Automated Listener Requests

Introduction

One of the key features of any radio station is interaction with it's listeners and for a music based station this will be in the form of taking requests from them in the form of a particular track. Many stations today provide this facility in the form of phone, text (via SMS mobile phones) or via the station's website.

A number of smaller stations also have periods of unattended operation, where there is no presenter present and the station puts out continuous music (interspersed with station jingles and possibly advertisements) – either by choice or due to staffing/cost commitments. Often this period of *automation* is during overnight hours, typically between midnight and 6am.

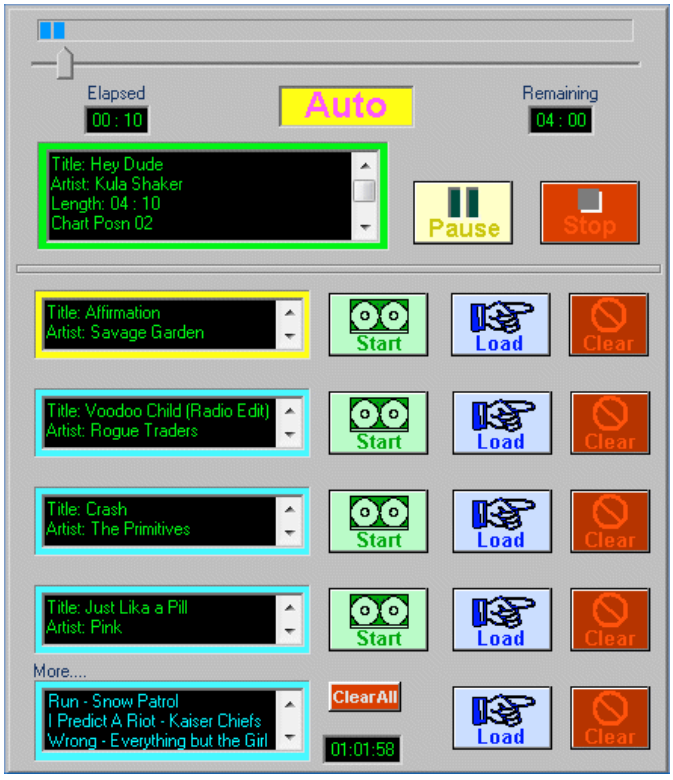
The idea of Automated Listener Requests is to combine the two concepts together such that listeners can make a request during a period of automation and have the system play it out without any human interaction.

This document describes the features & capabilities within the OAS Playout system to support this idea. It's important to note that Playout itself cannot provide a complete system to achieve this and some of the concepts and requirements here will require implementation by someone with experience in IT systems.

Automation in OAS Playout

Firstly, let's cover the support within Playout for automation – the ability for the system to sequence tracks automatically without requiring presenter intervention.

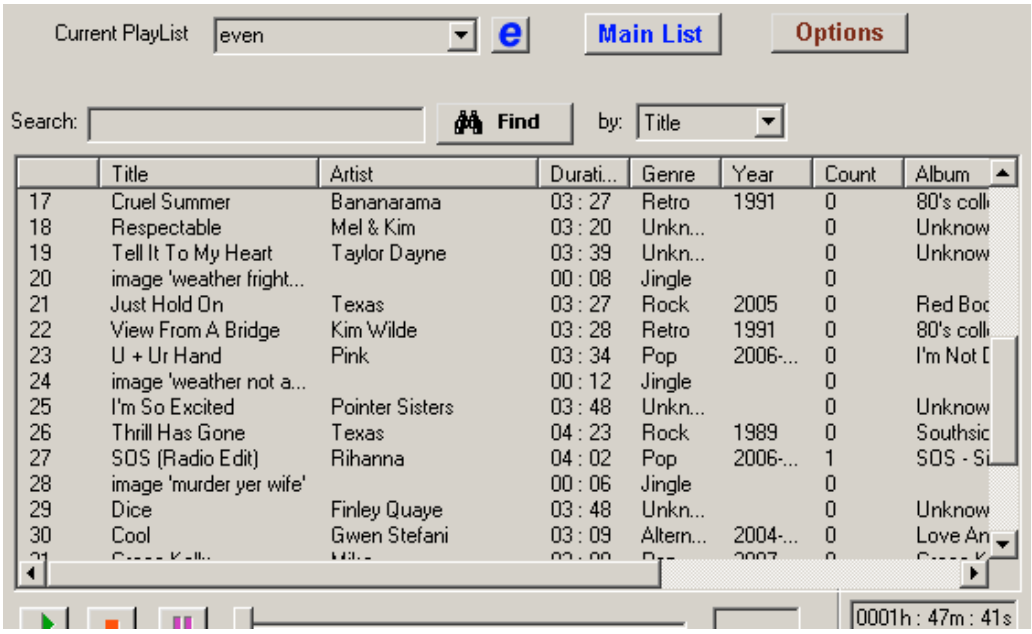
The simplest approach is to use Playout's *auto* mode then load all the tracks into one of the main players.



Playout in *Auto* Mode

Hit the *Start* key and off it goes. You can load as many music tracks, jingles and adverts into the player as is necessary to support the period of automation.

Whilst this can be done manually, by picking tracks from the main *Music*, *Jingles* and *Adverts* categories in the player it is far quicker (and simpler) to create a Playlist within the Manager tool which includes the all the tracks you want to include in the sequence.

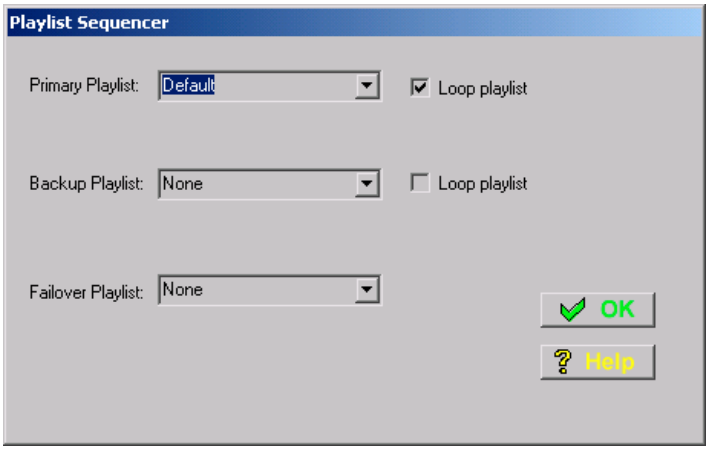


An automatically generated playlist in Playout Manager

Playout Manager includes tools, which allow the automatic creation of a playlist, inserting jingles and/or adverts into the list at predefined intervals. The music tracks are picked randomly from the music list based on rules which you can define. For more information on this see the *Playout Manager online help* and the *Playlisting Guide* (on the *Start* menu under OAS Playout).

Once you have a playlist, it can then be easily loaded into Playout.

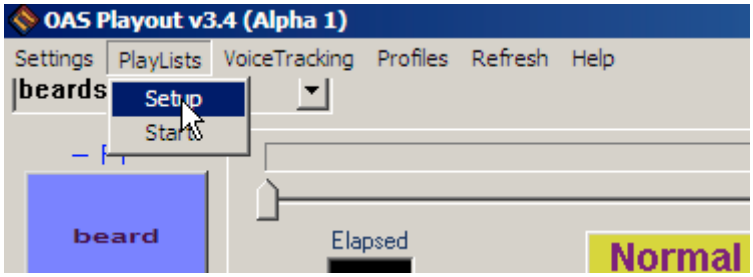
Version 2.5 of OAS Playout introduced an even quicker mechanism to automatically run a playlist. From the *Playlists* menu in OAS Playout, choose the *Setup* option and it is then possible to define the playlist you want to run.



Define a playlist for automation

In this example the playlist *Default* is chosen for automation.

When the time comes to run the playlist, the presenter simply picks *Start* from the menu and the playlist is then run.



Automation menu in Playout

At first glance there seems to be very little difference in the two approaches in achieving automation in Playout however they do work very differently.

In the first instance, if you observe the player once automation is underway, you will notice that all the user controls on the player have been disabled (to prevent inadvertent interruptions) and more importantly it has only loaded the first few tracks – regardless of how large the playlist is.

Playout in automation mode

In fact if you observe it in operation, you will note that as the player slowly empties (as track's complete) a few more are added to 'top up' the list.

There are other benefits too in using this mode of operation over the 'simple' approach including error handling and the ability to loop the list should the need arise – this is all covered in *Playout's Online Help* and in the *User Manual* – see the section on *Automation*.

However it is this 'trickling' of tracks into the list which is the key to supporting the listener request concept. What Playout is doing is rather than loading the entire playlist into the system in one go, it keeps checking the playlist for the next "unplayed" track to add. This adds a degree of additional flexibility because it allows you to change or add tracks to a playlist whilst it is running. For example, you could decide to extend the playlist because a presenter has been delayed. This can simply be done

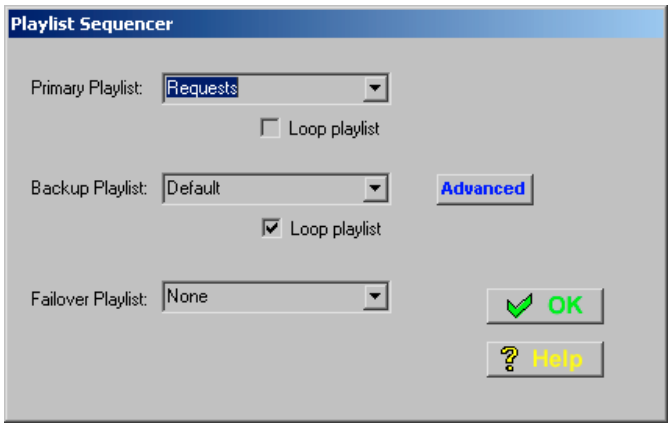
by adding new tracks to the existing list. You can even regenerate the entire playlist and Playout will pick it up from the start once it loads the next few tracks into the system.

Listener request concept in Playout

This section covers how the listener requests are handled within Playout. The concept is quite straightforward – any requests will be added into another Playlist which will then be processed by Playout (quite how they get into this list will be covered a bit later).

The Playlist can be created in the same way as any other ie. via Playout Manager and for clarity we will simply call it 'Requests'.

Once it is created, the automation settings in Playout (via the *Playlists* menu in Playout, choose the *Setup* option) need to be adjusted to look similar to the following.



Playlist settings for requests

The newly created (and empty) list *Requests* is now marked as the *Primary* list whilst your normal automation list is set as the *Backup* list. You shouldn't set the *Loop playlist* button on the Requests list but it can be set on the backup one (this will ensure that if all tracks from the list complete, it will be restarted therefore avoiding "dead air").

What this has done is instruct Playout to perform the following each time it needs to "trickle" more tracks into the Player:

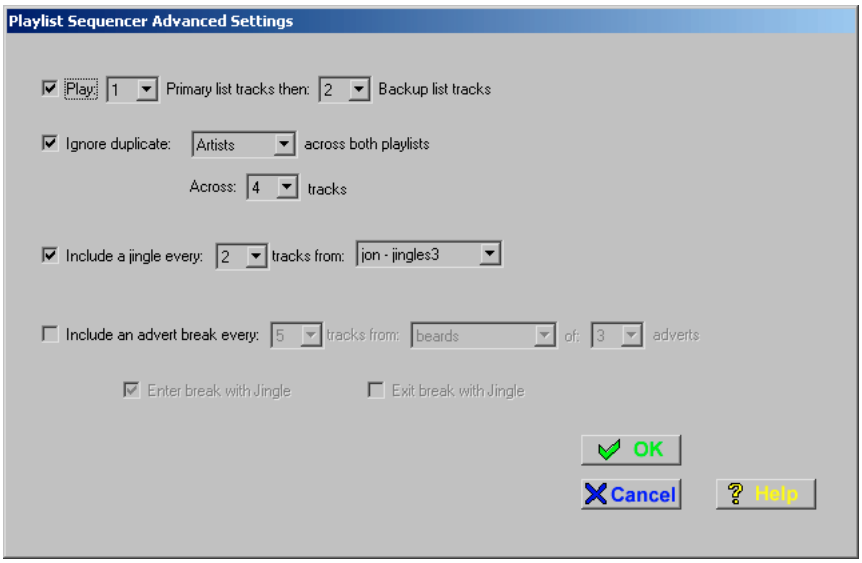
Firstly, it looks at the *Requests* playlist and looks to see if there are any "unplayed" tracks listed held there. If so, they are loaded into the player. If there are no tracks available, it then goes to the Backup playlist – ie. your normal automation list.

To get a feel for this, you can run Playout up and manually add a few tracks into the *Request* list through Playout Manager. You should see them automatically be added into the player as the automation proceeds.

This has then established the mechanism behind how listener requests will be handled inside Playout. However there are two significant problems with this simplistic approach. The first is the issue of a listener requesting either the same track or a track by the same artist, which has just been played. This could occur either because someone else has just requested a track or it is included in your main automation list. There needs to be some mechanism to avoid tracks being repeated too often.

The second issue is in the scheduling of your adverts and jingles. If you've included them in your automation list, typically at a predetermined interval (say every 5 tracks) then this is going to be upset if you get a lot of requests in at once. You could get 10 (or more) requested tracks played without going to an ad break or playing a jingle.

Both these issues are addressed by settings, which are specified by pressing the *Advanced* button when you set the playlists for automation.



Advanced Sequencer Settings

This dialog allows the refining of how the two playlists are processed within Playout.

The first issue, that of duplicate tracks is handled by setting the '*Ignore duplicate Artists across both playlists*' box (it also allows Albums to be specified instead for stations which use music more governed by album than artist). This ensures that if Playout encounters a track by the same artist (and by definition, the same track) from either playlist, it will hold of playing it until a further 4 tracks have been played. The exact figure can be changed to suite.

The second issue, that of the inclusion of adverts and jingles requires a slight adjustment to the creation of the automation playlist. Recall that up till now, these have been included in the automation playlist itself however in order for the correct sequencing of ads & jingles they now no longer should be. Instead, you need to define the jingles (and/or adverts) in two separate *Custom Audio Cliplists* (the same mechanism as is used to create Button Wall definitions in Playout Manager). These two lists then need to be selected from the drop downs in the 'Include a jingle every 'n' tracks' and 'Include an advert break every 'n' tracks'. There are then further options which allow an ad break to start and/or end with a jingle if required.

With these options set, Playout will then automatically schedule jingles and/or adverts at the correct intervals during playback from both lists. Jingles are picked at random from the provided jingle list whilst adverts are picked sequentially from the list, looping once all the ads have been played once. **What this also means is that you should now no longer include jingles, adverts etc. in the playlists themselves** (whether generated manually or automatically via a Playlist Generator) but allow Playout to schedule them for you via the method just outlined.

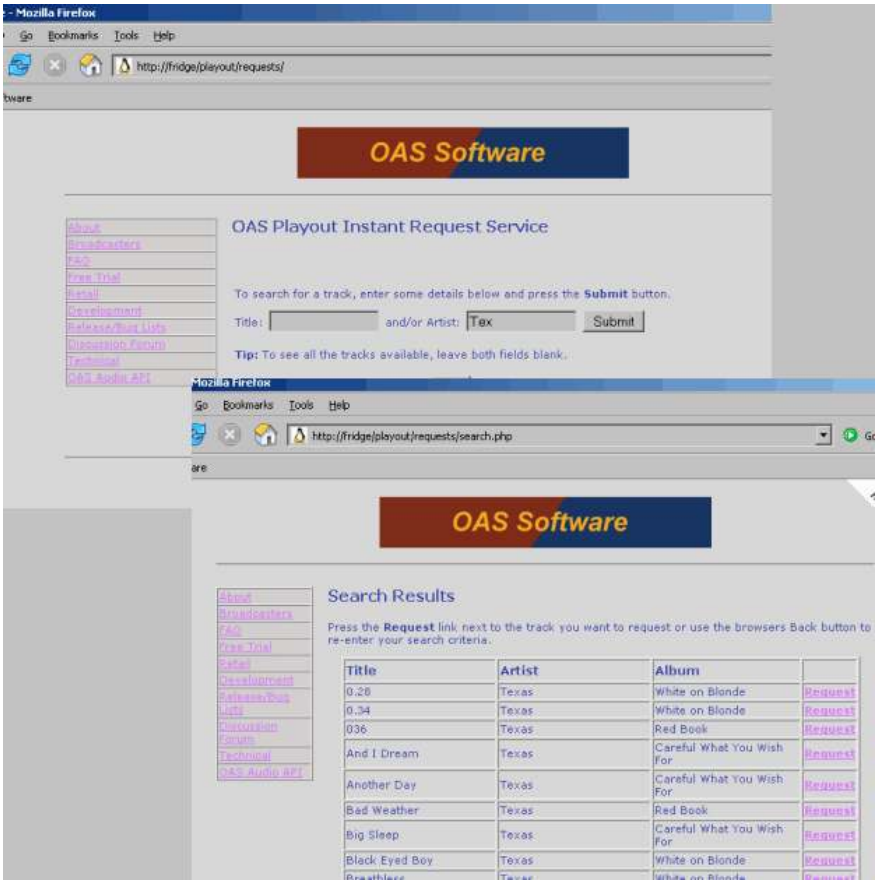
The Advanced dialog provides one further option, that of forcing a mix of tracks from both the main automation & request list instead of always choosing tracks from the request list. This may provide a better mix of tracks and it is up to you as to whether to enable this option or not.

With these options in place, a system is now in place to allow Playout to successfully include listener requests to be seamlessly included during periods of automation. All that is required now is the mechanism to get the tracks into the *Request* playlist.

Populating the *Requests* Playlist

The remaining issue is the mechanism required in getting the listener requests into the playlist. As outlined early on, current technology allows listeners to interact with their station principally in 3 different ways – by phone, mobile phone text or a web site.

First we consider requests via a web site. One approach to this is to enable listeners to directly search the Playout database itself, specifying desired title, artist etc. Once they have found their desired track, pages on the site allow the track to be added into the *Requests* playlist.



Requests via the web

The basis for such a system already exists and has been used successfully on at least one small-scale radio station. It is not overly difficult to set up if you have some IT/web design experience and the resources available, mostly because the Playout database is stored in an open standard, which can be queried from many server-side scripting languages eg. PHP. There is a tutorial including sample scripts which cover how to go about setting up a site to achieve just this on the OAS site at:

http://www.onasticksoftware.co.uk/playout-interaction.html

Requests received via this mechanism I term “exact” requests because the listener can determine exactly what he/she wants from the database.

There are implementation (and security) considerations though to consider with this approach, not least having the knowledge and resources to implement such a system. It also doesn’t lend itself to supporting requests via any other means that a PC with a direct Internet connection (except possibly a mobile phone with WAP capability).

The other two forms of making requests – phone & text are “inexact” requests because the listener has to guess the title (and/or artist) of a particular track and either phone up and say it or put it into a text. Therefore there is an extra degree of complication in terms of:

Does the station even have the track they want?
The title/artist phrased by the listener may not match exactly the phrasing held in our database?

A difficult one handled during testing by the OAS team being the track by “Queen”:

“Hammer to fall” being the title held in the database. A listener requests “Waiting for the hammer to fall”.

If it is not possible (or desirable) to implement the database searching web pages concept mentioned previously, the only route available to submitting requests via the internet would be either email or a simple form on the station’s web site form (where the listener manually types in their request details). These too then become “inexact” requests, requiring some form of processing to determine the closest matching track.

OAS Request Manager

OAS Request Manager is an optional additional component of the OAS Playout system designed to try and match these “inexact” requests to the closest matching track held in the database. It then puts the track into the Requests playlist. It can run on the same machine hosting Playout or a separate one although it needs direct access to the Playout database. Once activated, it sits in the system tray (near the clock) and handles requests in the background.

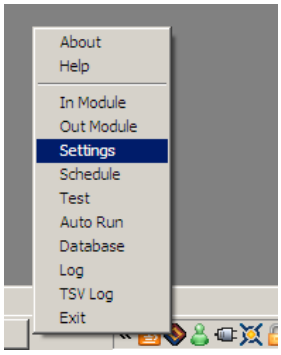
OAS Request Manager

At the moment, all this “inexact” request notion is a bit abstract, so let’s consider the obvious practical use here – handling requests received by mobile phone text.

Firstly, there needs to be a mechanism in place to get the texts into the Request Manager. How this is achieved is in part down to the IT system you have in place however a common approach is to provide an “SMS to Email” gateway, which in simple terms means the text messages arrive in an email inbox. This also lends itself nicely to receiving requests via email.

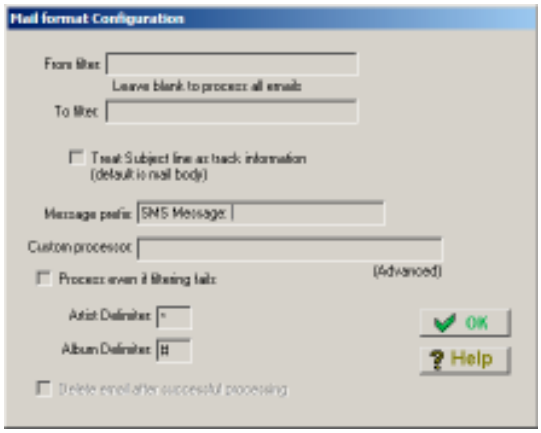
In simple terms then, Request Manager will look for new mail arriving in the default email inbox on the machine it is running on. It will then read the first line of any new mail and assume that it is intended as a request, which is then processed for a closest match and if a matching track found, stored in the Request playlist.

To do this, Request Manager needs some configuring. This is controlled via a menu activated by pressing the Right Hand mouse button over the Request Manager icon in the system tray.



Configuring Request Manager

First, configure activate the email settings via the *In Module* menu option.



In-Module email settings

The key things here are the following:

To/From filter: If you only want to treat emails sent from or to a specified address, put the address(es) in these dialogs. Leave them blank to treat all mail items as requests.

Treat Subject line as track information: Check this box if your SMS provider puts the text message into the Subject of the email (rather than the message body).

Message Prefix: The default mechanism for Request Manager to process emails is to assume the first alphanumeric character (ie. 'A-Z' or '0-9') it encounters in the email is the start of a request, then read to the end of the line. If your SMS provider formats the SMS messages in a special way this provides a mechanism to look for certain keywords which indicate the start of the actual message. Consider the following example based on information sent from a real radio station:

```
Shouts Data: Array
(
    [shoutdate] => 2009-03-03+20:36:21
    [shoutmessage] => ImageFM+Queen+Hammer+To+Fall
    [shoutnumber] => 447900987066
)
```

Clearly, if left to the default means of processing, Request Manager will always deem the request to be "*Shouts Data: Array*". The Message Prefix should be used to indicate a search phrase which Request Manager can look for which will mark the start of the request text. In this instance the text:

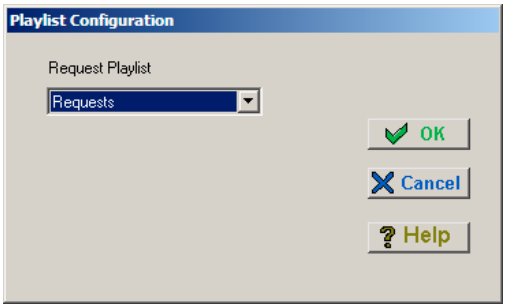
```
[shoutmessage] => ImageFM+
```

will suffice. The request itself will then be correctly parsed as 'Queen+Hammer+To+Fall'. The '+' symbols will be discarded and treated as spaces by the main Request Manager software.

Process even if filtering fails: Check this box if you want Request Manager to process the request even if it cannot find the specified *Message Prefix* in the mail. The default behaviour is to ignore the mail if the prefix cannot be found. This is useful if you have multiple sources of requests arriving into the inbox eg. SMS messages and 'plain' messages sent direct from listener's emails or by a suitable form on a web site.

Artist/Album Delimters: The use of these special characters will be discussed later on in this guide, for now leave them set to their default values.

Click OK to close the dialog, then configure the Out Module via the *Settings*, *Out-Module* menu.



Out-Module settings

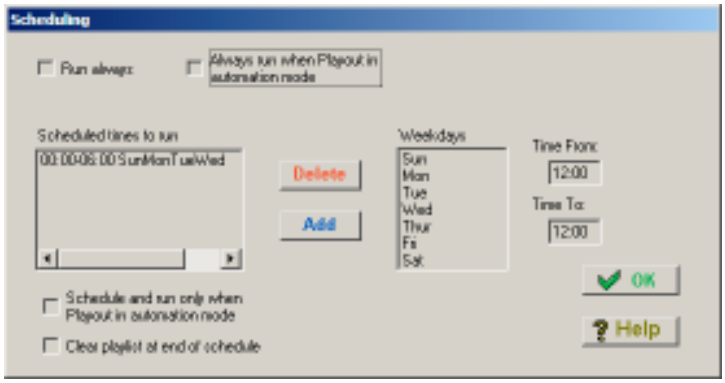
Here just simply pick the name of the playlist, which holds the listener requests – based on our examples it is just called "Requests".

How it works

It is necessary to understand a little of how Request Manager interprets the requests and attempts to match them to tracks in the Playout database because there are aspects of this which can be changed and depending on your station's music base *will* need changing.

Every 5 minutes, Request Manager will read the default inbox and search for unread messages. Any found will be read (and appear as 'read' in the email inbox). This happens regardless of whether you filter mail from a specific *From/To* address.

The conditions when Request Manager will run (ie. check for email) can be configured via the *Schedule* menu item.



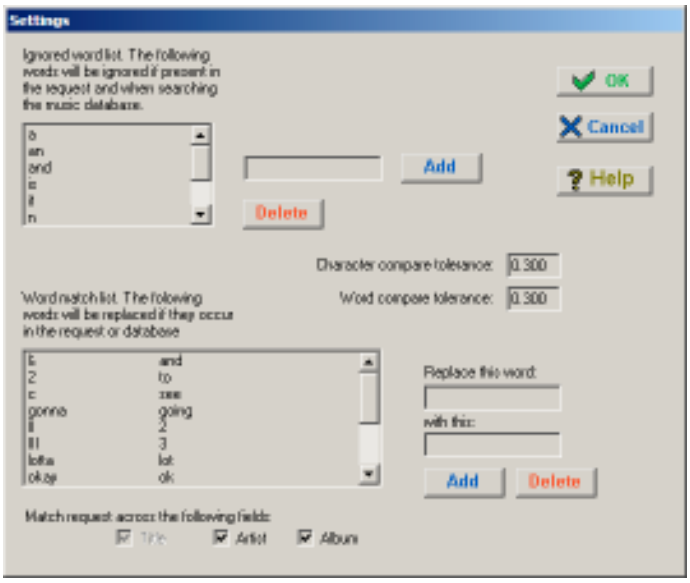
Scheduling options

Request Manager can be configured to check for new mail:

- At specific times/days of the week
- Whenever Playout is running in automation mode (only if Request Manager is running on the same machine as Playout)
- A combination of the above two options (ie. only if Playout is in automation AND is scheduled to run at the current date/time)
- All the time

The *Clear playlist at end of schedule* will automatically delete the contents of the defined Requests playlist once a period of accepting requests has ended eg. at the end of a scheduled period. This will ensure that the next time requests are being processed, any older ones are have been discarded and to prevent the playlist from increasing indefinitely in size.

The request is processed in accordance with a set of rules, some of which may be tailored via settings controlled in the *Settings*, *Main* menu item:



Processing rules

1. Firstly, all punctuation from the title is removed and everything converted to lower case.
2. The request text is broken into individual words.
3. Any words appearing in the 'Ignored word list' are deleted from the list of words. The list is case-insensitive ie. 'A' is the same as 'a'. This is intended to remove commonly occurring words such as 'a', 'the' 'is' etc. which listeners are liable to omit. This list may be amended to add or delete new words via the *Settings* menu.
4. Any words appearing in the left-hand column of the 'word match list' are converted to the equivalent those in the right hand column. This helps deal with slang, "text speak" and other common abbreviations. As before, new words may be added or deleted from the list.

Request Manager now begins searching the database for tracks, which meet these criteria.

An example may help at this point:

A listener requests the track:

"I feel like a wman" with the aim of hearing the track held in the database "Man, I feel like a woman" by the artist Shania Twain.

Rule	**Listener request**	**Database track**
	I feel like a wman	Man, I feel like a woman
Remove punctuation, make lower case	i feel like a wman	man i feel like a woman
Remove 'ignored words'	feel like wman	man feel like woman
Apply 'word matches' (no effect in this instance)	feel like wman	man feel like woman
Total Matched Words	**2**	**4**

The system now attempts to compare each word in the listener request with that in the database track. In terms of absolute matches, it can be seen that only 2 of the words ("feel" and "like") can be matched due to a spelling mistake (or text short-hand) in the request. However the manager software allows for a degree of mis-match due to spelling errors at this is controlled by the *Character compare tolerance* value in the settings. The default of 0.3 allows for a margin of error of a third when comparing words. So, given that only 1 character is wrong between "wman" and "woman" this is deemed acceptable (it could also be matched to "man" in this instance.

As a result, the system deems that 3 out of the 3 words (in the original request) can be matched.

During the compare process, the system keeps track of the best possible matches it encounters. If multiple tracks are detected which appear to have a good hit rate then a further set of rules are applied (including reducing spelling tolerances and taking into account word ordering) in an attempt to find the best match. The track with the best match after this process is then submitted as the request.

The *Word compare tolerance* value governs how many word matches in the request must occur in order for the track to be given consideration as a valid candidate for selection. Consider the above example, in this instance 100% of the words are deemed "matched". However if the track was not actually in the database you could still get matches but of a lower order – the track "Man Eater" by "Hall and Oats" would match 1 out of the 3 words but is clearly no where near the requested track. The default value for the Word compare tolerance specifies that at least one third of the words in the request must be matched to the track in the database in order to be considered a suitable candidate for selection. Therefore it is clear than "Man Eater" only gives exactly one third of a word match and therefore will not be included for selection. If no other tracks give a higher match then no request will be made.

The testing so far seems to indicate these figures give good results however it may be necessary to tweak them slightly if in practice they prove to be either too stringent or not stringent enough.

All this is logged in a text file within Request Manager so that should the software make the incorrect decision we can revise what it is doing to try and improve things.

Including artist (and album) information in the request

For mainstream western pop music, 95% of the time the title alone is probably good enough to distinguish the track in the database. However for some other music genres (Asian for example which utilises album information heavily to identify the show a song may have come from) and in that 5% of other instances it will be useful to include artist or album details in the request.

This can be achieved in two ways. Firstly, by including a unique *delimiter character* in the request itself which Request Manager will then use to extract artist and or album information. This character should be something which is unlikely to occur in a title/artist of a track – the defaults being the asterisk ('*') character for artist and hash ('#') for album. These are specified and can be altered in the *In Module* settings.

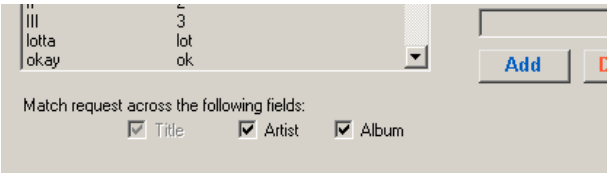
For example (given the delimiters above):

Hammer to fall	Just the title specified, will look for all tracks matching this title.
Hammer to fall * Queen	Title and artist specified
Hammer to fall # Greatest Hits	Title and album specified
Hammer to fall * Queen # Greatest Hits	Title, artist and album specified. Although in reality it is highly unlikely anyone would want to specify all 3.

When artist and/or album information is specified in this manner, the request is processed in the same manner as before however the rules – punctuation removal, word replacement etc. is also applied to the artist & album information and will return the closest matching title/artist combination.

This approach though is not particularly useful for text messaging, as it is unlikely the listener will bother to include the delimiter and there is plenty of scope for confusion arising when explaining it's purpose. Rather, the delimiter approach is more intended for handling requests based on a form on a station's web site. In this scenario, there would be distinct boxes for specifying artist/album information and the scripting software would then format up an email, inserting the delimiters automatically where required.

It is therefore possible to configure Request Manager, via the Settings menu to compare the words in the request string against the artist and/or album fields in the database –



The key issue to consider though is that the system has to make a best guess then as to which words form up the title part and which the artist (and/or album part) and can result in some very odd looking matches. For example consider a request for "Power of Love", in one of our databases this results in the following shortlist:

Title	**Artist**	**Album**
love me like that	Michelle Branch	Hotel Paper
The Power Of Love	Frankie Goes To Hollywood	The Best Christmas Album In The World...Ever! 2000 (CD1)
The Power Of Love	Lewis; Huey & The News	Power Cuts
Industrial Disease	Dire Straits	Love Over Gold
Love Over Gold	Dire Straits	Love Over Gold
Love, Reign O'er Me	Who; The	Quadrophenia (Disc 2 of 2)

The two instances of "Power of Love" are self-explanatory however the other candidates not so. It is only when you consider the spelling tolerances that in the case of the Michelle Branch track, the word "Paper" is only 2 letters different to "Power", also the words "the" and "of" are in the *ignored words list* that it is deemed a suitable candidate. However what is less obvious is that because the system cannot make any assumptions about which words in the request are the title and which the artist (in this case none of them) it ends up deeming seemingly unrelated tracks as candidates for the request.

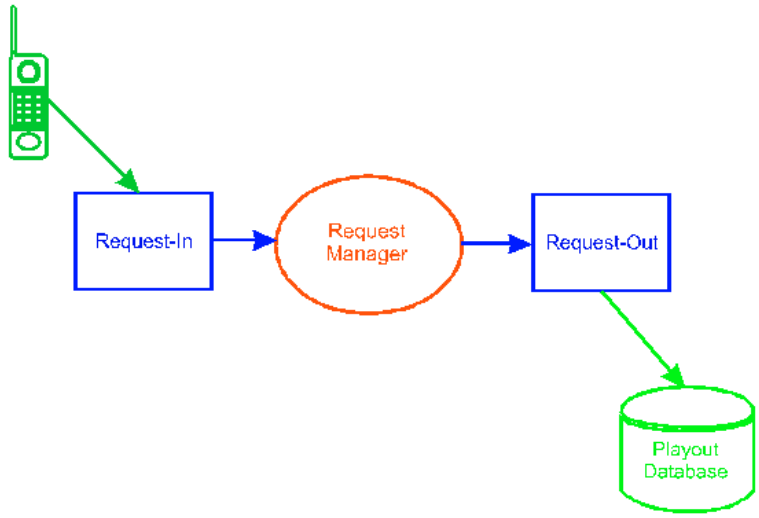
In this case, the additional rules which are applied to determine a closer match will pick one of the "Power of Love" tracks – which one is of course indeterminate based on their being no artist information supplied by the user but it will pick one of these as preference to the others on the list. This may not always happen though, just be aware that there may be some odd choices made depending on the wording in the request and the tracks available in your database.

The two approaches of including artist/album information in the search can be combined, in which case if no delimiters are found in the request email Request Manager will adopt the "best guess" approach of trying to find title/artist information across the whole request.

Request Manager Architecture

This final section is primarily directed to those in charge of deploying the system ie. the IT experts. In the last section we've concentrated on the concept of using Request Manager to obtain SMS based requests, which have been re-directed to an email inbox. However this may well not suite all environments for example the messages may be stored in a database or set of files on a server. There may also be scope for handling requests recorded via a normal phone line.

Request Manager is composed of a series of modules. It is designed to obtain the listener "inexact" request via the "Request-In" module, process it to determine the exact track which is required and (if a match can be found), put the track into the *Requests* playlist via the "Request-Out":



OAS Request Manager

Thus far, the "Request-In" module is the part responsible for extracting the requests from the email inbox and the "Request-Out" module for physically putting the required track into the database.

In order to facilitate the retrieval of requests from a different storage system, it is simply a case of switching in a different "Request-In" module. This could also accommodate handing requests via a normal phone call, although the reality at present is that whilst "voice recognition" technology has improved significantly in recent years it is still really out of the scope of most radio stations. As and when this changes though a suitable new module could be adapted for Request Manager.

Currently only one Request-In module exists, which performs the function described thus far – in obtaining the listener requests via an email inbox.

The rationale for generating additional Request-Out modules is less obvious – the default one does little else than put the desired request into the playlist. However Request Manager does currently ship with an additional module which allows the request to be submitted to a web server – set up as described earlier using the scripts on our website as a template:

http://www.onasticksoftware.co.uk/playout-interaction.html

It obviously bypasses the entire interactive search and confirm stages of the normal search process, instead issues a request to the server to directly add a specific request into the Playout system. The rationale behind this is that the sample scripts allow a refinement on allowing requests into the system, which is under your control. For example, they can prevent the same track from occurring more than once per session or restrict the same number of tracks from a given artist. This is subtly different from the rules available inside Playout which will not stop certain tracks from being played (although this is planned for a future release), merely delay them to ensure a consistent even spread. Ultimately,

submitting requests via this route allows you to fully control what tracks are included and which are rejected in a totally flexible manner.