using a non live updating system like this, if you use a “not played in the last...” time window to cycle a particular smartlist, the time window chosen to drop music from a playlist must be longer than whatever is your typical iTunes sync interval. If you set a playlist to drop out songs played in the past week but only sync every two weeks then nothing will be rotated on and off of your iPod.

The Playlists at last

Now we are ready to describe the notion of making the general mix. For this system, the goal is a pool of 500 songs. Going back to the three categories described above (highly rated, random, & least recently played), you’ll need to choose how much of each you want. We’ll use the following percentages for this example: 20% highly rated (100 songs), 45% random (225 songs), & 35% least recently played (175 songs).

Playlist GOODSTUFF (the 4 & 5 star rated songs) is put together like so (this is a good example of how to chain playlists so pay attention):

First, you need to decide on a split of 5 versus 4 stars; in this example, we’ll go with a 60:40 split. Second, you need to decide on the minimum interval between hearing songs; in this example, we’ll use an interval of 7 days for 5 star songs and 14 days for 4 star songs.

Now, here’s where forcing full library rotation comes in: if you just stuck with selecting according to {Last Played IS NOT IN THE LAST X days}, you’d wind up hearing a lot of the same songs repeatedly over time due to the way iTunes favors certain tracks. Here’s a basic way of getting around that behavior, create 4 playlists: **4 STAR OLD**, **4 STAR RANDOM**, **5 STAR OLD**, & **5 STAR RANDOM**. By selecting some of the highly rated stuff at random and some by least recently played this forces iTunes to shuffle through all your high rated music while still being more or less random to all appearances.

Playlist	Conditions	Selection Limitation
4 STAR OLD	Match ALL {Rating IS 4 stars} {Playlist IS NOT EXCLUSION }	20 songs selected according to least recently played
4 STAR RANDOM	Match ALL {Rating IS 4 stars} {Last Played IS NOT in last 14 days} {Playlist IS NOT 4 STAR OLD } {Playlist IS NOT EXCLUSION }	20 songs selected at random

Playlist	Conditions	Selection Limitation
5 STAR OLD	Match ALL {Rating IS 5 stars} {Playlist IS NOT EXCLUSION}	30 songs selected according to least recently played
5 STAR RANDOM	Match ALL {Rating IS 5 stars} {Last Played IS NOT in last 7 days} {Playlist IS NOT 5 STAR OLD} {Playlist IS NOT EXCLUSION}	30 songs selected at random

From those four sublists you put together your 100 songs of **GOODSTUFF**:

Playlist	Conditions	Selection Limitation
GOODSTUFF	Match ANY {Playlist IS 4 STAR OLD} {Playlist IS 4 STAR RANDOM} {Playlist IS 5 STAR OLD} {Playlist IS 5 STAR RANDOM}	100 songs selected at random

Concept: The reason for limiting the 100 songs of the 4 sublists to 100 songs at random may not be immediately obvious: iTunes does not have any way to randomize a smart playlist unless its selection is limited. So, the list is limited so that the songs from the 4 lists will be further mixed up, otherwise they'd just be put in the same order they appear in the 4 sub-lists and we want things as random as possible to try and foil the iTunes patterning in "random" selections.

Next step is to create the random playlist, this one is much simpler:

Playlist	Conditions	Selection Limitation
RANDOM	Match ALL {Playlist IS NOT GOODSTUFF} {Playlist IS NOT EXCLUSION} {Last Played IS NOT IN THE LAST 21 days}	225 songs selected at random

Concept: By telling **RANDOM** that it can't have anything that appears on **GOODSTUFF**, duplicates are avoided and, therefore, all songs are unique. You will notice this was also used in the first lists such as **5 STAR OLD** & **5 STAR RANDOM**. This theme will be repeated throughout these playlists.

Next, create the playlist for the oldest played music:

Playlist	Conditions	Selection Limitation
OLD STUFF	Match ALL {Playlist IS NOT GOODSTUFF} {Playlist IS NOT RANDOM} {Playlist IS NOT EXCLUSION}	175 songs selected according to least recently played

Then put it all together:

Playlist	Conditions	Selection Limitation
GOOD, OLD, & RANDOM	Match ANY {Playlist IS GOODSTUFF} {Playlist IS OLD STUFF} {Playlist IS RANDOM}	500 songs selected at random

Now, that is indeed a 500 song, auto-updating pool, but what we’re going to do in this case is dope it with the 4th category of music, the music you’re in the mood for now. This step is purely optional, but it’s an example of one way you can get a focus on music you want to hear sooner than later.

Create a regular playlist, we’ll call it HERBS. Onto HERBS drag any music that you want to hear more often than a random cycling through the library will accomplish. There’s no limit to what you can put on this list as it is only a source for a smart playlist that will be limited.

After you have your HERBS, create a smart playlist, we’ll call it SPICES:

Playlist	Conditions	Selection Limitation
SPICES	Match ALL {Playlist IS HERBS} {Last Played IS NOT IN THE LAST 14 days}	100 songs selected at random

I could make things more complex, but HERBS is a list that is managed manually and constantly changing so there’s no need to make it any more complicated than this. There is no need to involve EXCLUSION because nothing goes on HERBS that you don’t intentionally put there.

If you’ve been paying attention you should almost be able to predict what comes next: you “cut” the 500 song pool you made, GOOD, OLD, & RANDOM, with some random number of these 100 SPICES:

Playlist	Conditions	Selection Limitation
MIXPOOL	Match ANY {Playlist IS GOOD, OLD, & RANDOM} {Playlist IS SPICES}	500 songs selected at random

The exact final mix is further randomized when you mix two playlists in this manner. You can wind up with anywhere from 0 to all 100 of the tracks from [SPICES](#) in [MIXPOOL](#).

There you go, that's how you generate a pool of randomized songs that will actually sound like something you would have picked on your own if you had the time to go through everything in your library for hours out of the day and put together a 500 song playlist and constantly update it. At this point, you can slap [MIXPOOL](#) on your iPod and play it, or even shuffle it if you're so inclined.

Unfortunately, without taking advantage of live updating on the iPod, there's only so much you can do. You could break the songs of into smaller lists if you wanted to make it easier to keep track of where you were at. You can use [MIXPOOL](#) as a source for more refined mix lists (more on this in later examples). You could make second [MIXPOOL](#) like mix list with different weighting in the choices. Ultimately, though, the real power comes from exploiting on the go live updating. Bring on the next chapter...

Chapter 2.3: The Intermediate

This next system is a more advanced version of the first one. I originally devised a version of this system for music management on a 5G 60GB iPod. In this system you're going to take advantage of on the go live updating, probably the iPod's greatest feature as no other player comes close to matching this ability. Using on the go live updating, we will create multiple uniquely filtered playlists that refresh and refill themselves automatically from a weighted mix list similar to **MIXPOOL** from the first example.

This example system is based upon a music pool of "only" 1000 songs. This system demonstrates how you would create an auto-updating set of lists on the iPod. In the end, it's just a tweaked and advanced version of the first example, so understand this and you'll be able to do just about anything with smartlists.

The basis of this system is, again, to build your **EXCLUSION** playlist first since almost everything else stems from it. For example, if I were still using this system with my library, **EXCLUSION** would look like this:

Playlist	Conditions	Selection Limitation
EXCLUSION	Match ANY {Kind CONTAINS Video} {Kind CONTAINS WAV} {Genre CONTAINS Audiobook} {Genre CONTAINS Comedy} {Genre CONTAINS Holiday} {Genre CONTAINS Podcast} {Rating IS 1 star} {Rating IS 2 stars}	none

In the previous example, we broke categories down into random and least-recently-played sub-categories. For this system, we'll introduce a third sub-category: least-often-played. By adding this category there will be a bit of nudge for more recently added music until they catch up on play count with older songs. It's a bit of a trade off, by adding in this category, your weighting for ratings, genre, etc. will be watered down somewhat, but, in turn, you will gain a more even exposure to your music library.

As a result, there's now a new distribution of song types:

- 20% 5 star songs*
- 20% 4 star songs*
- 20% random songs
- 20% least recently played
- 20% least often played

** these are also broken out into random, least recently played, and least often played*

So, the most basic lists are as follows:

Playlist	Conditions	Selection Limitation
4 STAR - LEAST	Match ALL {Playlist IS NOT EXCLUSION } {Rating IS 4 stars} {Last played IS NOT IN THE LAST 9 days}	67 songs by least often played
4 STAR - OLD	Match ALL {Playlist IS NOT EXCLUSION } {Rating IS 4 stars} {Last played IS NOT IN THE LAST 9 days} {Playlist IS NOT 4 STAR – LEAST }	67 songs by least recently played
4 STAR - RANDOM	Match ALL {Playlist IS NOT EXCLUSION } {Rating IS 4 stars} {Last played IS NOT IN THE LAST 9 days} {Playlist IS NOT 4 STAR – LEAST } {Playlist IS NOT 4 STAR – OLD }	67 songs by random
5 STAR - LEAST	Match ALL {Playlist IS NOT EXCLUSION } {Rating IS 5 stars} {Last played IS NOT IN THE LAST 6 days}	67 songs by least often played
5 STAR - OLD	Match ALL {Playlist IS NOT EXCLUSION } {Rating IS 5 stars} {Last played IS NOT IN THE LAST 6 days} {Playlist IS NOT 5 STAR – LEAST }	67 songs by least recently played
5 STAR - RANDOM	Match ALL {Playlist IS NOT EXCLUSION } {Rating IS 5 stars} {Last played IS NOT IN THE LAST 6 days} {Playlist IS NOT 5 STAR – LEAST } {Playlist IS NOT 5 STAR – OLD }	67 songs by random

Playlist	Conditions	Selection Limitation
LEAST	Match ALL {Playlist IS NOT EXCLUSION } {Last played IS NOT IN THE LAST 12 days} {Playlist IS NOT 4 STAR – LEAST } {Playlist IS NOT 4 STAR – OLD } {Playlist IS NOT 4 STAR – RANDOM } {Playlist IS NOT 5 STAR – LEAST } {Playlist IS NOT 5 STAR – OLD } {Playlist IS NOT 5 STAR – RANDOM }	200 songs by least often played
OLD	Match ALL {Playlist IS NOT EXCLUSION } {Last played IS NOT IN THE LAST 12 days} {Playlist IS NOT 4 STAR – LEAST } {Playlist IS NOT 4 STAR – OLD } {Playlist IS NOT 4 STAR – RANDOM } {Playlist IS NOT 5 STAR – LEAST } {Playlist IS NOT 5 STAR – OLD } {Playlist IS NOT 5 STAR – RANDOM } {Playlist IS NOT LEAST }	200 songs by least recently played
RANDOM	Match ALL {Playlist IS NOT EXCLUSION } {Last played IS NOT IN THE LAST 12 days} {Playlist IS NOT 4 STAR – LEAST } {Playlist IS NOT 4 STAR – OLD } {Playlist IS NOT 4 STAR – RANDOM } {Playlist IS NOT 5 STAR – LEAST } {Playlist IS NOT 5 STAR – OLD } {Playlist IS NOT 5 STAR – RANDOM } {Playlist IS NOT LEAST } {Playlist IS NOT OLD }	200 songs by random

These base lists are then combined into a pooled list, **BASE POOL**:

Playlist	Conditions	Selection Limitation
BASE POOL	Match ANY {Playlist IS 4 STAR – LEAST } {Playlist IS 4 STAR – OLD } {Playlist IS 4 STAR – RANDOM } {Playlist IS 5 STAR – LEAST } {Playlist IS 5 STAR – OLD } {Playlist IS 5 STAR – RANDOM } {Playlist IS LEAST } {Playlist IS OLD } {Playlist IS RANDOM }	1000 songs by random

A change you might notice from the first example is that these basic lists are no longer being pulled into intermediate lists such as **GOODSTUFF**. The reasoning behind the old method was to increase randomization of track order. Because tracks from **BASE POOL** will undergo at least three more randomized selections before reaching your ears, we’re opting for a cleaner approach.

This example also abandons the manually managed **HERBS & SPICES** for “cutting” the main mix. In this example, we will instead place the emphasis on new and unrated music automatically:

Playlist	Conditions	Selection Limitation
SPICES & HERBS	Match ALL {Playlist IS NOT EXCLUSION } {Date Added IS IN THE LAST 60 days} {Last played IS NOT IN THE LAST 12 days}	250 songs by random
UNRATED	Match ALL {Playlist IS NOT EXCLUSION } {Rating IS 0 stars} {Last played IS NOT IN THE LAST 12 days}	250 songs by random

*Note: There is no attempt to avoid duplication with either anything from **BASE POOL** or these two lists because the goal is to favor these two groups of songs.*

Then the three lists are mixed together to get to the intermediate list:

Playlist	Conditions	Selection Limitation
MUSIC POOL[OFF]	Match ANY {Playlist IS BASE POOL } {Playlist IS SPICES & HERBS } {Playlist IS UNRATED }	1000 songs by random

Now, here is where the part about needing all dependent playlists on the iPod to achieve live updating comes in. If you simply put **MUSIC POOL[OFF]** onto the iPod, any higher level smartlists that reference it won't live update, so what you do is instead copy the contents of **MUSIC POOL[OFF]** to a regular playlist: **MUSIC POOL[ON]**, and that's the music pool playlist that we'll sync to the iPod.

To refresh your iPod's contents, you clear the contents of **MUSIC POOL[ON]** and fill it with the **MUSIC POOL[OFF]**'s current contents at some interval based upon your listening habits. For example, if you listen to several hours of music a day, you may find it better to do this every few days. If you only listen to an hour or so of music each day, you might only need to refresh **MUSIC POOL[ON]** every couple of weeks.

Concept: The idea here is that **MUSIC POOL[ON]** is the entire library as far as smart lists on the iPod are concerned. Every single smartlist will either point to **MUSIC POOL[ON]** or a playlist "upstream" of it rather than the library and/or iPod contents in general. This is the only way to have a complex smartlist system like this and avoid both syncing all involved playlists (that may or may not fit on your iPod anyhow) and the potential problems of performance on large capacity iPods (covered in chapter 2.4).

So far, so good, but now we have a new problem. The reason you need to copy your music pool to a regular playlist is so you don't have to sync all dependent lists (e.g. **EXCLUSION**) to the iPod. You still need somehow to ensure that once a song is played and/or rated unfavorably that it won't play again. You could essentially recreate the playlists you used to create **MUSIC POOL[ON]**, only this time pointing to **MUSIC POOL[ON]**, but that is overkill. The simplest solution is to make a smartlist to filter newly excluded content from **MUSIC POOL[ON]**:

Playlist	Conditions	Selection Limitation
DYNAMIC POOL	Match ALL {Playlist IS MUSIC POOL[ON] } {Rating IS NOT 1 star} {Rating IS NOT 2 stars} {Last played IS NOT IN THE LAST 6 days}*	1000 songs by random

* The Last played rule is the most flexible here and depends upon your average sync interval and types of mix lists built from **DYNAMIC POOL**. For this example, 6 days was picked assuming a sync interval of only 3 to 5 days. The disadvantage of this simple approach is that rotation for all your songs is fixed. If this fixed period or your sync interval exceeds whatever is normally the shortest rotation interval, the fine tuning of rotation from variable rotation interval periods will be lost.

A more advanced way to build **DYNAMIC POOL** is to filter according to last played for the three rotation intervals you have defined (5 star = 6 days, 4 star = 9 days, everything else = 12 days). The three intermediate playlists are:

Playlist	Conditions	Selection Limitation
5 STAR[ON]	Match ALL {Playlist is MUSIC POOL[ON] } {Rating IS 5 stars} Last played IS NOT IN THE LAST 6 days}	1000 songs by random
4 STAR[ON]	Match ALL {Playlist is MUSIC POOL[ON] } {Rating IS 4 stars} Last played IS NOT IN THE LAST 9 days}	1000 songs by random
THE REST[ON]	Match ALL {Playlist is MUSIC POOL[ON] } {Rating IS NOT 5 stars} {Rating IS NOT 4 stars} {Rating IS NOT 2 stars} {Rating IS NOT 1 star} Last played IS NOT IN THE LAST 12 days}	1000 songs by random

If you choose this advanced method, **DYNAMIC POOL** is:

Playlist	Conditions	Selection Limitation
DYNAMIC POOL	Match ANY {Playlist is 5 STAR[ON] } {Playlist is 4 STAR[ON] } {Playlist is THE REST[ON] }	1000 songs by random

Unless you go a very long time between syncs with your iTunes library, this advanced **DYNAMIC POOL** will replicate the original behavior of **MUSIC POOL[OFF]** almost identically.

Whichever method you use to make **DYNAMIC POOL**, the final step is to create top level auto-updating playlists for daily use. These are some examples of playlists for specific situations that I have used (you, of course, should create top level playlists filtered according to your needs):

Playlist	Conditions	Selection Limitation
MIX: ALL OF IT	Match ALL {Playlist IS DYNAMIC POOL }	25 songs by random

This generates a general list, it includes everything.

Playlist	Conditions	Selection Limitation
MIX: EXPLORATION	Match ALL {Playlist IS DYNAMIC POOL} {Rating IS 0 stars}	20 songs by random

This generates a list for listening to unrated songs.

Playlist	Conditions	Selection Limitation
MIX: METAL MINUTE	Match ALL {Playlist IS DYNAMIC POOL} {Genre CONTAINS Metal}	30 minutes at random

This generate a list of one half hour of music from the metal genres

Playlist	Conditions	Selection Limitation
MIX: POKER NIGHT	Match ALL {Playlist IS DYNAMIC POOL} {Play count IS GREATER THAN 0} {Rating IS GREATER THAN 2 stars} {Genre DOES NOT CONTAIN <a copy of this rule for everything my buddies hate>} {Artist DOES NOT CONTAIN <a copy of this rule for everything my buddies hate>}	4 hours worth of music by random

This generates a list for playing poker with my friends. It avoids stuff like classical, movie soundtracks, new age, world, and J-Pop. Additionally, because it can only pull in songs that I've listened to at least once and rated 3 or higher stars, it avoids any untested songs that might not be very good, at least to my friends' ears.

Playlist	Conditions	Selection Limitation
MIX: SHORT ATTENTION SPAN	Match ALL {Playlist IS DYNAMIC POOL} {Time IS LESS THAN 6:01}	20 songs by random

This is a list that keeps all songs at six minutes or less in length, its name says it all.

Playlist	Conditions	Selection Limitation
MIX: SOME OF IT	Match ALL {Playlist IS DYNAMIC POOL } {Genre DOES NOT CONTAIN <a copy of this rule for everything my wife hates or my kids shouldn't hear>} {Artist DOES NOT CONTAIN <a copy of this rule for everything my wife hates or my kids shouldn't hear>} {Name DOES NOT CONTAIN <a copy of this rule for examples of adult language in the titles or specific songs that are objectionable, e.g. the original uncensored version of The End by The Doors>}	20 songs by random

This generates a list for when I'm with my wife and/or kids. It avoids stuff like heavy metal, hard rap, punk, and songs involving excessive profanity, violence, or sexual content.

Then all you have to do is set the iPod to sync to the following playlists :

MUSIC POOL[ON]
5 STAR[ON] *
4 STAR[ON] *
THE REST[ON] *
DYNAMIC POOL
MIX: ALL OF IT
MIX: EXPLORATION
MIX: METAL MINUTE
MIX: POKER NIGHT
MIX: SHORT ATTENTION SPAN
MIX: SOME OF IT

** if you used the simpler **DYNAMIC POOL** you wouldn't have these playlists*

That's it, this gives you self updating and customized playlists on your iPod that almost automatically pull from your entire library. Because the **MIX** lists are self updating, whenever you reach the end of a given playlist, merely go back into it from the Playlist menu and it will be completely refreshed (newer iPods are even slicker, they just kick you to the first track of the refreshed playlist, just need to hit play). Additionally, there is no need to remember where you were in a playlist because if you go and listen to another playlist, full album, podcast, etc., and come back, it will automatically have dropped all of the previously listened-to tracks.

Tip: When creating your own customized top level playlists, do not be afraid of adding rules. I have given very simple examples to make the concepts clear, but you can have dozens of rules for a single playlist and probably not affect the iPod's performance. For example, my current version of a kid safe playlist has over 50 rules.

Chapter 2.4: Your final exam

The final example of a linked smartlist system we'll go over is my favorite but, depending on your iPod and what you're trying to manage, may be more of a proof of concept than anything practical. This system is completely automated; there is no copying of intermediate lists, there is no problem of too large **EXCLUSION** lists, it just works once you set it up (with some caveats).

If you think about it long enough, you might realize that in any list that you were able to use the {Playlist IS NOT **EXCLUSION**} rule for eliminating unwanted tracks, you could have accomplished the exact same thing by instead putting NOT versions for each condition that is set as TRUE in your **EXCLUSION** list.

For example, the playlist, **RANDOM**, shown in the first playlist system might look something like this:

Playlist	Conditions	Selection Limitation
RANDOM	Match ALL {Playlist IS NOT GOODSTUFF } {Last Played IS NOT IN THE LAST 21 days} {Kind DOES NOT CONTAIN Video} {Kind DOES NOT CONTAIN WAV} {Genre IS NOT Audiobook} {Genre IS NOT Comedy} {Genre IS NOT Holiday} {Genre IS NOT Podcast} {Rating IS NOT 1 star} {Rating IS NOT 2 stars}	225 songs selected at random

And once you've realized that, it's only a little bigger jump in thought to realize that no matter how big your iTunes library, here's a way to put the whole smartlist management system on any iPod no matter what its capacity and have everything 100% automated. You'll never need to manually copy a thing or wait for an iTunes sync to update playlists since you can have all the playlists on the iPod. Sure, that's a lot of rules to set for each smartlist, but isn't the full automation going to be worth it?

Maybe, but maybe not - your iPod is not necessarily much of a computer. Once you move to a fully automated system, the entire contents of your iPod are fair game. Every time something tracked by any smartlist changes, the state of all related smartlists must be updated. This can result in ten or even twenty thousand item long lists being sorted and resorted again and again in the meager memory space of your iPod. Trying to run a multiply linked playlist system like this may not be so glorious as you first imagine. I know because I tried this with a fully loaded 60GB 5G iPod and wound up with 90 second or more delays when moving between smartlists because of the amount of computations that had to be performed (not to mention the intense amount of hard drive activity while it did all of this). On the other hand, while it may not work for managing a fully loaded full sized iPod, it definitely does work very well with smaller capacity iPods such as nanos or touches with a few thousand songs at the most. It's also even possible that the 2007 or 2008 classic might have enough processor power to manage it (but I doubt it - if anyone has tried this, feel free to contact me and I'll update this section). I offer this playlist system as a tantalizing morsel, but use the concept wisely. If you've got a nano, the system is brilliant,

with delays of only a second or two when moving between smartlists . If you've got a fully loaded full size iPod, you've been warned.

Tip: When you have a lot of playlists, you will notice that iTunes only shows as many as will fit on one screen when selecting for the {Playlist IS/IS NOT} rule. Use the Page Up & Page Down buttons to move to playlists off screen.

For simplicity's sake, this system will be based around the same music distribution as the intermediate system (from page 18). Onto the playlists:

Playlist	Conditions	Selection Limitation
XX[A]: 4 STAR - LEAST	Match ALL {Rating IS 4 stars} {Genre IS NOT Holiday} {Last played IS NOT IN THE LAST 8 days}	67 songs by least often played
XX[A]: 4 STAR - OLD	Match ALL {Rating IS 4 stars} {Genre IS NOT Holiday} {Last played IS NOT IN THE LAST 8 days} {Playlist IS NOT 4 STAR – LEAST}	67 songs by least recently played
XX[A]: 4 STAR - RANDOM	Match ALL {Rating IS 4 stars} {Genre IS NOT Holiday} {Last played IS NOT IN THE LAST 8 days} {Playlist IS NOT 4 STAR – LEAST} {Playlist IS NOT 4 STAR – OLD}	67 songs by random
XX[A]: 5 STAR - LEAST	Match ALL {Rating IS 5 stars} {Genre IS NOT Holiday} {Last played IS NOT IN THE LAST 6 days}	67 songs by least often played
XX[A]: 5 STAR - OLD	Match ALL {Rating IS 5 stars} {Genre IS NOT Holiday} {Last played IS NOT IN THE LAST 6 days} {Playlist IS NOT 5 STAR – LEAST}	67 songs by least recently played

Playlist	Conditions	Selection Limitation
XX[A]: 5 STAR - RANDOM	Match ALL {Rating IS 5 stars} {Genre IS NOT Holiday} {Last played IS NOT IN THE LAST 6 days} {Playlist IS NOT 5 STAR – LEAST} {Playlist IS NOT 5 STAR – OLD}	67 songs by random

Since these examples are based on my own systems, and I only rate music, the 4 & 5 star related lists are relatively simple since by requiring a given rating everything except music is automatically excluded. The next playlists will be more detailed.

Playlist	Conditions	Selection Limitation
XX[A]: LEAST	Match ALL {Last played IS NOT IN THE LAST 11 days} {Playlist IS NOT XX[A]: 4 STAR – LEAST} {Playlist IS NOT XX[A]: 4 STAR – OLD} {Playlist IS NOT XX[A]: 4 STAR – RANDOM} {Playlist IS NOT XX[A]: 5 STAR – LEAST} {Playlist IS NOT XX[A]: 5 STAR – OLD} {Playlist IS NOT XX[A]: 5 STAR – RANDOM} {Kind DOES NOT CONTAIN Video} {Kind DOES NOT CONTAIN WAV} {Genre DOES NOT CONTAIN Audiobook} {Genre DOES NOT CONTAIN Comedy} {Genre DOES NOT CONTAIN Holiday} {Genre DOES NOT CONTAIN Podcast} {Rating IS NOT 1 star} {Rating IS NOT 2 stars}	200 songs by least often played

Playlist	Conditions	Selection Limitation
XX[A]: OLD	Match ALL {Last played IS NOT IN THE LAST 11 days} {Playlist IS NOT XX[A]: 4 STAR – LEAST} {Playlist IS NOT XX[A]: 4 STAR – OLD} {Playlist IS NOT XX[A]: 4 STAR – RANDOM} {Playlist IS NOT XX[A]: 5 STAR – LEAST} {Playlist IS NOT XX[A]: 5 STAR – OLD} {Playlist IS NOT XX[A]: 5 STAR – RANDOM} {Playlist IS NOT XX[A]: LEAST} {{Kind DOES NOT CONTAIN Video} {Kind DOES NOT CONTAIN WAV} {Genre DOES NOT CONTAIN Audiobook} {Genre DOES NOT CONTAIN Comedy} {Genre DOES NOT CONTAIN Holiday} {Genre DOES NOT CONTAIN Podcast} {Rating IS NOT 1 star} {Rating IS NOT 2 stars}	200 songs by least recently played
XX[A]: RANDOM	Match ALL {Last played IS NOT IN THE LAST 11 days} {Playlist IS NOT XX[A]: 4 STAR – LEAST} {Playlist IS NOT XX[A]: 4 STAR – OLD} {Playlist IS NOT XX[A]: 4 STAR – RANDOM} {Playlist IS NOT XX[A]: 5 STAR – LEAST} {Playlist IS NOT XX[A]: 5 STAR – OLD} {Playlist IS NOT XX[A]: 5 STAR – RANDOM} {Playlist IS NOT XX[A]: LEAST} {Playlist IS NOT XX[A]: OLD} {Kind DOES NOT CONTAIN Video} {Kind DOES NOT CONTAIN WAV} {Genre DOES NOT CONTAIN Audiobook} {Genre DOES NOT CONTAIN Comedy} {Genre DOES NOT CONTAIN Holiday} {Genre DOES NOT CONTAIN Podcast} {Rating IS NOT 1 star} {Rating IS NOT 2 stars}	200 songs by random

With these base level playlists, you build the first of the second tier playlists:

Playlist	Conditions	Selection Limitation
XX[B]: BASE POOL	Match ANY {Playlist IS XX[A]: 4 STAR – LEAST } {Playlist IS XX[A]: 4 STAR – OLD } {Playlist IS XX[A]: 4 STAR – RANDOM } {Playlist IS XX[A]: 5 STAR – LEAST } {Playlist IS XX[A]: 5 STAR – OLD } {Playlist IS XX[A]: 5 STAR – RANDOM } {Playlist IS XX[A]: LEAST } {Playlist IS XX[A]: OLD } {Playlist IS XX[A]: RANDOM }	1000 songs by random

This system keeps the notion of weighting the music mixes towards recent and/or unrated music and these categories make up the last two playlists of the second tier :

Playlist	Conditions	Selection Limitation
XX[B]: SPICES & HERBS	Match ALL {Date Added IS IN THE LAST 3 months} {Last Played IS NOT IN THE LAST 6 days} {Kind DOES NOT CONTAIN Video} {Kind DOES NOT CONTAIN WAV} {Genre DOES NOT CONTAIN Audiobook} {Genre DOES NOT CONTAIN Comedy} {Genre DOES NOT CONTAIN Holiday} {Genre DOES NOT CONTAIN Podcast} {Rating IS NOT 1 star} {Rating IS NOT 2 stars}	300 songs by random
XX[B]: UNRATED	Match ALL {Rating IS 0 stars} {Last Played IS NOT IN THE LAST 9 days} {Kind DOES NOT CONTAIN Video} {Kind DOES NOT CONTAIN WAV} {Genre DOES NOT CONTAIN Audiobook} {Genre DOES NOT CONTAIN Comedy} {Genre DOES NOT CONTAIN Holiday} {Genre DOES NOT CONTAIN Podcast}	300 songs by random

Finally, these three second tier lists are combined into the final playlist:

Playlist	Conditions	Selection Limitation
XX[C]: FINAL POOL	Match ANY {Playlist IS XX[B]: BASE POOL } {Playlist IS XX[B]: SPICES & HERBS } {Playlist IS XX[B]: UNRATED }	1000 songs by random

When you make your top level mix playlists, you point them to [XX\[C\]: FINAL POOL](#) like you did with [DYNAMIC POOL](#) in the intermediate system. That's it, the end of the third system, you already know how to make top level playlists for listening. Go forth and populate your iPod with smartlists...

Chapter 2.5: Some tips and further ideas

The purpose of this document is not to show you the be-all, end-all method of smartlist organization, but instead to give some detailed examples so you can really dig into the smartlist system in iTunes and learn how to do it on your own. Generate other playlists, modify these schemes, come up with your own, do whatever you want. Your final goal is to have some set of top-level playlists that reflect your needs, not mine. Once you come up with that, you simply set the iPod to be synced to those top-level playlists (along with any necessary supporting lists) and then listen and enjoy. With the ideas presented in this document, you should be able to come up with a playlist scheme that works for you while minimizing the work needed on your part. To that end, here's some more ideas you might find useful.

Use time limits for selection criteria instead of number of songs

Using time units instead of number of songs for setting the upper selection limit for smartlists may make the total capacity defined by your smartlists more predictable. For example, in my music collection, the vast majority of the music is encoded around ~192 VBR kbps, but the length of songs varies massively, ranging from 2 minute 1960s pop songs to 45 minute Grateful Dead jams. When bit rate tends to be more or less the same, it is more consistent in terms of disk capacity to limit the base playlists by the total time rather than number of tracks. Of course, you can just go all the way and limit by file sizes if you find that more appealing.

An alternative to the least played category

iTunes uses really lazy tie-breakers for selection limitations on smartlists. Whenever two or more songs match the same smartlist limitation criterion, iTunes generally defaults to Artist as the secondary selection ordering. As such, smartlist limiting criteria that produce lots of ties will almost certainly result in clumping of artists. Selecting according least often played (as was used in both chapters' 2.3 and 2.4 playlist systems) is one of the criteria that produces many ties, potentially thousands in a large library.

An alternative to selecting according to least often played is to instead add a rule {Play count IS LESS THAN X}, where X is some number that represents the bulk of your “neglected” tracks and then limit selection at random. As the lower play counts in your library increase, you will have to edit this value in your smartlists to keep it working properly, but it achieves the same goal, nudging newer tracks to the front of the line, but now avoids artist clumping. As an example, the playlist **XX[A]: 4 STAR - LEAST** from page 27 would look like this if using this alternative method:

Playlist	Conditions	Selection Limitation
XX[A]: 4 STAR – LEAST	Match ALL {Rating IS 4 stars} {Genre IS NOT Holiday} {Last played IS NOT IN THE LAST 8 days} {Play count IS LESS THAN 3}	67 songs selected at random

A quick example of a weighted playlist system not based on ratings

My wife doesn't like to be bothered with managing her iPod, ergo, that's my job. Among the things she can't be bothered to do is rate songs other than in the case of something that comes up she hates. Hated songs she'll give one star to so I will remove it from her library, but otherwise ratings are simply not an

option for weighting her management scheme. Fortunately, I found a good solution that shows alternative ways to the ratings based systems explained earlier. I'm just going to give a sketch of the system since the naming conventions should tell you the gist of how the playlists work.

Her favorite band by far is the Grateful Dead, so they get their own special base level playlists: [Grateful Dead – Least](#), [Grateful Dead – Old](#), and [Grateful Dead – Random](#). These three smartlists represent one-third of the main mix.

Since there are no ratings to use, I used the groupings field as a work around. I created a smartlist, [GOOD STUFF \[RAW\]](#), that pulls in all the music from other favorite artists, selected all of it, and entered “GOOD STUFF” in the grouping field. If she expresses particular interest in an artist that isn't on [GOOD STUFF \[RAW\]](#), the artist is added. If she complains she's hearing too much of an artist that is on the playlist, the artist is removed. Two other smartlists allow me to make sure the “GOOD STUFF” grouping tag is present for every track on the [GOOD STUFF \[RAW\]](#) playlist and not present for any track not on this playlist. None of these three playlists is intended to be synced to an iPod; they exist so I can manage the grouping tag for weighting the mix on her iPod toward favored music.

To take advantage of the “Good Stuff” grouping tag, I make a smartlist: [GOOD STUFF \[REFINED\]](#) that, among other things, has the rule {Grouping IS Good Stuff}. Functionally, this single list replaces the multiple playlists based around 4 and 5 star rated songs in the earlier detailed playlist systems. [GOOD STUFF \[REFINED\]](#) defines another one-third of the main mix.

The final one-third of the main mix is defined by the same sort of [RANDOM](#), [LEAST](#), and [OLD](#) playlists you should be familiar with by now.

These seven base level smartlists are pulled into one super smartlist: [EVERYTHING AND MORE](#) from which I can base any number of more refined top level smartlists for listening.

Smartlists – Not just for breakfast anymore

You can use smartlists to manage any media content on your iPod. I give some ideas for podcasts and audiobooks in the next chapter, but video, be they movies, television series, home movies, etc., can be managed using the same strategies described in this document if you have a video enabled iPod.

Genius, some quick thoughts

The Genius feature released in 2008 by Apple has the potential to be one of the more innovative features on iPods. Genius, hypothetically, provides a polished and themed mix list, a nice alternative to the sort of top level smartlists described in this document that may or may not be particularly focused depending on your personal system. However, it is a compliment to smartlists, not a competitor nor replacement. Even if Genius playlists were to get so good you didn't want to listen to a smartlist based top-level playlist, you're still going to need a way to get a good selection of music on your iPod for Genius to work with in the first place, and that's going to be a smartlist based system.

Chapter 3.1: Podcasts & Audiobooks

Although the podcatching functionality of iTunes is perfectly acceptable, when it comes to podcast management, iTunes is less stellar. There is only one possible way to display your podcasts (reverse released order), podcasts must be downloaded with iTunes or they won't appear on the Podcast menu, management settings are global to all your podcasts, and so on. Basically, the only way to achieve any fine degree of control over podcasts is to set the iPod options to update all podcasts but only keep checked episodes. This works, but still leaves you at the mercy of iTunes unintuitive reverse released order as well as having to monkey about with checking and unchecking podcast tracks every time you refresh your iPod contents. Fortunately, smartlists are ideal for getting around Apple's unpolished implementation.

The first step to taking back good management of podcasts is to turn off the sync option on the Podcast tab of the iPod options. After stopping iTunes from managing your podcasts, it is very simple to do it with smartlists. I choose to have one smartlist per podcast, so some might look like this:

Playlist	Conditions	Selection Limitation
POD: DEMOCRACY NOW!	Match ALL {Album IS Democracy Now!} {Playcount IS 0}	5 songs selected by album
POD: ON THE MEDIA	Match ALL {Album CONTAINS On The Media from NPR/WNYC} {Playcount IS 0}	5 songs selected by least recently added
POD: SLATE MAGAZINE	Match ALL {Album IS Slate Magazine Podcasts} {Playcount IS 0}	5 songs selected by album

This method does require that podcasts are tagged with something consistently identifiable, such as a unique artist or album name. Fortunately, the vast majority of podcasts are tagged with such information. Similarly, selection limitation works best when track numbers or file names are consistently tagged (ordered selection can usually be accomplished via date added as well). However, even if you have to manually add some tag information to your podcasts to achieve better functionality, it's still better than the inflexible method that iTunes uses.

There are advantages in using smartlists to manage podcasts. You can set how many episodes to keep on a podcast by podcast basis. Some podcasts are just a few minutes long, others an hour or more, and we can't forget that many podcasts are video based now. Unless space is not an issue, it makes sense to set the number of episodes kept on your iPod based upon their relative space consumption and your rate of listening; the iTunes method forces you to choose one global number for kept episodes of all podcasts. Another improvement is that if you listen to your podcasts from these playlists, you can determine how the playlists are sorted instead of being stuck with the reverse released order the built in Podcast menu uses. In addition, listened to podcasts are dropped off the playlist rather than remaining until your next sync. A related possible advantage of listening through playlists is that files will play sequentially. When you listen through the Podcast menu, the iPod automatically stops playing after each

file. While I don't see the issue, I've seen enough complaints in forum discussions to realize this is something many users don't like.

Even when using smartlists to manage moving podcasts on and off your iPod, you may find you still want to use the Podcast menu on your iPod. No problem, nowadays, podcasts downloaded through iTunes will show up on the Podcast menu no matter if they are synced with the iTunes podcast management options or playlists. With a newer iPod, you even have the option of tucking the podcast management smartlists inside of an organization folder out of the way of your main use playlists and folders.

Audiobooks, even more than podcasts, lend themselves well to smartlists. First, there is no built in option for syncing anything other than all (checked) audiobooks, so unless you want to spend your time checking and unchecking audiobook tracks in iTunes, you're going to have to create a playlist and you might as well make it a smart one. Second, the iPod Audiobook menu is the worst of any of the special media kind menus on the iPod. Even though Apple finally gave us the ability to designate any file type as an audiobook in iTunes, they still have failed to fix the way audiobooks are displayed on the iPod. No matter how many different audiobooks, no matter how many files from each book are present, they all get displayed in one, big, long, flat list with no way to tell what you've listened to and what you haven't. Fortunately, smartlists get past the shortcomings of the iPod's built in audiobook functions nicely.

Audiobooks are handled exactly the same way as podcasts. You make smartlists based upon some unique identifier (usually Album, which is the book's title, and/or Artist, the author's name) and use the play count criteria for rotating book sections on and off your iPod.

Some example Audiobook playlists are given here:

Playlist	Conditions	Selection Limitation
AB: A Song Of Fire And Ice	Match ALL {Album CONTAINS A Song Of Fire And Ice} {Play count IS 0}	4 songs selected by album
AB: The Shining	Match ALL {Album IS The Shining [Unabridged]} {Play count IS 0}	4 songs selected by album

Although very simple, these playlists work great. You can keep an appropriate amount of each book on your iPod relative to how each book is arranged. For example, some audiobooks are in a few, large, several hour long files, others are split into multitudes of small 2 to 3 minute long files. Additionally, you don't ever have to keep track of which chapter or track you left off on; the smartlist will drop a track as soon as you finish it. Where you left off will always be the first track on the playlist when you come back to your audiobook.

Final words, etc.

Hopefully this document gave you some ideas about what you can do with smartlists and how to do those things. This all started off as an email to some friends about using smartlists. In its original iteration, it was virtually unreadable, I trust these later versions are an improvement on that.

For me, a lot of the early impetus to develop an efficient smartlist management system back in 2004 was that I had a 4 GB iPod mini and a huge library (it's currently exceeding 200 GB). However, even assuming that some future iPod could contain my entire library, I would still continue to use automated smartlists for the majority of my listening. First, because I would never want to expend the mental energy on always trying to think of something to listen to. Second, human nature being what it is, if you always choose your music, you will always find yourself listening to the same stuff. A huge part of why I like digital music revolution is the power it gives me to explore so much variety in music. Capacity issues aside, a non-human selector is a must, both for purposes of saving time and energy, as well as just getting me to explore everything in the library.

Copyright 2004-2009 Charles Hannum Jr

This is version 6.0 of this smartlist document.

The *iPod* and *iTunes* are products and trademarks of Apple, Inc.

Special thanks to those who have provided feedback and suggested inclusions for this paper. Further thanks to those who have found it useful enough to pimp it out to others.

I can be found at the forums at <http://www.iLounge.com>, user name “Code Monkey”.

This document created and published to PDF with the OpenOffice.org Writer application.

Appendix A: Available rules criteria for smartlists

These are the rules criteria you can filter with smartlists as of iTunes 8.1 and some ideas for what you can do with them:

Album – Lets you filter for the contents of the album id3 tag field. Can be used to block or include rule defined albums from a playlist. Can be filtered with the IS/IS NOT, CONTAINS/DOES NOT CONTAIN, STARTS WITH, and ENDS WITH conditionals.

Album Artist – Lets you filter for the contents of the album artist id3 tag field. Can be used to block or include rule defined album artists from a playlist. Can be filtered with the IS/IS NOT, CONTAINS/DOES NOT CONTAIN, STARTS WITH, and ENDS WITH conditionals.

Album Rating – Lets you filter for the relatively new iTunes album rating. Albums take on a “ghost” album rating based on the average of their individual track ratings or can be rated directly in iTunes. Possibly useful for eliminating content from “sub par” albums for a playlist. Can be filtered with the IS/IS NOT, IS GREATER/LESSER THAN, and IS IN THE RANGE conditionals.

Artist – Lets you filter for the contents of the artist id3 tag field. Can be used to block or include rule defined artists from a playlist. Can be filtered with the IS/IS NOT, CONTAINS/DOES NOT CONTAIN, STARTS WITH, and ENDS WITH conditionals.

Bit Rate – Lets you filter for the average bit rate for audio files. Not much use for managing content on iPods but can be useful for general library management. Can be filtered with the IS/IS NOT, IS GREATER/LESSER THAN, and IS IN THE RANGE conditionals.

BPM – Lets you filter for the numerical value stored in the BPM id3 tag field. BPM stands for Beats Per Minute, a measure of musical tempo. If you actually have your music tagged with its proper BPM, this could be used for some interesting smartlist systems. Otherwise, it's a field that you can fill in with any numerical value if you want to have another way of filtering content. Can be filtered with the IS/IS NOT, IS GREATER/LESSER THAN, and IS IN THE RANGE conditionals.

Category – Lets you filter for the contents of a text tag field on most podcasts. I have no idea what use it could be since it's so non-standard. For example, the most common thing it's filled in with in my library's podcasts is the word, “podcasts”. Second most common values are redundant values for the Artist field. Only rarely is it something useful like an actual category (*e.g.* ABC's podcast for the television show *Lost* has the category of “Movies & Television”). At any rate, if you can think of something to do with it, it can be filtered with the IS/IS NOT, CONTAINS/DOES NOT CONTAIN, STARTS WITH, and ENDS WITH conditionals.

Comment – Lets you filter for the contents of the comment id3 tag field. Can be used to block or include content from a playlist based on the comment field. Can be filtered with the IS/IS NOT, CONTAINS/DOES NOT CONTAIN, STARTS WITH, and ENDS WITH conditionals.

Compilation – Lets you filter if the Compilation flag is set to true or false. Can be filtered with IS TRUE and IS FALSE conditionals. Only useful if you want to create a playlist of exclusively compilation tracks or a playlist with no compilation tracks. It is of dubious value for iPod management.

Composer – Lets you filter for the contents of the composer id3 tag field. Can be used to block or include rule defined composers from a playlist. If you don't use the composer field for composers, it can be used as another field for categorizing and filtering music or other media. Can be filtered with the IS/IS NOT, CONTAINS/DOES NOT CONTAIN, STARTS WITH, and ENDS WITH conditionals.

Date Added – Lets you filter for the date the media was added to your iTunes library. I generally use it for weighting toward recently added music, but can be used for anything related to when media was added to your library. For example, you can make a themed playlist based on music you added during a particular year. Can be filtered with the IS/IS NOT, IS BEFORE/AFTER, IS IN THE RANGE of specific dates or IS/IS NOT IN THE LAST X (DAYS/WEEKS/MONTHS) conditionals.

Date Modified – Lets you filter for the date the actual files were last modified. Not obviously useful for iPod management but has applications for library management. This rule won't necessarily pick up on file changes that take place outside of iTunes. Can be filtered with the IS/IS NOT, IS BEFORE/AFTER, IS IN THE RANGE of specific dates or IS/IS NOT IN THE LAST X (DAYS/WEEKS/MONTHS) conditionals.

Description – This filters for the podcast specific tag field for its description. I don't see it as very useful for smartlist management of podcasts since, like Category, the contents are either redundant or too variable to filter for management purposes. Can be filtered with the IS/IS NOT, CONTAINS/DOES NOT CONTAIN, STARTS WITH, and ENDS WITH conditionals.

Disc Number – Lets you filter for the disc number id3 tag field. You're limited to filtering for the disc number itself, no filtering for the “of X” part of the tag. Not very useful for smartlist management of an iPod, but has niche applications for library management. Can be filtered with the IS/IS NOT, IS GREATER/LESSER THAN, and IS IN THE RANGE conditionals.

Genre – Lets you filter for the contents of the genre id3 tag. This is one of the more useful tags for smartlist management since inclusion and exclusion of genres is one of the easier ways to create themes for mix lists. Can be filtered with the IS/IS NOT, CONTAINS/DOES NOT CONTAIN, STARTS WITH, and ENDS WITH conditionals.

Grouping – Lets you filter for the contents of the grouping id3 tag. Groupings allow you to create custom meta-fields that are inclusive of multiple genres, artists, etc. It's a good tag field to give you more filtering options for media in your library, particularly since it has no fixed purpose. Can be filtered with the IS/IS NOT, CONTAINS/DOES NOT CONTAIN, STARTS WITH, and ENDS WITH conditionals.

Kind – Lets you filter for the iTunes description for a file type. You may need to use the “Get Info” or turn on the Kind column until you are familiar with the iTunes specific descriptors for different file types. For example, an mp3 file is “MPEG audio”, an AAC file is “AAC audio”, and a wave is “WAV audio”. Video of file type mp4 is “MPEG-4 Video” and DRM protected video from the iTunes store is “Protected MPEG-4 Video”. This rule criterion is the most reliable way for including or excluding the different media types iTunes manages. Can be filtered with the IS/IS NOT, CONTAINS/DOES NOT CONTAIN, STARTS WITH, and ENDS WITH conditionals.

Last Played – Lets you filter for when the tracks were last played. One of the most powerful rules for rotating music in mix playlists, you can determine how often any particular file, genre, rating, etc. is

allowed to cycle onto your top level playlists. Can be filtered with the IS/IS NOT, IS BEFORE/AFTER, IS IN THE RANGE of specific dates or IS/IS NOT IN THE LAST X (DAYS/WEEKS/MONTHS) conditionals.

Last Skipped – Lets you filter for the date the tracks were last skipped. Can be used to get skipped tracks off of playlists temporarily. If you add a rule to a top level playlist preventing skipping within a certain period, it will remove skipped tracks for that length of time. Can be filtered with the IS/IS NOT, IS BEFORE/AFTER, IS IN THE RANGE of specific dates or IS/IS NOT IN THE LAST X (DAYS/WEEKS/MONTHS) conditionals.

Name – Lets you filter for the contents of the track name id3 tag. This can be used for making lists with a particular theme (*e.g.* a mix made from songs with the word “sunshine” in the title) or for excluding certain words from playlists (*e.g.* excluding songs with obvious profanity from a playlist). Can be filtered with the IS/IS NOT, CONTAINS/DOES NOT CONTAIN, STARTS WITH, and ENDS WITH conditionals.

Play Count – Lets you filter for the number of times a track has been played according to iTunes. Can be used for things like finding your least played tracks, tracks you must really love since they've been played so much, or excluding tracks based on play count. In cases that you generally only listen to a track once, such an audiobook or podcast, Play Count also becomes a great way of cycling smartlists involved in their management. Can be filtered with the IS/IS NOT, IS GREATER/LESSER THAN, and IS IN THE RANGE conditionals.

Playlist – Lets you require that tracks are either from or not from a particular playlist. If you've read this document, you already know what to do with this one. Can be filtered by the IS/IS NOT conditional.

Podcast – Lets you filter if the podcast flag is set to true or false. This flag is automatically set by iTunes when it downloads a podcast and cannot be edited. The rule can be used an alternate way of including or excluding podcasts from a smartlist. Can be filtered with IS TRUE and IS FALSE conditionals.

Rating – Lets you filter for the value of the iTunes specific rating (there is an id3 tag for rating, also from 0-5 stars, but iTunes uses its own metadata version). Assuming you rate your music, this is one of the most powerful ways to weight otherwise randomized music mixes toward your favored tracks. Can be filtered with the IS/IS NOT, IS GREATER/LESSER THAN, and IS IN THE RANGE conditionals.

Sample Rate – Lets you filter for the sample rate of files. Most music files should be 44.1 KHz (Redbook CD standard), but it is possible you have media that is higher or lower (for example, audiobooks tend to be mono and 22.05 KHz). Probably not useful for iPod management, but could have niche library management applications. Can be filtered with the IS/IS NOT, IS GREATER/LESSER THAN, and IS IN THE RANGE conditionals.

Season – Lets you filter for the value of the season iTunes field for television shows. If you're managing a television series on a video enabled iPod, this rule can be used to include or exclude rule defined seasons. iTunes will let you assign a numerical value to this field for any media type, not just video, so you might be able to come up with a custom application. Can be filtered with the IS/IS NOT, IS GREATER/LESSER THAN, and IS IN THE RANGE conditionals.

Show – Lets you filter for the contents of the show iTunes field for television shows. If you're managing a television series on a video enabled iPod, this rule can be used to include or exclude rule defined shows. In general, it's a redundant field as the Album field is usually populated with the identical value or one similar enough for it not to matter. iTunes will let you assign a value to this field for any media type, not just video, so you might be able to come up with a custom application. Can be filtered with the IS/IS NOT, CONTAINS/DOES NOT CONTAIN, STARTS WITH, and ENDS WITH conditionals.

Size – Lets you filter for the individual file size in megabytes of media in the iTunes library. Not particularly useful, but if you wake up one day and think to yourself that you want an ~500 MB smartlist made of one hundred ~5 MB audio files, it's there. Can be filtered with the IS/IS NOT, IS GREATER/LESSER THAN, and IS IN THE RANGE conditionals.

Skip Count – Lets you filter by the iTunes skip count metadata. iTunes tracks the total number of times a file has been skipped (a file is considered skipped after it has played at least 5 seconds but not more than 20 seconds). Probably not much use for iPod management, but could be used to find songs you have skipped so many times that you might want to consider excluding them from playlists in the future. Can be filtered with the IS/IS NOT, IS GREATER/LESSER THAN, and IS IN THE RANGE conditionals.

Sort Album – Lets you filter by the contents of the sort album iTunes field. I can't imagine any iPod management scheme it would be useful for, but could be useful for library management to locate tracks that need the sort album information fixed. For example, it is common to preface names that start with a numeral with the letter 'a' and then a space so they will sort in proper alphanumerical order instead of the numbers last order iTunes inexplicably adopted mid 1997. A smartlist that pulled in all names starting with numerals would let you quickly locate and change them appropriately. Can be filtered with the IS/IS NOT, CONTAINS/DOES NOT CONTAIN, STARTS WITH, and ENDS WITH conditionals.

Sort Album Artist – Lets you filter by the contents of the Sort Album Artist iTunes field. I can't imagine any iPod management scheme it would be useful for, but could be useful for library management to locate tracks that need the sort album artist information fixed (see **Sort Album** for more information). Can be filtered with the IS/IS NOT, CONTAINS/DOES NOT CONTAIN, STARTS WITH, and ENDS WITH conditionals.

Sort Artist – Lets you filter by the contents of the Sort Artist iTunes field. I can't imagine any iPod management scheme it would be useful for, but could be useful for library management to locate tracks that need the sort artist information fixed (see **Sort Album** for more information). Can be filtered with the IS/IS NOT, CONTAINS/DOES NOT CONTAIN, STARTS WITH, and ENDS WITH conditionals.

Sort Composer – Lets you filter by the contents of the Sort Composer iTunes field. I can't imagine any iPod management scheme it would be useful for, but could be useful for library management to locate tracks that need the sort composer information fixed. While I don't know of any composers whose names start with numerals like the example I gave in **Sort Album**, it is commonplace write composers' names as “last name, first name” so that Ludwig von Beethoven sorts and is located at “Beethoven”, not “Ludwig”. You could use smartlists to find all composers with a comma in the name and have them displayed as “first last name” but sorted by “last, first name”. Can be filtered with the IS/IS NOT, CONTAINS/DOES NOT CONTAIN, STARTS WITH, and ENDS WITH conditionals.

Sort Name – Lets you filter by the contents of the Sort Name iTunes field. I can't imagine any iPod management scheme it would be useful for, but could be useful for library management to locate tracks that need the sort name information fixed (see **Sort Album** for more information). Can be filtered with the IS/IS NOT, CONTAINS/DOES NOT CONTAIN, STARTS WITH, and ENDS WITH conditionals.

Sort Show – Lets you filter by the contents of the Sort Show iTunes field. I can't imagine any iPod management scheme it would be useful for, but could be useful for library management to locate videos that need the sort show information fixed (see **Sort Album** for more information). Can be filtered with the IS/IS NOT, CONTAINS/DOES NOT CONTAIN, STARTS WITH, and ENDS WITH conditionals.

Time – Lets you filter according to the time length of tracks. Useful for including or excluding tracks either because they're too short or too long. For example, you don't want a 30 minute jam for your 20 minute commute playlist. Can be filtered with the IS/IS NOT, IS GREATER/LESSER THAN, and IS IN THE RANGE conditionals.

Track Number – Lets you filter according to the numerical value in the track number id3 tag. Like **Disc Number**, can only filter for the track number itself, not the “of X total tracks” part of the field. Possibly interesting for making themed playlists, otherwise it's not much use other than looking for tracks missing their track number for the purpose of library management. Can be filtered with the IS/IS NOT, IS GREATER/LESSER THAN, and IS IN THE RANGE conditionals.

Video Kind – Lets you filter for the iTunes video kind flag (Movie, Music Video, or TV show). Has some potential for smartlist management of video files but largely redundant. Can be filtered by IS MOVIE, IS MUSIC VIDEO, or IS TV SHOW conditionals.

Year – Lets you filter for the numerical value in the year id3 tag. Useful for creating themed playlists from particular decades or musical eras. Can be filtered with the IS/IS NOT, IS GREATER/LESSER THAN, and IS IN THE RANGE conditionals.

Appendix B: “Selected By” Criteria

There are the ways you can select and limit media matching the rules in a smartlist (accurate as of iTunes version 8.1):

Random – If you use this category then matching tracks are chosen at random from all possible matching tracks.

Album – Not necessarily, as you might think, a way to get smartlists to select whole albums. When you choose this category, matching tracks are chosen in alphabetical order according to their album name until the selection limit is reached. For example, all matching tracks from “Abbey Road” will be selected before matching tracks from “Beatles For Sale”. The secondary order they are selected within each album is determined by track number if that field is populated, otherwise they are selected according to track name. This selection method is most useful when managing something like an individual podcast subscription or audiobook; tracks will be moved on and off the playlist in correct order as long as track number or names are populated and sequential. It can also be used as the closest thing that exists in iTunes for generating a list of randomized albums although there will almost certainly be at least one partial album.

Artist – Similar to the **Album** “selected by” method, matching tracks are chosen in alphabetical order according to the Artist tag. Secondary selection order is Album, then Track Number, followed by Track Name. I honestly have no idea why you would ever use this. In any scenario you were working with a single artist, selecting by album would order everything exactly the same, and any scenario you were working with a mix, you would almost certainly be better off selecting at random and sorting the playlist according to Artist.

Genre – Identical in concept to the **Artist** “selected by” method. Matching tracks are chosen according to the alphabetical order of their Genre tag. Secondary selection order is Artist, then Album, then Track Number, followed by track Name. Like **Artist**, this option has no rational application I can think of, but it's there if you want it.

Name – Similar in concept to the **Album** and **Artist** “selected by” methods. Matching tracks are chosen in alphabetical order according to their track name, there is no secondary selection order. Like **Artist**, this option has no rational application I can think of, but it's there if you want it.

Highest Rating – Matching tracks are selected according to rating, starting at whatever is the highest rating and working downward. For example, it will first grab any 5 star tracks, then 4 star tracks, and so on until the selection limit is reached. Secondary selection order is Artist, then Album, then Track Number, then Track Name. This selection method can potentially be used for smartlist management because it allows you to grab the “cream” from some other set of rules without forcing a particular high end for the rating. For example, you could create a smartlist to find all music from the years 1984 through 1987 in your library and then grab the 20 highest rated tracks without enforcing any particular rating; in other words, they could all turn out to be only 3 stars, or they could all be 5 stars. The downside is that this method has a high probability of clumped artists.

Lowest Rating – Works exactly like **Highest Rating** except that it selects from matching tracks starting with the lowest rating and working upward. Unlike Highest Rating, however, I can't imagine for what scenario this “selected by” method is useful.

Most Recently Played – Matching tracks are selected according to last played date and time, starting with the most recent date and time and working backward. Probably not the most useful “selected by” method, but with some imagination could have some uses, something like the “The 20 most recently played tracks that haven't been played since before one year ago and have names that start with 'T'” playlist.

Least Recently Played – Selects from matching tracks according to last played date and time, starting with the oldest last played date and time and working forward. This selection method is very useful for forcing full library rotation and was used extensively in this document.

Most Often Played – Selects from matching tracks according to the highest play count and working downward. It is potentially a way to select for your favored music if you don't use ratings and often listen to music by manually selecting it instead of using mix lists or shuffling. Since secondary selection order is Artist, then Album, then Track Number, and then Track Name this method is very likely to produce artist clumping.

Least Often Played – Matching tracks are selected according to the lowest play count, working upward. Can be used, as it was in this document, to weight toward recently added music and/or “forgotten” tracks. Since secondary selection order is Artist, then Album, then Track Number, and then Track Name this method is very likely to produce artist clumping.

Most Recently Added – Matching tracks are selected according to the date and time they were added to the iTunes library, starts with the most recently added date and time and works backward. Could be used as alternative to the Date Added rule for making a playlist weighted toward recently added music, but mostly useless for smartlist management of an iPod.

Least Recently Added – Matching tracks are selected according to the date and time they were added to the iTunes library, starts with the least recently added date and time and works forward. Could be used as alternative to the Date Added rule for making a playlist weighted toward older music in your library, but mostly useless for smartlist management of an iPod.

Appendix C: Smartlist Tricks

Although this document was focused on using smartlists for managing iPod content, there are a few tips and tricks that are helpful to know.

Find tracks whose file is missing

Through ignorance or accident, sometimes you wind up with tracks that are listed in your iTunes library but don't have valid paths to the actual file. Maybe their drive letter got renamed, maybe you wanted them deleted, but did so from your OS' file browser instead of with iTunes, maybe a virus or another music program renamed them. Whatever the cause for the disconnect, if you go to play or otherwise work with these tracks, the dreaded exclamation point icon appears next to the track name. If you try to play such a track, iTunes informs you that it cannot find the file and asks if you want to try and locate the file for iTunes. How you deal with the broken file links is outside the scope of this little tip, but the first thing you have to do is find all your tracks that have broken links to their files so you can take action.

Make a smartlist, [FIND BROKEN FILE LINKS](#), and a regular playlist, [GOOD FILE LINKS](#).

Playlist	Conditions	Selection Limitation
FIND BROKEN FILE LINKS	Match ALL {Bit Rate IS GREATER THAN 1} {Playlist IS NOT GOOD FILE LINKS .	none

[FIND BROKEN FILE LINKS](#) will pull in every single track in your iTunes library at first. Select all the tracks in the list view for [FIND BROKEN FILE LINKS](#) and drag them onto the playlist icon for [GOOD FILE LINKS](#). When you manually drag tracks like this, iTunes will verify their file paths and only those tracks with good links will be added to [GOOD FILE LINKS](#). As soon as you do this, all the tracks with good file paths are dropped from [FIND BROKEN FILE LINKS](#) and just the tracks with bad paths (they will all now have the exclamation point icon) will remain for you to deal with as is appropriate.

Find missing (or not missing) tags

You can search for empty tags by not entering anything into the search field for the rule. For example, the rule. {Album IS} will return all tracks that have nothing entered in the album field. This can be a way to relatively quickly get your tags in order if you need to do so. The opposite is also true, if you don't want anything stored in a particular field either because you want it deliberately empty or just need to clear it so you can use it for another management strategy, you can use smartlists to quickly find all the non-empty tracks. For example, if you wanted to clear the composer field you could use the rule {Composer IS NOT} to return all tracks that have *something* entered in the composer field and clear them in one batch.

Manage duplicates easier

iTunes has the ability to show you all the tracks it thinks are duplicates (currently under the **File** menu). This shows you all tracks in your library (or playlist) that have identical artist and track names. The limitation is that you may not want to either delete or rename all these alleged duplicates. For example, a

track might appear on multiple movie soundtracks and you want to keep those albums intact. If your library is large enough, you can wind up with hundreds or more of these intentional duplicates, which makes finding the duplicates you do want to delete or rename difficult.

A trick you can do with smartlists is to make a smartlist, [TRUE DUPLICATES](#), and a regular playlist, [POSSIBLE DUPLICATES](#).

Playlist	Conditions	Selection Limitation
TRUE DUPLICATES	Match ALL {Playlist IS POSSIBLE DUPLICATES } {Composer DOES NOT CONTAIN duplicate filtered}	none

From your main library, select “Show Duplicates” and then select all the tracks it returns and drag them over to [POSSIBLE DUPLICATES](#). In [TRUE DUPLICATES](#), sort the list according to track name and collapse album art to make reading it easier. Go through the list and delete anything that needs deleted, rename anything that needs renaming, and leave your intentional duplicates alone. Select all these tracks and, with the Get Info function, fill in the composer field (see my note below) with “duplicate filtered”.

In the future, whenever you want to screen your library for unintentional duplicates, clear the contents of [POSSIBLE DUPLICATES](#) and repeat the steps from the above paragraph. Tracks you've already screened in the past will already be blocked from [TRUE DUPLICATES](#), making such screening considerably faster and easier.

Note: In my example [TRUE DUPLICATES](#), I used the composer field for indicating a file has already been screened by me, but any field you don't normally use can be substituted. This might even be a good place for those extraneous television show fields you can add for any track now.