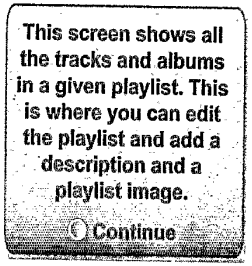
12: Playlist Items (No description) √

Playlist Items is the screen which shows all the tracks and albums in a playlist.



Help Popup

Play time, tracks and rating shown next to photo.

Playlist photo

Playlist items can be tracks or albums.

12. Playlist Items

Show "Title – Artist" and horizontally scroll when highlighted. Wait for XXXms (e.g. 1000 or 1 sec) before scroll begins, not immediate.

Features

- Include tracks or albums.
- Albums with tracks in order of album
- Clicking on an album goes into tracks in album, stop particular tracks from albums (see **Album Page**)
- Generic playlist image if none chose
- Changes the order on sort

Softkey

- Open (if album)
- Play
- Play Next – Add to end of Current Playlist
- Add to Playlist – Add to selected playlist
- Move* – move the selected item in list. See Edit Playlist page for desig
- Description* – enter/edit description. See Edit Playlist page for design
- Photo* – Modify the image associated with the playlist
- Sort* - see Sort Playlist popup
- Shuffle
- Rate - see Rate Track
- Now Playing – go to Play Track (if playing)
- Main menu
- Exit

* Only available on playlists which are owned/editable
This is a very long list – we might want to leave playlist level options (like Description and Photo to the Playlist Mak

Navigation	
◀◀	Back/Top Menu
◀	Back
▲▲	Up item
▲	Up item
●	Open (if album)
●●	Open (if album)
▼	Down item
▼▼	Down item
▶	Open (if album selected)
▶▶	Open (if album selected)

OMNIFONE™

12: Playlist Items (With description) √

This is the version of **Playlist Items** when there is a description.

Navigation	
◀◀	Back/Top Menu
◀	Back
▲	Up item
▲	Up item
●	Open (if album)
●●	Open (if album)
▼	Down item
▼	Down item
▶	Open (if album selected)
▶▶	Open (if album selected)

Playlist description

12. Playlist Items

Editable playlists

Only playlists which the user created in the first place can be edited using the
Remove
Move
Description
Photo
Sort options

Softkey

- Open (if album)
- Play
- Add to Playlist – Add to selected playlist
- Add to Current – Add to Current Playlist
- Remove – remov ethe selected item
- Move – move the selected item in list. See Edit Playlist page for desig
- Description – enter/edit description. See Edit Playlist page for design
- Photo – Modify the image associated with the playlist
- Sort - see Sort Playlist popup
- Shuffle
- Rate - see Rate Track
- Now Playing – go to Play Track (if playing)
- Main menu
- Exit

OMNIFONE™

12: Playlist Items (moving items within)

The **Playlist Edit** screen is used to edit playlists. Items can be discarded (removed) or items can be selected and then moved up or down using the joystick.

Playlist Description Popup

The popup is shown when Description is selected from the soft key menu

"Save" takes same action as O Joystick and saves edits.

12. Playlist Items (in edit order mode)

Features

- Select an item then move it up or down then drop.

When the user has selected the Move menu option then the highlighted item can be moved up and down in the list. This is achieved by using the joystick up and down. The highlighted item moves up and down in the list accordingly.

To exit this mode the user presses O Joystick

This mechanism is based on the Shortcut editor within Sony Ericsson phones which work in the same way to move items up and down in the short cut menu (typically acessed by pushing the joystick up on the home screen of the

Sort Playlist Popup

Popup shown twhen the user selects Sort soft key option

Navigation	
◀◀	Back/Top Menu
◀	Back
▲▲	Up item/move item up (if selected)
▲	Up item/move item up (if selected)
●	Drop item
●●	Drop item
▼	Down item/move item down (if selected)
▼▼	Down item/move item down (if selected)
▶	Open (if album selected)
▶▶	Open (if album selected)

OMNIFONE™

14: Add Photo

Add Photo allows the user to choose a photo off his phone memory and load it as the image to represent his playlist.

Need to decide what is easier: controlling camera or looking through file system. Accessing file system might give more options to the types of images the user could use.

14. Add Playlist Photo

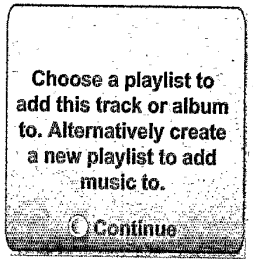
Attach Image To Playlist

- Browse photo images on their phone.
- Filter out files we can't handle
- Allow select, auto-size/crop
- Post (orig) image to the server with the playlist
- Some people prefer the two line artist and track layout – shall we look at see on device?

OMNIFONE™

15: Add to Playlist √

Add To Playlist is the screen shown if the user hits the "Add To Playlist" softkey whilst browsing their music or the available music on the network. The user is asked to select an existing playlist or to add a new one.



Help Popup

Maybe we should grey these out

See New Playlist Popup

15. Add to Playlist

Features

- If this is the first track to be added to a playlist then can we give them the option to start playing it when added?
- If they create a new playlist then have that as selected after creation.
- IF the track is not owned then we will need a popup along those at 80-83 which ask whether the user wants to buy but then go on to allow adding to playlist rather than asking about the play process.

Navigation	
◀◀	No function
◀	No function
▲▲	Up item
▲	Up item
●	Select
●●	Select
▼	Down item
▼▼	Down item
▶	No function
▶▶	No function

OMNIFONE™

20: Current Playlist

The **Current Playlist** is accessed from the **Playlists Menu**. It is the non-permanent playlist which is currently being played.

'Current Playlist' Discussion

The current default queue of music you line up for play.
Thought about calling it "Play Queue" but thought it had overly negative connotations? It does however tell a good story and sets expectations with the users. But most systems use Current Playlist so I'll stick with that for the moment.
Play Order?
Playing?

'Non-Permanent'

What does this mean? Should we keep it for them because it's useful and have a "Clear" option? Perhaps we should. In which case it'd be as permanent as any other playlist.

How everything is played via a playlist

Navigation	
◀◀	Back/Top menu
◀	Back
▲▲	(fast) Up item
▲	Up item
●	Open folder
●●	Open folder
▼	Down item
▼▼	(fast) Down item
▶	Open (if album)
▶▶	Open (if album)/Play next

Show "Track – Artist" (if sorted that way) and horizontally scroll when highlighted. Wait for XXXms (e.g. 1000 or 1 sec) before scroll begins, not immediate.

20. Current Playlist

Back to **Playlists**?

Some indication to show what's currently playing.

Albums included which includes all tracks by default in album order. Listed as Album name – Artist if sorted by Track. Otherwise Artist – Album if sorted by Artist.

Did originally think about allowing playlists inside of playlists. However decided against due to cyclical references and potential depth of edit screens & menu hierarchy.

Features

- You can manage this list whilst track is playing
- Select what will be the next track (when this one is finished)
- Are there actions which will implicitly clear the current playlist. Clicking on Play should clear and add to list eh?
- Some people prefer the two line artist and track layout – shall we look at see on device?
- Save this as playlist

Softkey

- Open (if album) – see Album Page
- Play next – popup?
- Sort by Artist/Track toggle – track – artist/artist - track
- Rate Track/Album – see Rate Track
- Shuffle/Orig Order – does this change the order on screen?
- Edit
- Empty
- Play/Pause (Current Playlist)
- Main Menu
- Exit

OMNIFONE™

22: Choose Genre

The **Choose Genre** screen is shown to new users before entry to a few different recommended tracks, artists, albums or playlists screens. As they have consumed no music so far so we have no genre signature for them. We therefore need to know something before choosing a recommendation set for this guy. This is done by inserting a **Choose Genre** screen so as to get a start point on this user's genre signature. The screen appears to be part of the normal app flow but after they have started to consume music this step won't be here because we have a more personalised genre signature for them.

Discussion
Do we tell them at this point what we are up to with asking them this? Does this ask more questions than it answers?

MusicStation is a personalised service which moulds to your tastes and provides recommendations. Choose a genre of interest now to start the process of learning your tastes.
Continue

22. Choose Genre

Features
- Quick up and down then joy in opens the **Recommended Playlists** screen.
- Should we allow multiple genres to be selected here?

Navigation	
◀◀	Back/Top menu
◀	Back
▲	Up item
▲	Up item
●	Choose genre
●●	Choose genre
▼	Down item
▼	Down item
▶	No action
▶▶	No action

25: Get New Playlist – Playlist Group (Top Playlists etc)

See previous slide for the definition of the contents of this screen.

Questions

- Top Playlists is probably the same list as the Playlists chart on the Buzz tab

25. Get New Playlist – Playlist Group

25. Get New Playlist – Playlist Group

25. Get New Playlist – Playlist Group

Navigation	
◀◀	Back/Top menu
◀	Back
▲▲	Up item
▲	Up item
●	Open playlist
●●	Open playlist
▼	Down item
▼▼	Down item
▶	Open playlist
▶▶	Open playlist

Softkey

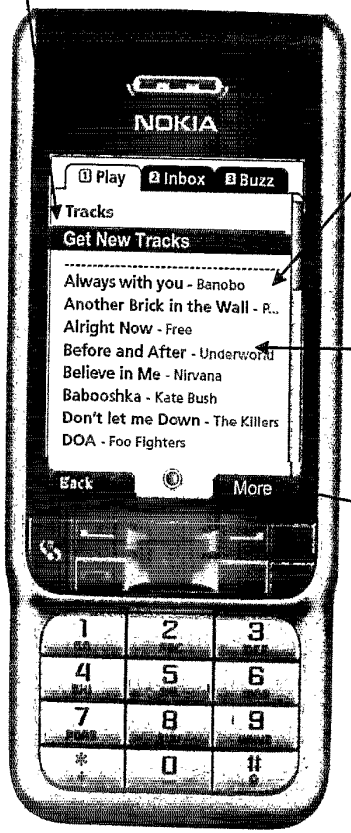
- Play
- Play next
- Open – see Playlist Items – perhaps no edit though?
- Rate – see Rate Track
- Now Playing – go to Play Track (if playing)
- Main menu
- Exit

OMNIFONE™

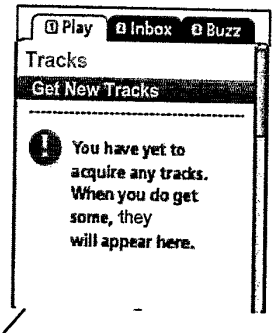
30/31/32: Tracks Main and Get New Tracks

This is the main menu of the tracks

Suggestion from MK to use "Discover Tracks"



30. Tracks Main



No tracks yet
When I do not yet own any tracks (pay per track)
or listened to any tracks (sub) then displays:

You have yet to acquire any tracks.
When you do get some, they will appear here.

You have yet to listen to any tracks.
When you do, they will appear here.

Get New Tracks

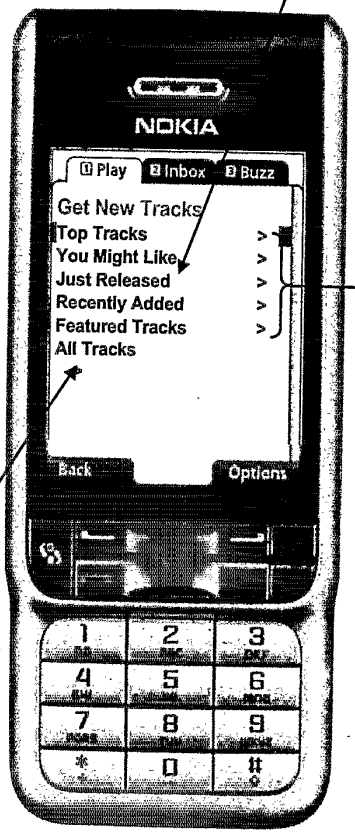
All my tracks are listed here for quick and easy access. Meaning of what my tracks is depends on whether pay-per-track or subscription account.

Softkey
- Play (playlist)
- Play Next
- Add to Playlist
- Rate
- Now Playing – go to Play Track (if playing)
- Play/Pause (Play Queue)
- Main Menu
- Exit

All Tracks goes to 60 Simple Search with the Tracks radio button pre-selected

Discussion Point
- Before Beta need a discussion about how/if we handle the differentiation between owned and downloaded tracks and how they should be displayed here. See email exchanges 1st week of April 06.

See over for details of the groups of tracks shown on the Get New Tracks screen



31. Get New Tracks

32. Get New Tracks – Track Grou
(contents depend on group selected, definitions of which are on next slide)

OMNITONE™

Track Groups shown on Get New Tracks

This is the set of track groups that are shown on the 30. Tracks Main screen. The exact details of the algorithms used for generating these lists are yet to be defined, but the general intention of each group is described.

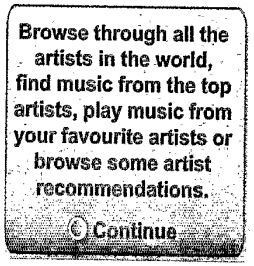
Group Name	Description	Filtered to exclude tracks the user has listened to (subscription) / own (pay-per-track)
Top Tracks	The top tracks across the whole system instance (I.e. an installation of MusicStation for a particular operator). It should be based on all tracks listened to and explicitly rated by all users. It is not personalised to the current user.	No
You Might Like	Tracks that we recommend to the user based on their (recent) listening habits, and taking into account any explicit ratings that they have given.	Yes
Just Released	The same as Top Tracks but only those with a release date within the last two weeks.	No
Recently Added	A list of (back catalogue) tracks which have been recently added to the system. Even those are new to the system they could potentially be old back catalogue releases. This list should be based on the user's (recent) listening habits in some way.	No
Featured Tracks	A list of tracks which have been editorially pushed for promotion. Personalised for the artists and genres the user listens to.	Yes
All Tracks	*This is not actually a group, but instead displays the 60. Simple Search screen with the Tracks radio button pre-selected*	N/A

Considerations

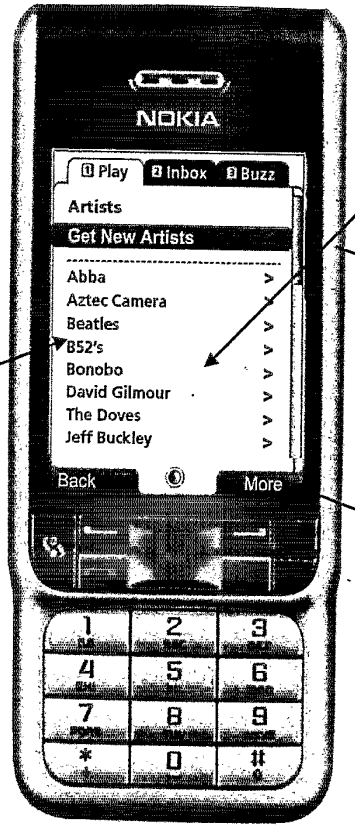
- Include a track chart for the week and an all-time track chart.
- Recently Added could be a hierarchy of artist/track etc
- Consider detailed rules for each of these lists.

40/41: Artists Main & Get new Artists√

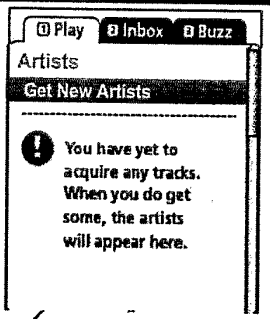
Artists Main is the main menu for looking for music by artist.



Help Popup



40. Artists Main



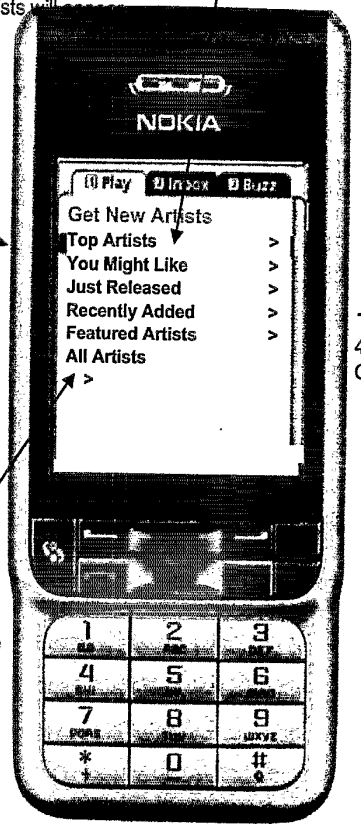
No tracks yet
When I do not yet own any tracks (pay per track)
or listened to any tracks (sub) then displays:

You have yet to acquire any tracks.
When you do get some, the artists will appear here.

You have yet to listen to any tracks.
When you do, the artists will appear here.

See over for details of each of these gr

Get New Artists



41. Get New Artists

42. Get New Artists – Artists Grou

All my artists are listed here for quick and easy access. Meaning of what my artists is depends on whether pay-per-track or subscription account. For subscription: When a user listens to a whole track from a new artist, the artist gets added to this list. The track also gets added to **My Tracks** and if the track is in an album, the album gets added to **My Albums**.

Softkey
- Open
- Now Playing – go to Play Track (if playing)
- Play/Pause (Current Playlist)
- Main menu
- Exit

All Artists goes to 60 Simple Search with the Artists radio button pre-selected

Navigation	
	Back/Top menu
	Back
	Up item
	Up item
	Open folder
	Open folder
	Down item
	Down item
	Open folder
	Open folder

Discussion Point
- Before Beta need a discussion about how/if we handle the differentiation between owned and downloaded artists' tracks and how they should be displayed here. See email exchanges 1st week of April 06.

OMNIFONE™

Artist Groups shown on Get New Artists

This is the set of artist groups that are shown on the 41. Get New Artists screen. The exact details of the algorithms used for generating these lists are yet to be defined, but the general intention of each group is described.

Group Name	Description	Filtered to exclude artists the user has listened to (subscription) / own (pay-per-track)
Top Artists	The top artists across the whole system instance (I.e. an installation of MusicStation for a particular operator). It should be based on all tracks listened to and explicitly rated by all users. It is not personalised to the current user.	No
You Might Like	Artists that we recommend to the user based on their (recent) listening habits, and taking into account any explicit ratings that they have given.	Yes
Just Released	The same as Top Artists but only those with a track or album release date within the last two weeks. It's the artists of the items in the Just Released Tracks and Just Released Albums groups (see Get New Tracks and Get New Albums screens)	No
Recently Added	A list of artists which have been recently added to the system for the first time. This is artists for whom we have added content, where we had *no* content from that artist before.	No
Featured Artists	A list of artists which have been editorially pushed for promotion. Personalised for the artists and genres the user listens to.	Yes
All Albums	*This is not actually a group, but instead displays the 60. Simple Search screen with the Artists radio button pre-selected*	N/A

Considerations

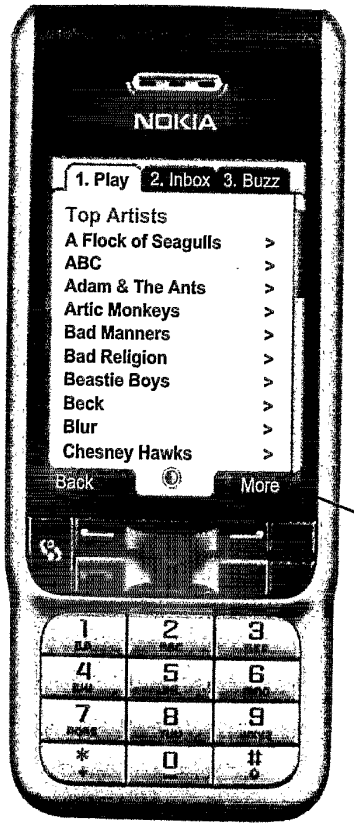
- Include a artist chart for the week and an all-time artist chart.
- Recently Added could be a hierarchy of artist/album etc
- Consider detailed rules for each of these lists.
- Need to define logic we use in classifying an artist as listened to/owned for purposes of filtering out from these lists.

OMNIFONE™

42: Get New Artists – Artists Group (e.g. Top Artists √)

This screen displays the contents of an Artists Group as selected from 41. Get New Artists. The contents of this screen depends on the group selected and is defined by the rules on the previous slide.

Features
- Blah



42. Get New Artists – Artists Group

Softkey
- Open
- Now Playing – go to Play Track (if playing)
- Play/Pause (Current Playlist)
- Main menu
- Exit

Navigation	
◀◀	Back/Top menu
◀	Back
▲	Up item
▲	Up item
●	Open artist
●	Open artist
▼	Down item
▼	Down item
▶	Open artist
▶▶	Open artist

OMNIFONE™

45: Artist Homepage √

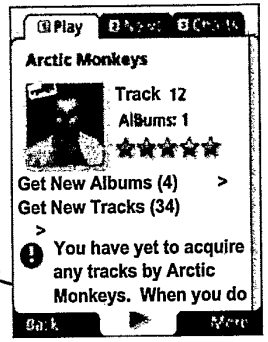
The **Artists Homepage** is the screen which collates all albums and tracks of this artist.

Help Popup

You have yet to acquire any tracks by Arctic Monkeys. When you do get some, they will appear here.

Or

You have yet to listen to any tracks by Arctic Monkeys. When you do, they will appear here.



No tracks owned

Either artists level image or cover from top selling album.

Should have a chevron to indicate open.

45. Artists Homepage

Playtime and count of the tracks which the user owns (ppt) or has listened too (sub)

Browse further tracks and albums by this artist. **Counts** displayed to right in parentheses are total number of albums/tracks available within the option.
Get New Albums goes to **46. Artist Albums**
Get New Tracks goes to **47 Artist Tracks**

All tracks by this artist which I own/downloaded (pay-per-track) or have listened to (subscription)

Features

- Maybe put the record label in there. But perhaps this could vary by track.
- Shows total album and tracks numbers
- Shows a rating which would be nice to be able to highlight and change??
- Artist biog – same as artist profile?

Softkey

- Open (if Get... option is highlighted)
- Play (if Track is highlighted)
- Add to Playlist (if album or track selected)
- Rate Artist
- Similar Artists (or should it be artists you might like?)
- Artist News
- Artist Profile
- Now Playing – go to Play Track (if playing)
- Main Menu
- Exit

Navigation©	
◀◀	Back/Top menu
◀	Back
▲▲	(Fast) Up item
▲	Up item
●	Open album (if selected)/Play?
●●	Open album (if selected)/Play?
▼	Down item
▼▼	(Fast) Down item
▶	Open album (if selected)
▶▶	Open album (if selected)/Play?

Discussion Point

- Before Beta need a discussion about how/if we handle the differentiation between owned and downloaded artists' tracks and how they should be displayed here. See email exchanges 1st week of April 06.

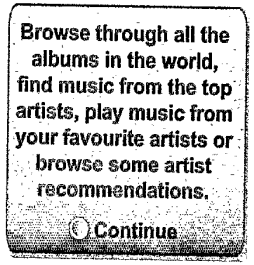
Discussion Point

- Consider including "similar aritsts" links in here – I.e. recommendations seeded from what you are currently looking at.

OMNIFONE™

46: Artist Albums√

Artist Albums lists all the albums for a particular artist.



Help Popup

46. Artist Albums

All of the artists albums are listed here in alphabetical order initially

Open an album goes to 52 Album Page

Softkey
- Open
- Now Playing – go to Play Track (if playing)
- Play/Pause (Current Playlist)
- Main menu
- Exit

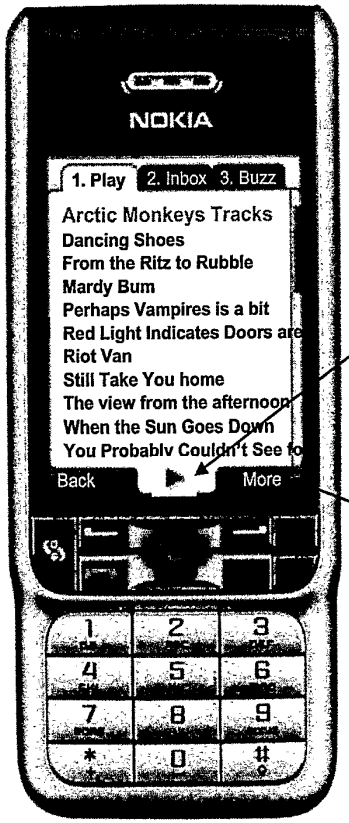
Navigation©	
⏮	Back/Top menu
◀	Back
⏫	Up item
▲	Up item
⏺	Open folder
⏺⏺	Open folder
▼	Down item
⏬	Down item
▶	Open folder
⏭	Open folder

47: Artist Tracks√

Artist Tracks lists all the tracks belonging to a particular artist.

Help Popup

All of the artists tracks are listed here in alpha order originally



47. Artists Tracks

Note that since this is a track listing the Play icon is shown

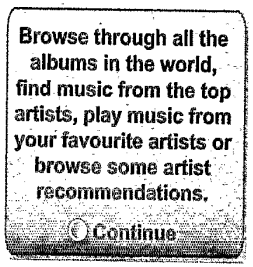
Softkey
- Play
- Now Playing – go to Play Track (if playing)
- Play/Pause (Current Playlist)
- Main menu
- Exit

Navigation©	
◀◀	Back/Top menu
◀	Back
▲	Up item
▲	Up item
●	Open folder
●●	Open folder
▼	Down item
▼	Down item
▶	Open folder
▶▶	Open folder

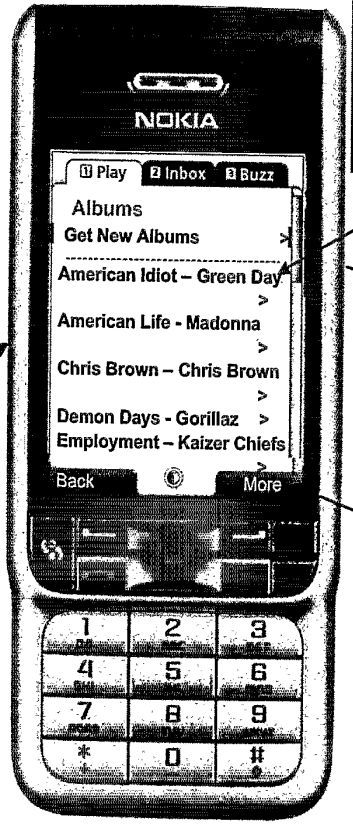
OMNIFONE™

50/51: Album Main & Get New Albums√

Album Main is the main menu for looking for music by album.



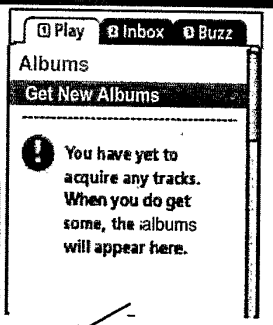
Help Popup



50. Album Main

All my albums are listed here for quick and easy access. Meaning of what my artists is depends on whether pay-per-track or subscription account.

Note that we display the artist name for clarity.

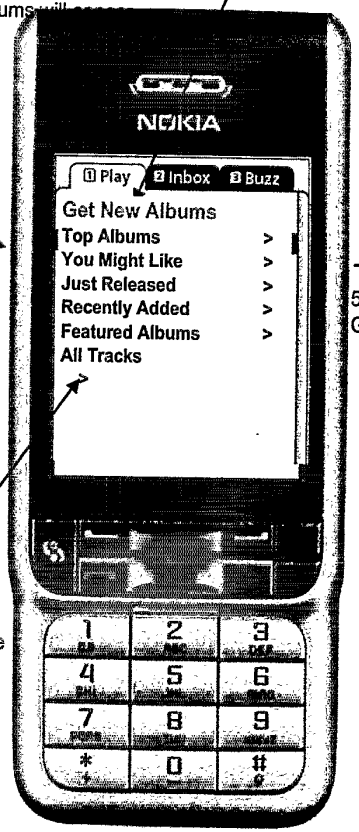


No tracks yet
When I do not yet own any tracks (pay per track)
or listened to any tracks (sub) then displays:

You have yet to acquire any tracks.
When you do get some, the albums will appear here.

You have yet to listen to any tracks.
When you do, the albums will appear here.

Get New Albums



51. Get New Albums

See over for details of these g

52. Get New Album: – Albums Gro

Softkey
- Open folder
- Now Playing – go to Play Track (if playing)
- Play/Pause (Current Playlist)
- Main menu
- Exit

All Albums goes to 60 Simple Search with the Albums radio button pre-selected

Navigation	
	Back/Top menu
	Back
	Up item
	Up item
	Open folder
	Open folder
	Down item
	Down item
	Open folder
	Open folder

Discussion Point
- Before Beta need a discussion about how/if we handle the differentiation between owned and downloaded albums and how they should be displayed here. See email exchanges 1st week of April 06.

OMNIFONE™

Album Groups shown on Get New Albums

This is the set of album groups that are shown on the 51. Get New Albums screen. The exact details of the algorithms used for generating these lists are yet to be defined, but the general intention of each group is described.

Group Name	Description	Filtered to exclude albums the user has listened to (subscription) / own (pay-per-track)
Top Albums	The top albums across the whole system instance (I.e. an installation of MusicStation for a particular operator). It should be based on all tracks listened to and explicitly rated by all users. It is not personalised to the current user.	No
You Might Like	Albums that we recommend to the user based on their (recent) listening habits, and taking into account any explicit ratings that they have given.	Yes
Just Released	The same as Top Albums but only those with a release date within the last two weeks.	No
Recently Added	A list of (back catalogue) albums which have been recently added to the system. Even those are new to the system they could potentially be old back catalogue releases. This list should be based on the user's (recent) listening habits in some way.	No
Featured Albums	A list of albums which have been editorially pushed for promotion. Personalised for the artists and genres the user listens to.	Yes
All Albums	*This is not actually a group, but instead displays the 60. Simple Search screen with the Albums radio button pre-selected*	N/A

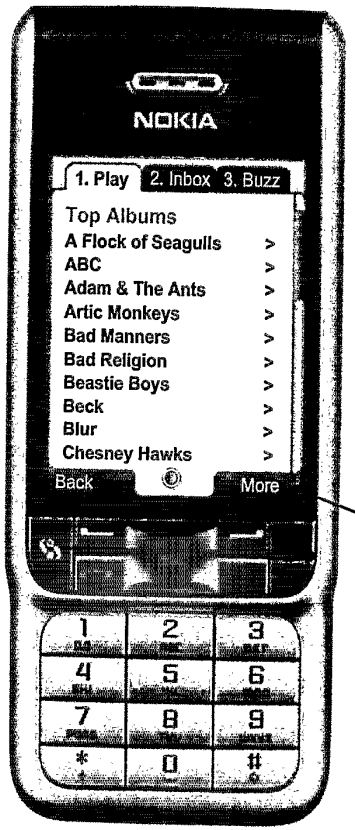
Considerations

- Include a album chart for the week and an all-time album chart.
- Recently Added could be a hierarchy of artist/album etc
- Consider detailed rules for each of these lists.
- Need to define logic we use in classifying an album as listened to/owned for purposes of filtering out from these lists.

52: Get New Albums– Albums Group (e.g. Top Albums

This screen shows the contents of the selected Album Group like Top Albums. The definition of the contents of this screen is given on the previous slide.

Features
- Blah



42. Get New Albums – Albums Group

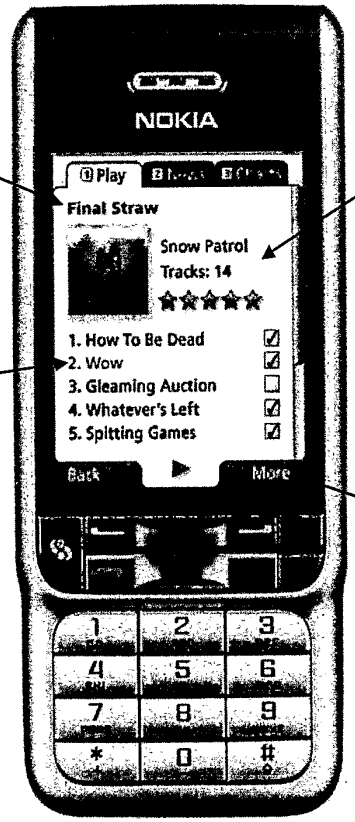
Softkey
- Open
- Now Playing – go to Play Track (if playing)
- Play/Pause (Current Playlist)
- Main menu
- Exit

Navigation	
◀◀	Back/Top menu
◀	Back
▲▲	Up item
▲	Up item
●	Open artist
∞	Open artist
▼	Down item
▼▼	Down item
▶	Open artist
▶▶	Open artist

55: Album Page √

The **Album Page** is the screen which collates all the tracks in an album. Users can deselect tracks from an album so that they don't play and aren't included in playlists.

Help Popup



Album name.

This is artist name and the number of tracks the system has for this album – not controlled by how many the user has.

This is an idea to show which track is currently playing.

Features
- Include other information like release date or reviews?

Album Tracks included:
- All albums start off with all tracks included.
- User can uncheck tracks as they like.
- Wherever used that album only plays the checked tracks.

Softkey
- Play
- Add to Playlist
- Rate Track
- Similar Artists (or should it be artists you might like?)
- Artist News
- Artist Profile
- Now Playing – go to Play Track (if playing)
- Main Menu
- Exit

52. Album Page

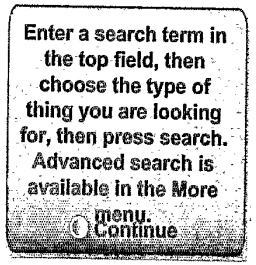
Navigation	
◀◀	Back/Top menu
◀	Back
▲▲	Up item
▲	Up item
●	Check/Uncheck
●●	Check/Uncheck
▼	Down item
▼▼	Down item
▶	Check/Uncheck
▶▶	Check/Uncheck

Discussion Point
- Consider including "similar albums" links in here – I.e. recommendations seeded from what you are currently looking at.

OMNIFONE™

60: Simple Search √

The **Simple Search** screen provides a text based search with control over what it is your are looking for (which defaults to All).



Help Popup

We could advertise **Advanced Search** here at the bottom of the page with a button say.

60. Simple Search

Features

- Can come back to this screen from the **Search Results** screen using **Search Again** softkey.
- Screen has a memory so retains the state of search. There is a rest softkey.
- Predictive T9 text or not?
- What case/numeric/punctuation in SLE?
- Different punctuation char lists on different keys in different phones?
- In the demo for performance reasons we limit the user to entering at least three characters on a track search to prevent them from returning huge numbers of tracks.

Softkey

- Search (fires search)
- Advanced Search (goes to **Advanced Search**)
- Reset Search (resets to null & defaults)
- Now Playing – go to Play Track (if playing)
- Play/Pause (Current Playlist)
- Main Menu
- Exit

Navigation ◎	
◀◀	Back/Top menu
◀	Back/Left in SLE
▲▲	Up item
▲	Up item
●	When in text box: Search; When in radio buttons: select option
●●	Select/Search?
▼	Down item
▼▼	Down item
▶	Right in SLE
▶▶	Right in SLE/Search?

OMNIfONE™

61: Advanced Search √

Search For What?
- All
- Artist
- Track
- Album
- Playlists – should allow this

Minimum Rating
- None
- Great
- Good
- Average
- Poor
- Bad

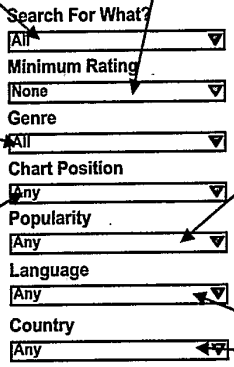
Minimum Rating

Should these be gold? Think not actually as one gold star looks like an award.

Does this not show that a 5 star based system is much easier on the eye and brain.

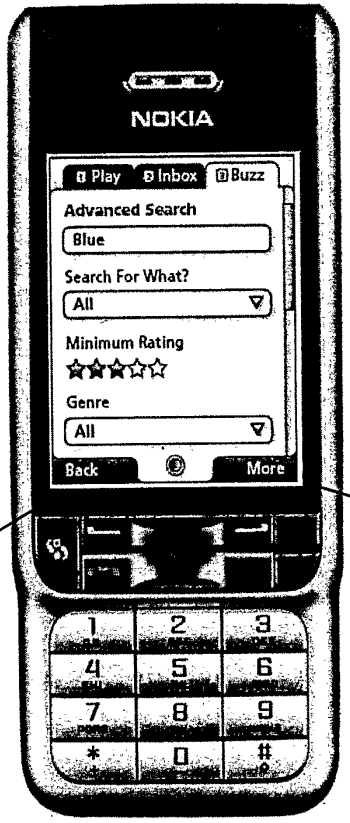
Genre
- All
- Genre1
- Genre2
- Genre3
- ...
- ...

Chart Position
- Any
- Number Ones
- Top Ten
- Top 40
- In charts
- Not in charts
- Somebody??



Popularity
- Any
- Popular?
- Classic?
- Not Popular?
- Can someone expand on what options & how this is achieved with what meta-data.

As per library meta-data.

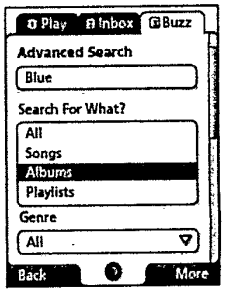


61. Advanced Search

Fires off search to the server, see **Search Results**.

Softkey
- Search (fires search)
- Simple Search (goes to **Simple Search**)
- Reset Search (resets to null & defaults)
- Now Playing – go to Play Track (if playing)
- Play/Pause (Play Queue)
- Main Menu
- Exit

Features
- Can come back to this screen from the **Search Results** screen using **Search Again** softkey.
- Once switched you stay on this type of search when you refine.
- SHOULD WE ADD IN HERE DECADE?



Drop-down list box expanded.

Navigation	
◀◀	Start of SLE (if keyword input)
◀	Back cursor/Select option (I.e. rating)
▲	Up to select keyword active
▲	Select control (e.g. keyword, rating, etc)
●	Search or DDLB/option activate
●●	Search/No function
▼	Select control (e.g. keyword, rating, etc)
▼	Down to select Popularity
▶	Back cursor/Select option (I.e. rating)
▶▶	End of SLE (if keyword input)

OMNIFONE™

62: No Search Results √

When the search returns no results this screen is shown

62. No Search Results

63: Mixed Results √

The **Track Results** screen is displayed when the user has chosen to have Tracks returned in their search.

Help Popup

Search Results
- If under the limit then just the number
- Else [150 of 375]

Do a **Most Popular** section at the top? Would need us to keeping a rolling popularity with a track so that we don't have to calculate on the fly.

General Results is the main search results. Repeat **Most Popular** in here too – guess so.

63. Mixed Results

Navigation	
◀◀	Start of results list
◀	Up one screen of results
▲▲	(Fast) up item highlight
▲	Up item highlight
●	Add to Play Queue
●●	Add to Playlist??
▼	Down item highlight
▼▼	(Fast) down item highlight
▶	Down one screen of results
▶▶	End of results list

When there are more than N (say 50) items in a list, the up/down holds become fast list navigation. When less than N the holds are standard speed. The holds should become fast after a second or two of normal speed.

Server Search Considerations
- Have to have stop words like "the" – oracle text blade stuff?
- Concatenation fields for cross object search.
- Café = cafe?

Large Results Handling Discussion

We need to decide on a limit to the amount of results we will show in one page. Bear in mind that if we choose 150 then there will be a cleverer thing going on in the background. The comms thread will post back the first part of the 150 results (say 50) and allow the GUI to display them. When the next 50 are down then they are seamlessly added to. Comms & GUI need to shared an understanding of what data's ready to use when.

The keys could behave differently with different amount of results. For example the held down joy might only go fast if there are more than 40 results in the list it's navigating.

Discussion
- Should we mix artists, tracks, albums and playlists in the one list? Or separate?
- There has been some discussion of using Next 50/Previous 50 to break the screen down into "manageable" chunks, but since we can now dynamically add to a screen as the user usees it, there's no real performance advnatage in not displaying all results in the same screen.
- If we do switch to a Next 50/Prev 50 approach then one question to ask is: When the results are split by tracks on phone / tracks not on phone. If the user owns over 50 of tracks they have searched for, then the first screen should just have user results. The second screen would then have some more user owned tracks, followed by a divider and then tracks not on the phone.

Softkey
- Play
- Find (overleaf)
- Artist Homepage
- Sort by Artist/Track toggle
- Search Again (back to simple/advanced search)
- Advanced Search (if in a result list from a simple search)
- Now Playing – go to Play Track (if playing)
- Main Menu
- Exit

OMNIFONE™

64: Artist Results √

The **Artist Results** screen is displayed when the user has chosen to have Artists returned in their search.

Help Popup

Search Results
- If under the limit then just the number
- Else [150 of 375]

Do a **Most Popular** section at the top? Would need us to keeping a rolling popularity with an artist so that we don't have to calculate on the fly.

General Results is the main search results sorted alphabetically. Repeat **Most Popular** in here too – guess so.

64. Artist Results

Handling large result sets

When reach maximum number of results that we want to display in a screen (e.g. 100) then display at bottom of the list:

:
Result 98
Result 99
Next 100 >

On the subsequent screen display

Results [100-200 of 375]
Previous 100 >
Result 101
Result 102
Result 103
:
Next 100 >

Softkey
- Open folder – Artist Homepage
- Find (overleaf)
- Search Again (back to simple/advanced search)
- Advanced Search (if in a result list from a simple search)
- Now Playing – go to Play Track (if playing)
- Main Menu
- Exit

Navigation	
⏮	Start of results list
◀	Up one screen of results
▲	(Fast) up item highlight
▲	Up item highlight
●	Open artist homepage
●●	Open artist homepage/Add to Play Queue??
▼	Down item highlight
▼	(Fast) down item highlight
▶	Down one screen of results/Open artist
⏭	End of results list

OMNIFONE™

65: Track Results √

The **Track Results** screen is displayed when the user has chosen to have Tracks returned in their search.

Help Popup

Do a **Most Popular** section at the top? Would need us to keeping a rolling popularity with a track so that we don't have to calculate on the fly.

General Results is the main search results. Repeat **Most Popular** in here too – guess so.

65. Track Results

Features

- Do we need a Track Detail screen beneath this where there's more info about the track.
- Maybe we shouldn't do sort at this stage?

Softkey

- Play
- Add to Playlist
- Find (overleaf)
- Artist Homepage
- Sort by Artist/Track toggle??
- Search Again (back to simple/advanced search)
- Advanced Search (if in a result list from a simple search)
- Now Playing – go to Play Track (if playing)
- Main Menu
- Exit

Navigation	
◀◀	Start of results list
◀	Up one screen of results
▲▲	(Fast) up item highlight
▲	Up item highlight
●	Play?
●●	Add to Playlist??
▼	Down item highlight
▼▼	(Fast) down item highlight
▶	Down one screen of results
▶▶	End of results list

66: Album Results √

The **Album Results** screen is displayed when the user has chosen to have Albums returned in their search.

Help Popup

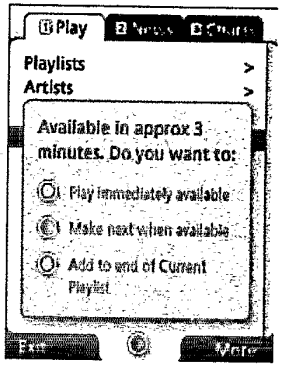
Do a **Most Popular** section at the top? Would need us to keeping a rolling popularity with an album so that we don't have to calculate on the fly.

General Results is the main search results. Repeat **Most Popular** in here too – guess so.

66. Album Results

Features

- Maybe we shouldn't do sort at this stage?



Softkey

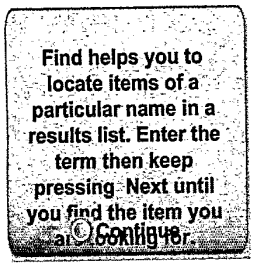
- Play
- Add to Playlist
- Open Album – Album Page
- Find (overleaf)
- Artist Homepage
- Sort by Artist/Track toggle??
- Search Again (back to simple/advanced search)
- Advanced Search (if in a result list from a simple search)
- Now Playing – go to Play Track (if playing)
- Main Menu
- Exit

Navigation	
◀◀	Start of results list
◀	Up one screen of results
▲▲	(Fast) up item highlight
▲	Up item highlight
●	Open Album
●●	Add to Playlist??
▼	Down item highlight
▼▼	(Fast) down item highlight
▶	Down one screen of results
▶▶	End of results list

OMNIFONE™

67: Find Results √

The **Find** function is a popup off the **Search Results** softkey menu. It allows the user to quickly move through results lists.



Help Popup

Move to next match

Close the **Find** popup

67. Find Results

Features

- When the find popup opens it defaults to the active search term.
- If it's a long list and next find is way down the results data on the server then it skips to the appropriate page of data from the server. Needs server to be working in handshake with the client.
- Could be that all matches on screen are coloured as shown.
- The currently highlighted matching term is a different colour too.

Navigation	
◀◀	Start of SLE
◀	Left cursor SLE
▲▲	(Fast) up item highlight
▲	Up item highlight
●	Find next
●●	Find next
▼	Down item highlight
▼▼	(Fast) down item highlight
▶	Right cursor SLE
▶▶	End of SLE

OMNIFONE™

68: Downloaded: Main

The **Downloaded** screen shows a list of all tracks on the phone available for immediate play irrespective of network connection.

What's On The Phone?

Someone suggested that we show what's immediately playable, what's downloading and what's still in the network. The suggestion was to use colours or icons.
It's possible this is correct. However I am nervous to invent a new GUI widget just yet as I'm not sure whether the information is necessary.

Features

- Downloaded icon
- Active based on connection

70: Play Track √

This screen controls and shows the current track which is playing. If a track is playing then this screen appears after XX seconds of inactivity in other screens. The full controls required for the playing process are provided by the joystick alone for a faster slicker experience.

Next Next Next - Fast Track Navigation

Tracks from the current playlist/album can be navigated very quickly using the joy left and joy right controls. However in order to facilitate a key usability requirement the names of the tracks, artist etc must be displayed near instantly.

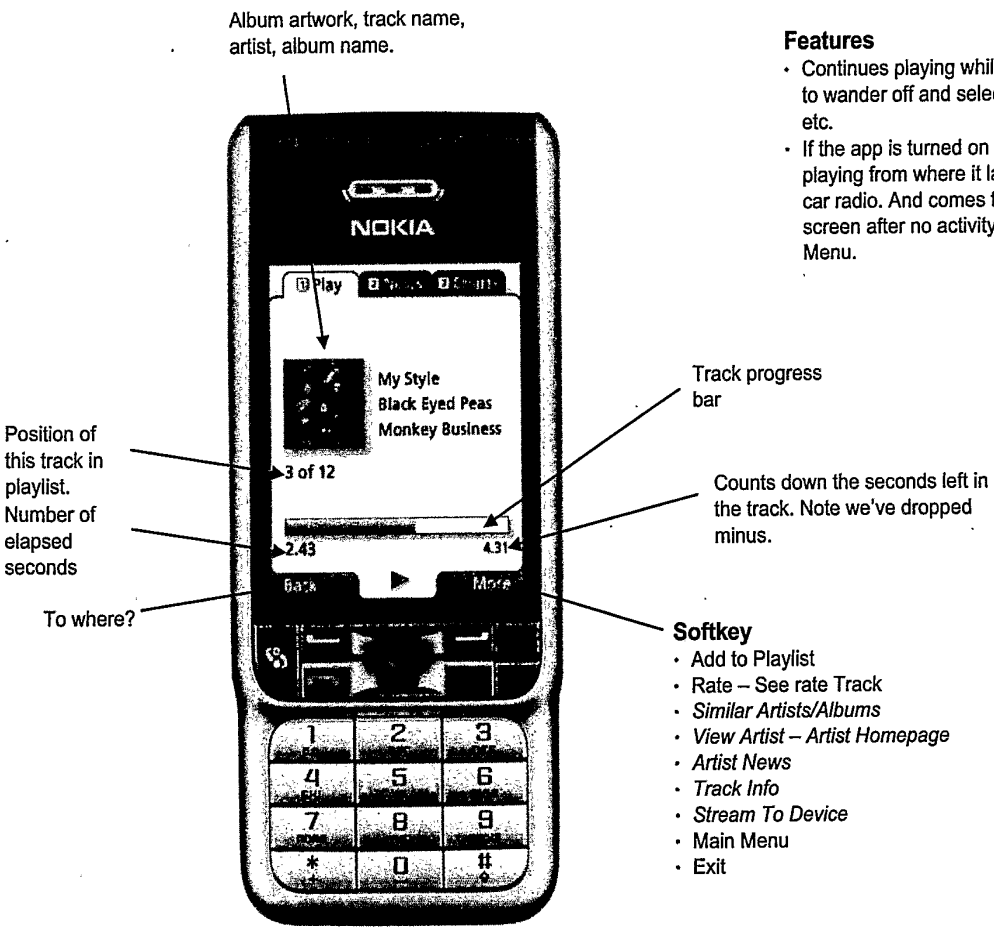
User's use this function to scan through music lists and will not like having to wait a second or more before the next track is shown. Album art, track times, animation of the titles etc are not necessary for this quick navigation.

Tab Context of Now Playing Screen

The 70. Now Playing screen is treated a little specially in that it can display on any tab, but it will not be considered as having context on that tab.

If you are on Tab 3 and playing a track then Now Playing screen will appear as if on tab 3. When you press [1] on this Now Playing screen, The last screen that you were in in tab 1 (other than Now Playing) should be the screen you go back to.

Player Contr▶	
◀◀	Rewind track
◀	Previous track (if less than 5 secs into track)
▲▲	Start of track (if more than 5 secs into track)
▲	Volume up
●	Play/Pause
●●	Play/Pause
▼	Volume down
▼▼	Volume down
▶	Next track
▶▶	Fast forward



70. Play Track

Features

- Continues playing whilst allowing the user to wander off and select their next playlist etc.
- If the app is turned on then it continues playing from where it last left off – just like a car radio. And comes to this screen? Or this screen after no activity for X secs on Main Menu.

Discussion Point

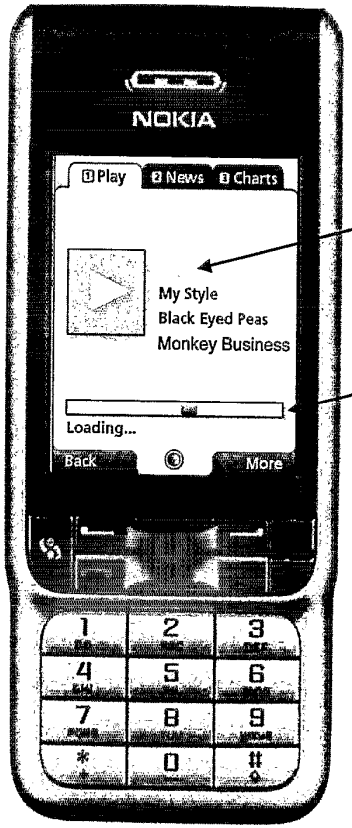
- Before Beta need a discussion whether we provide suggestions seeded from the current track on this screen. Chris E believes this is very important (it's a strong feature of Napster for example). Chris E has suggested one design.

OMNIFONE™

70: Play Track (Loading) √

Sometimes due to phone limitations there will be a brief delay between a user action which is meant to result in a track starting to play and that track beginning to play. When this happens the user needs feedback that the application and phone have not locked up.

Volume Cont ▶I	
◀◀	Silent
◀	Volume down
▲▲	Volume up
▲	Volume up
●	Back
●●	Back
▼	Volume down
▼▼	Volume down
▶	Volume up
▶▶	Full volume



70. Play Track (Loading)

Artist and title displayed immediately, but artwork and track timings not shown until got track playing

Moves back and forth while loading.

<u>Alternatives which require less Processing Power</u>

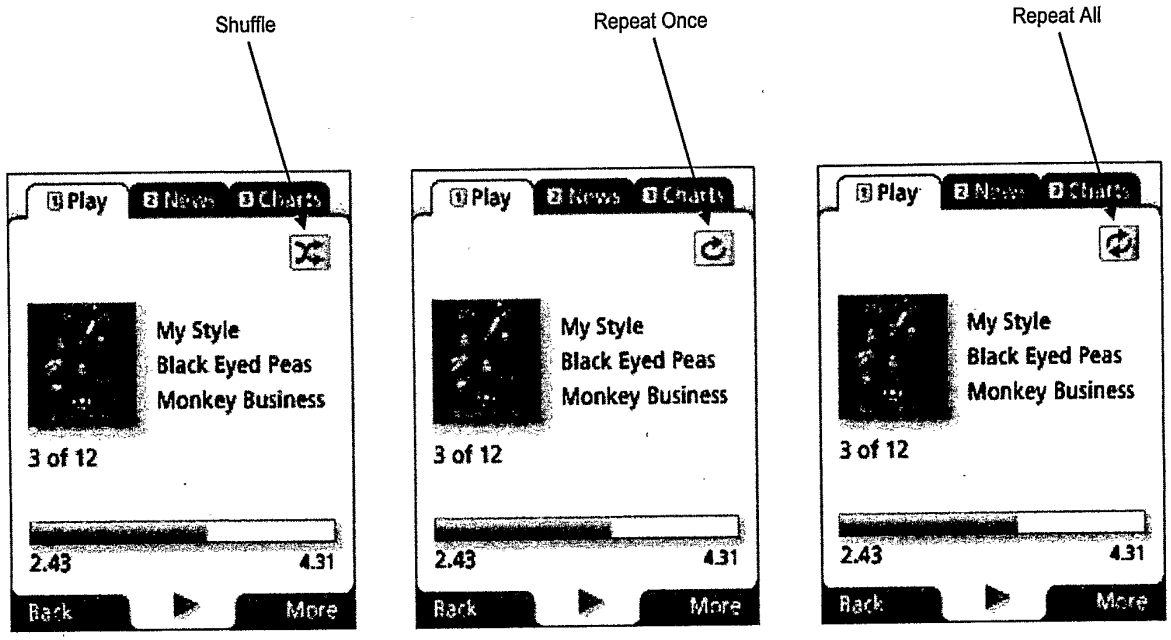
We found that displaying this bouncing loading bar had severe effects on the music file loading speed on some handsets so other less intensive mechanisms to consider are:

- Use the bar which exists as the loading progress filling up to 100%. This is only an incremental write to the screen and therefore should use less of the phone's resources.
- Otherwise animating the dots on the "Loading..." text

OMNIfONE™

70: Play Track (Repeat/Shuffle) √

This slide shows the **Play Track** screen with the icons showing all repeat and shuffle options.

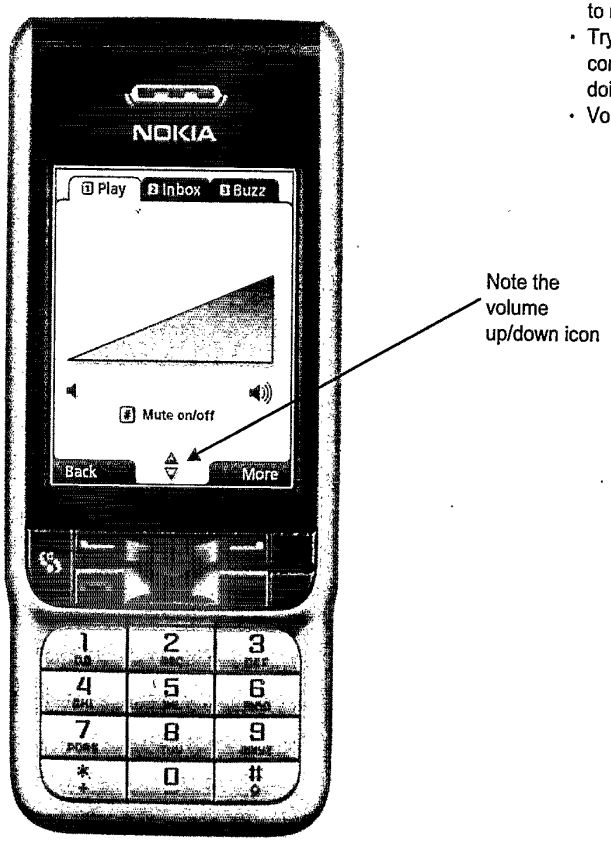


72: Play Track (Volume) √

When the joy up or down is used to change the volume the **Play Track** screen changes to show the current volume level.

Features

- After X secs of key/joy inactivity snaps back to normal **Play Track** screen.
- Try to detect the phones physical volume control and show this just as if the joy was doing it.
- Volume must react quickly for usability.

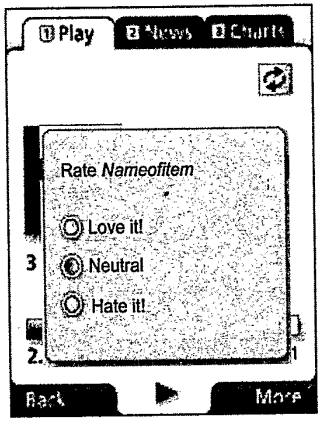


70. Play Track (Volume)

Volume Cont▶l	
◀◀	Silent
◀	Volume down
▲▲	Volume up
▲	Volume up
●	Back
●●	Back
▼	Volume down
▼▼	Volume down
▶	Volume up
▶▶	Full volume

74: Rate Track / Album / Artist / Playlist

This popup allows a user to rate a track/album/artist/playlist based on a three way choice. It is displayed whenever the user selects Rate from the soft key menu. Initial value is Neutral.

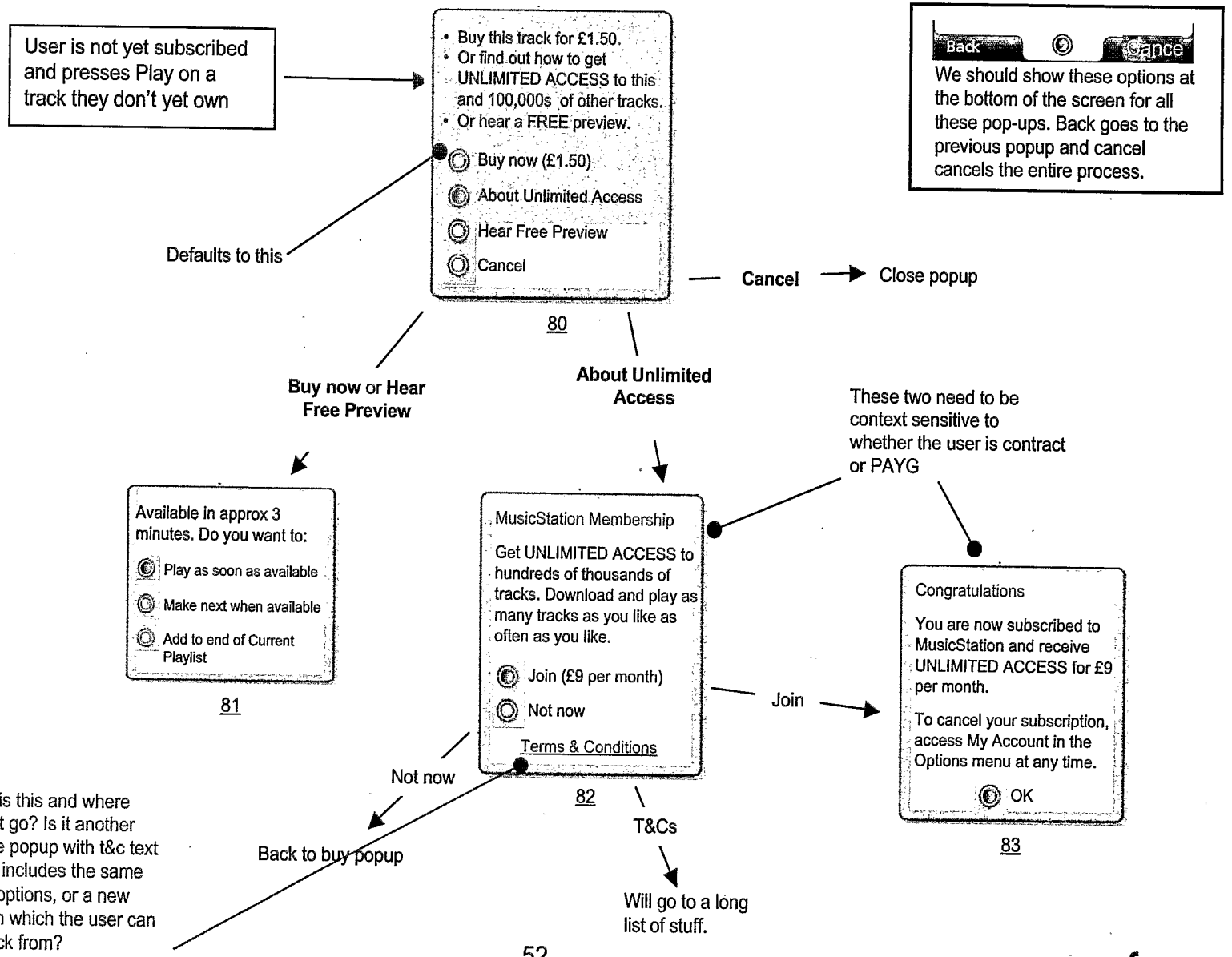


Discussion

- Should we use Like & Dislike + not set?
- Love it/hate it
- Rate artist, album, track.
- Scrap fantastic & dreadful
- Needs cancel with no default
- Colour sync on the button
- Need to be able to see rating as well as rate
- Disclosed only after you have rated?
- Is the rating the group rating or the nearest neighbour only as this will bring down to an average on most stuff eh?
- Go to stars?

80-83: Buy Track Process (Play or Add to Playlist)

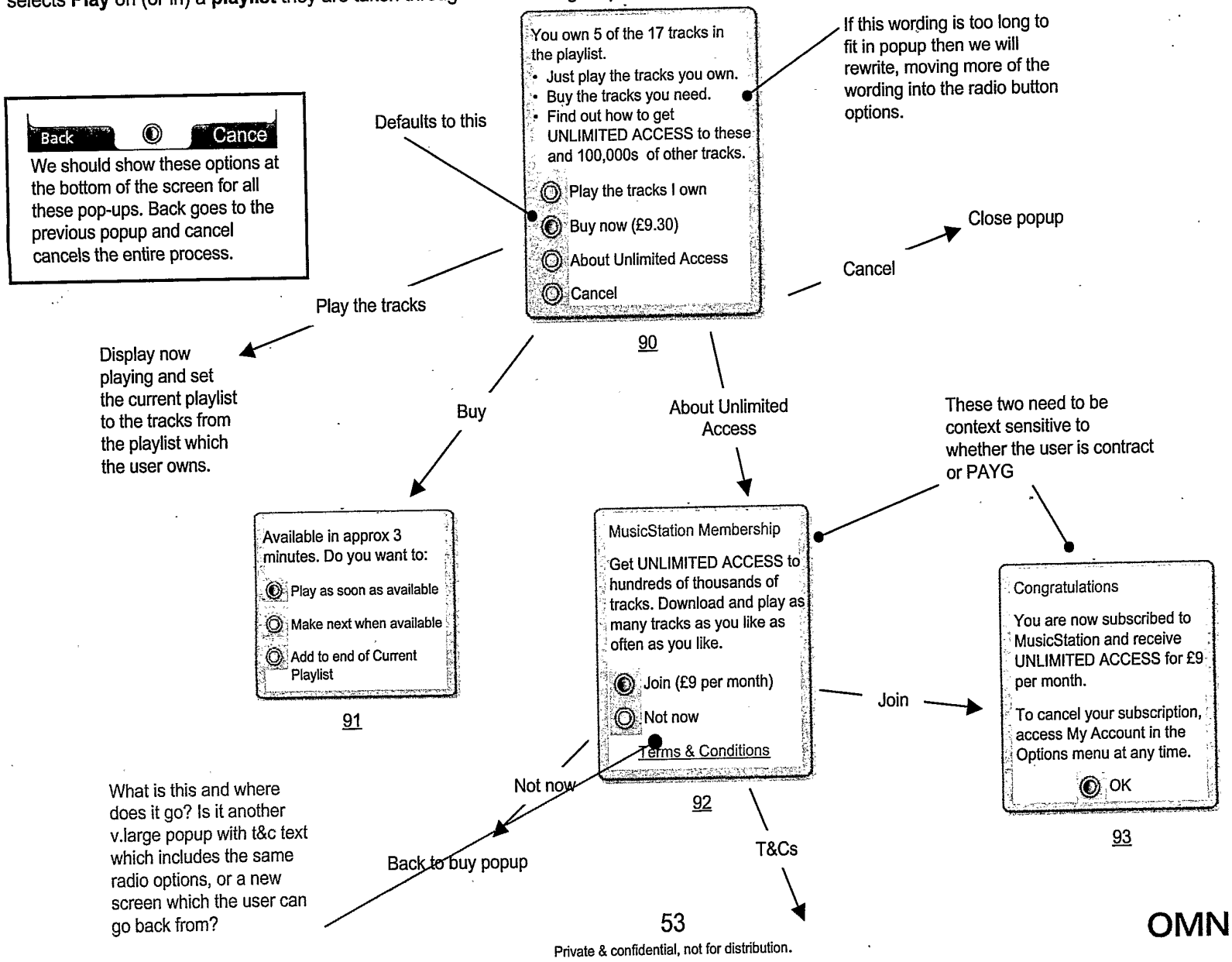
Here is the logic used for a user who is on a blended service where they are allowed to buy individual tracks but can upgrade to a subscription. When the user selects **Play** or **Add to Playlist** they are taken through the following sequence of screens.



OMNIFONE™

90-93: Buy Playlist Process (Play)

Here is the logic used for a user who is on a blended service where they are allowed to buy individual tracks but can upgrade to a subscription. When the user selects **Play** on (or in) a **playlist** they are taken through the following sequence of screens.



OMNIFONE™

95- Add to Playlist Buy Process (Play)

The logic of popups and screens that are displayed when the user selects Add to Playlist on an item which they do not own yet. Needs to prompt for buy/subscribe etc and If they buy/subscribe then it needs to carry onto playlist building rather than asking the "play next?" questions.

TBC

75: Track Lyrics

The place where we will find the lyrics for a track.

Discussion
- Karaoke lyrics

76: Track Info

Some MP3 players allow you to see some details about a track – should we do this?

Track Information

- Available for instance from the Play Track screen via a softkey.
- Includes file name, date received, file size, length (duration) file type (MP3 etc), DRM (distribution)

Feature

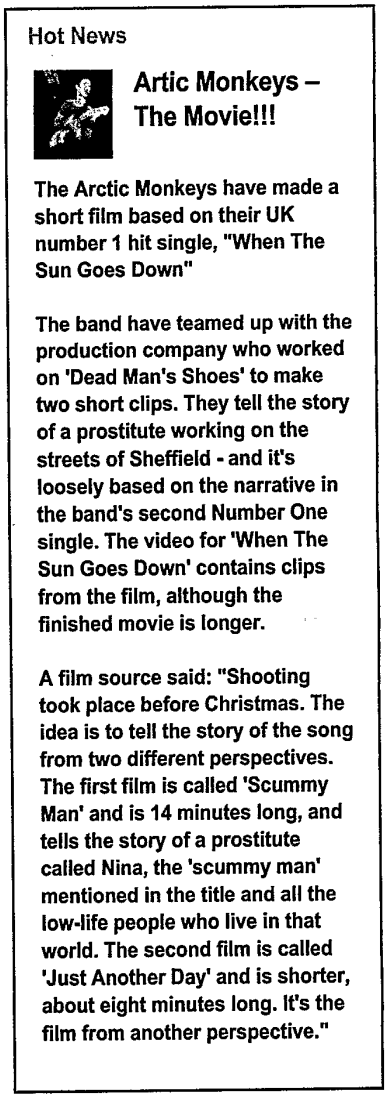
- Blah

300: Inbox & 310: Inbox Story

Details

- We only show stories on the front page of the Inbox. The blue headings are solely that. They are not selectable.

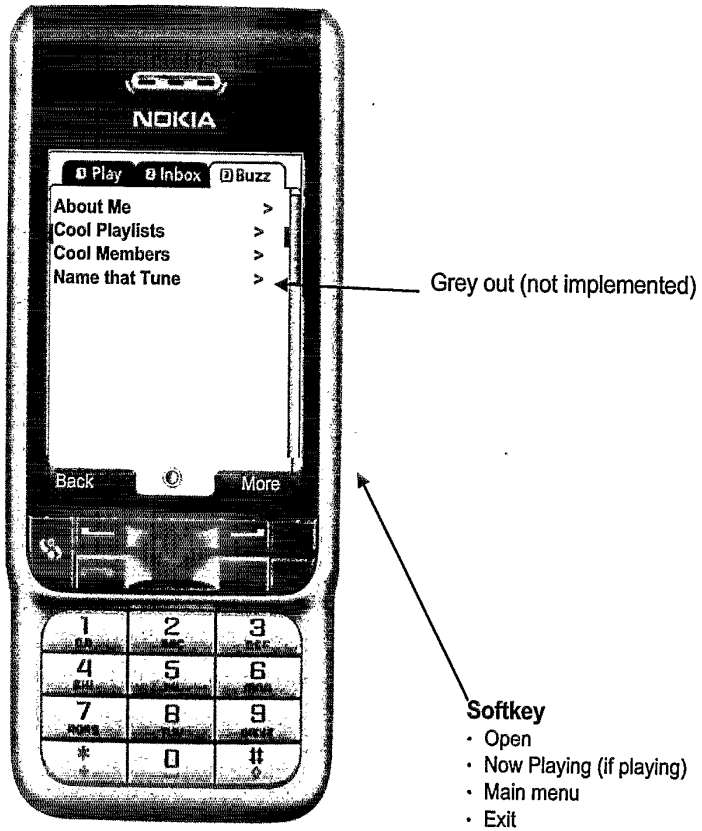
300: Inbox Main



310: Inbox Story

OMNIFONE™

200: Buzz Main



200. Buzz Main

202: About Me

202. About Me

If not defined yet, then "Create My Profile"

210. Edit Profile

220. Cool Factor

10. Playlists Main (but staying in the Buzz tab)

Softkey
- Open
- Now Playing (if playing)
- Main menu
- Exit

OMNIFONE™

204: Cool Playlists

204. Cool Playlists

Top **three** playlists from the playlists charts with editorial descriptions displayed alongside them.

260. Playlist Chart

270. Browse Playlists

12. Playlist Items (no edit, stay on Buzz tab)

Softkey
- Open
- Now Playing (if playing)
- Main menu
- Exit

206: Cool Members *

204. Cool Playlists

Top **three** members from the playmaker charts with editorial descriptions displayed alongside them.

240. Playmaker Chart

250. Browse Members

230. Member home page

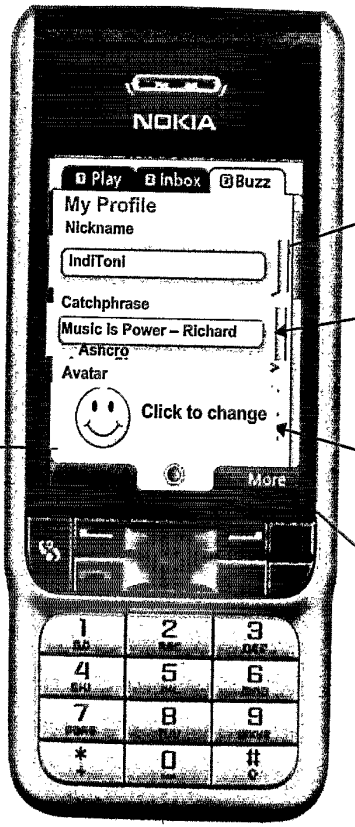
Softkey
- Open
- Now Playing (if playing)
- Main menu
- Exit

210: Edit Profile & Choose Avatar

Allows a user to view and edit their profile

Discussion

- We don't yet have multi-line text entry boxes but we may want to use them for the longer catchphrase item.



Editable text field. Nickname (max of 20 chars)

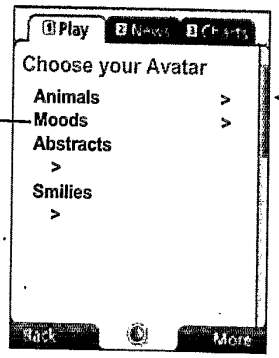
Catchphrase is a max of 80 characters

Member can choose their image from a pre-set selection of cartoon images arranged in 5 groups. Initially set to a random choice.

Softkey

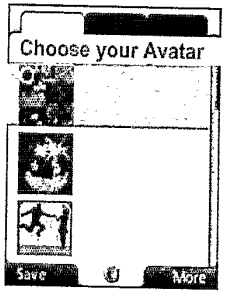
- Open
- Now Playing (if playing)
- Main menu
- Exit

210. Edit Profile



211. Select Avatar Folder

User selects from folders of images.



212 Select Avatar Image

User selects an image from the predefined set. This image selector needs improvement.

220: Cool Factor *

Allows a user to view their various positions and ratings

Discussion

- x

Possible cool factor ratings are:
1.Totally Awesome
2.Super Cool
3.Cool
4.Wannabe
5.Newbie

Image specific to the coolness rating of the user. There's an image per coolness rating

You are rated Super Cool. Your shared playlists have been listened to 1032 times.

You can increase your cool factor by creating and sharing more playlists which other people want to listen to.

Your playlist Banana has got to 100th in the Playlist chart this week!

Standard wording provides cool factor rating, a description of where the user's highest rated playlist is.

220. Cool Factor

230: Member Home page & Playlist Items

Displays details of a member to another user.

Discussion

- Issue with long catchphrase here since will be truncated in this current display. Would need a way for the user to view the full text somehow.

= "System" icons

Title is member name

The member's Avatar image

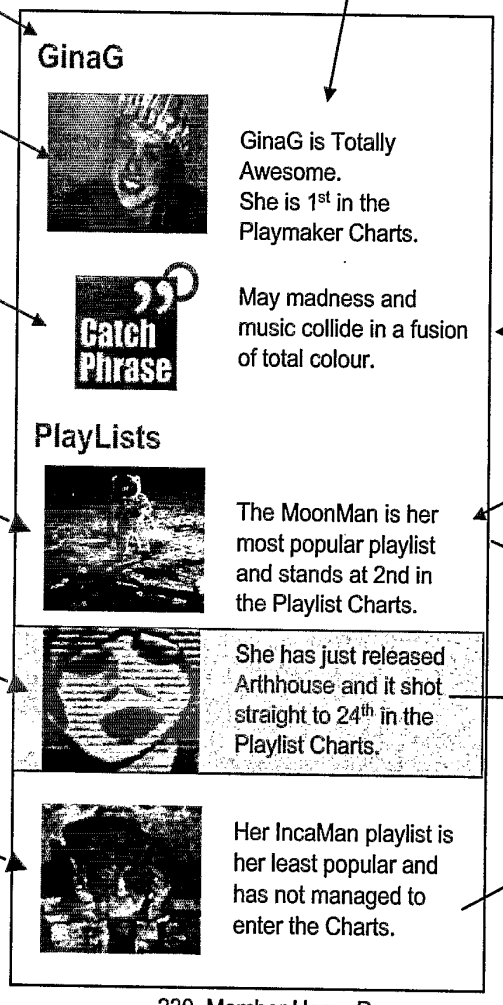
A "catch phrase" icon

Her most played playlist

Her most recent playlist

Her least played playlist

GinaG's cool factor rating and playlist chart position.



240. Playmaker Chart

The member's catchphrase

This is automatic text built around the position of the playlist in the charts.

230. Member Home Page

12. Playlist Items

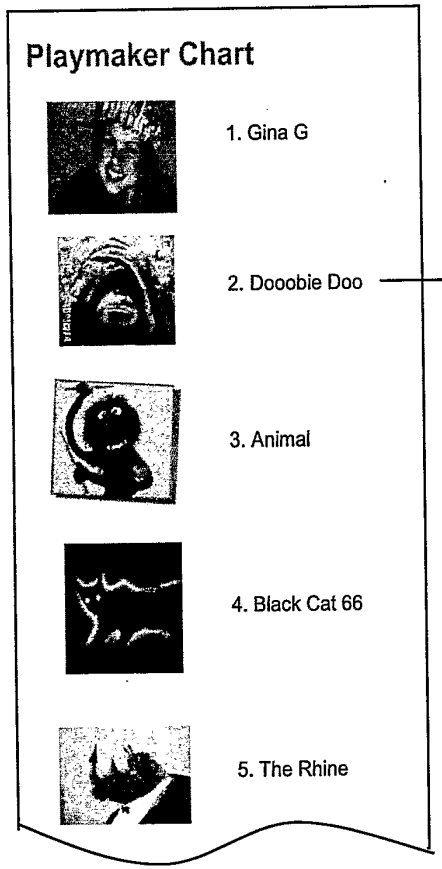
Note this screen will display a playlist in non-editable mode so has none of the edit options like rename, move, etc. Also r that this screen is shown on the Buzz tab.

240: Playmaker Chart *

Allows a user to view the top 20 playmakers (members) along with their position/rating on this chart

Discussion

- There's an issue here with long catchphrases and how users get to see the full text. Only a truncated form of the text (about 50 characters) will be shown on this screen.



230. Member Home Page

The top 20 members showing their avatar and their nickname

We originally had the user shown here displaying their position in the chart, but this will cause issues if they were 19,454th in the chart so have removed it.

240. Playmaker Chart

OMNIfONE™

250: Browse Members

Browse certain groups of members

Discussion

- x

250. Browse Members

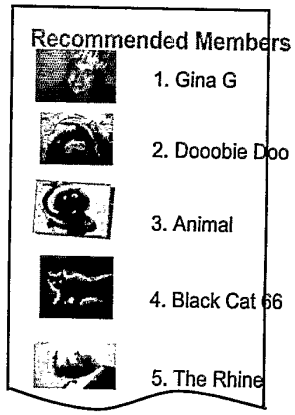
230. Playmaker Chart

251. Recommended Members

252. Featured Members

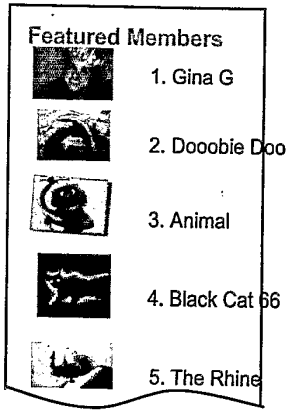
Free text search of members' nicknames and catchphrases. Will return first 50 matching results. No "Next 50" option.

[Grey out for demo]



251. Recommended Members

Members who have listened to the same sort of stuff as you.



252. Featured Members

Editorially controlled list of members

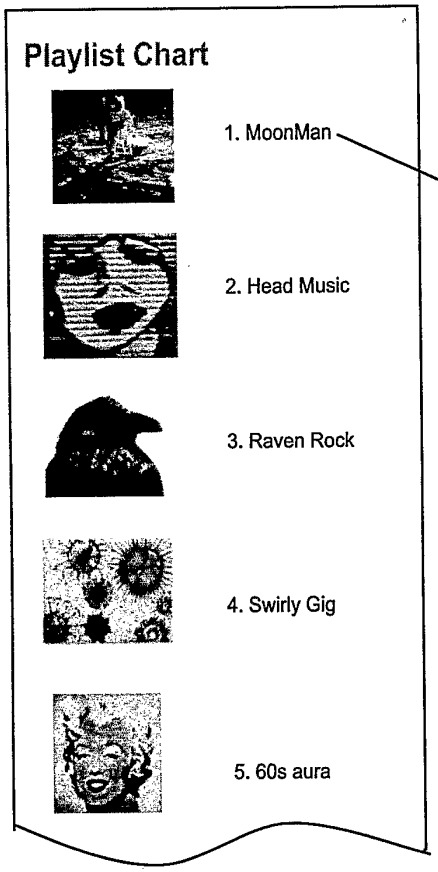
OMNIfONE™

260: Playlist Chart

View the chart of public playlists, and see where your public playlists appear in those charts

Discussion

- x



The top 20 playlists showing their image and their name

For any of the user's **public** playlists not in the top 20 show their playlists listed here, with their positions.

12. Playlist Items

Note this screen will display a playlist in non-editable mode, so has none of the edit options like rename, move, etc. Also note that this screen is shown on the Buzz tab.

260. Playlist Chart

OMNIFONE™

270: Browse Playlists

Allows a user to browse all public playlists

Discussion

- x

260. Playlists Chart

21. You Might Like (but on Buzz tab)

26. Featured Playlists (but on Buzz tab)

Free text search of public playlists' names and track contents. Returns a list of at most 50 items, with no "Next 50/Prev 50" options

[Grey out for demo]

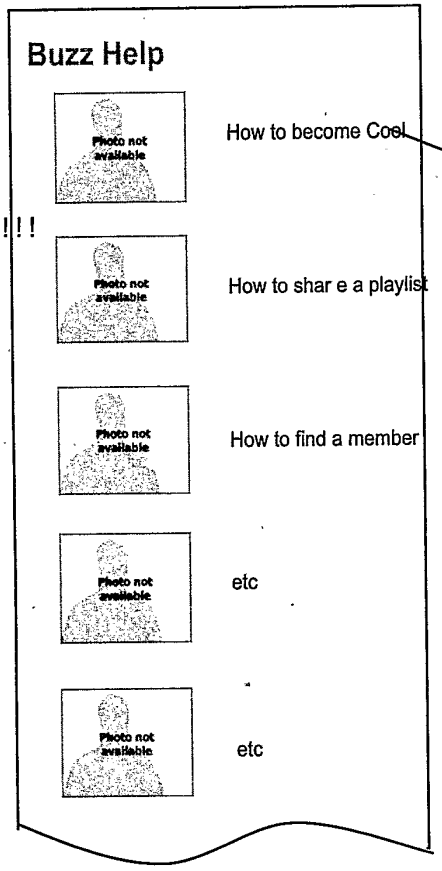
270. Browse Playlists

270: Buzz Help (draft)

Provides help on understanding and how to use Buzz

Discussion
- Show Mark's initial suggestion as to how this help screen should work.
- Not in scope for demo

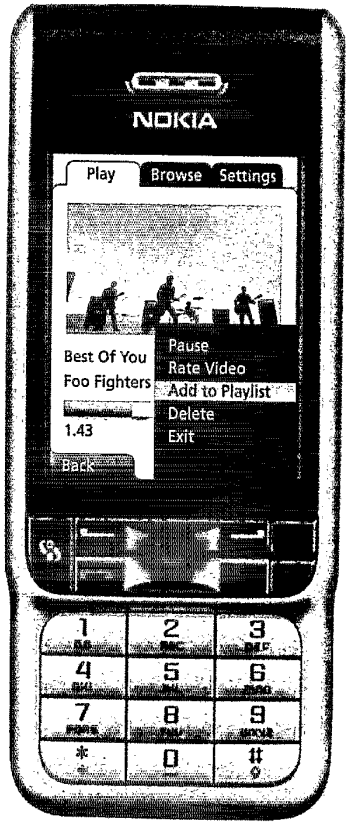
NOT IN SCOPE FOR DEMO ! ! !



280. Buzz Help Main

285. Buzz Help Item

95: Video



Features

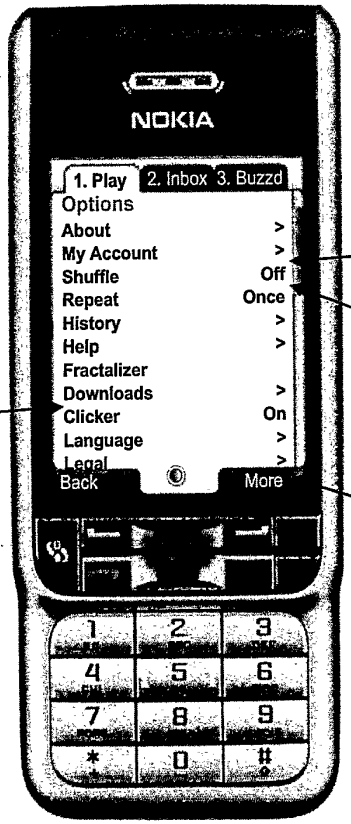
- Go full screen.
- Video clips of yourself.
- How do these work with audio playlists?

OMNIfONE™

100: Options Main

Accessible from the **Main Menu** controls settings and access to other features.

Other Options Main Menu Items
- Autoplay on app launch? On/Off
- Memory settings? Control % of phone memory to use. Control % of memory card to use.
- Tab help On/Off.



100. Options Main

Features
- Opt out of recommendations – under privacy/legal?
- Add history
- My Account needs to be added here (referenced in a couple of popup dialogs for subscription management)

Shuffle
- Off
- Tracks
- Albums

On & Off only? Perhaps also only on current playlist

Confusing?

Repeat
- Off
- Once
- All

Clicker controls if pressing keys makes a sound down the headphones. E.g. Muffled bum on key press.

Softkey
- Open (if folder)
- Select (if option control)
- Now Playing – go to Play Track (if playing)
- Play/Pause (Current Playlist)
- Main menu
- Minimise
- Exit

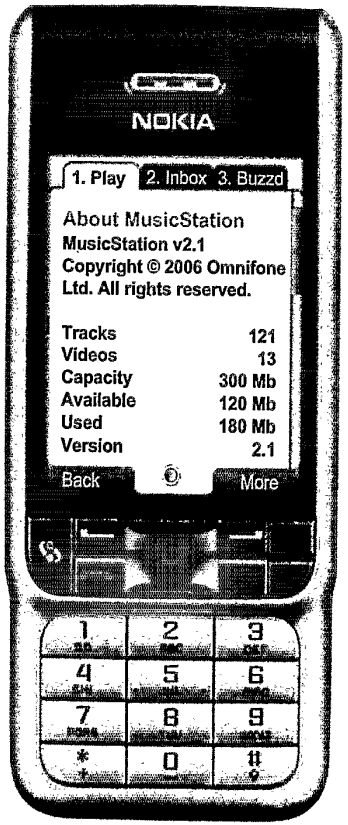
Navigation	
◀◀	Back/Top menu
◀	Back
▲▲	Up item
▲	Up item
●	Select/Open
●●	Select/Open
▼	Down item
▼▼	Down item
▶	Open
▶▶	Open

OMNIFONE™

101: About √

The **About** screen is accessible from the **Options** menu. It shows some key information about the application and it's contents.

Navigation	
◀◀	Back/Top menu
◀	Back
▲▲	Up item
▲	Up item
●	No function
∞	No function
▼	Down item
▼▼	Down item
▶	No function
▶▶	No function



101. About

Feature

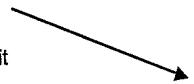
- Patents pending stuff?
- Show a serial number type identifier?

OMNIFONE™

105: History Main

Screen where the history is shown.

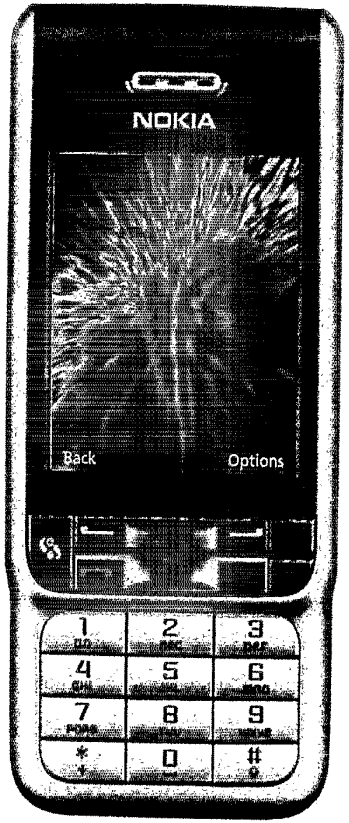
History option appears when there is history. History stored on the server. This is a history browser. Client only stores what it creates, otherwise gets it off the server.



Features

- Today
- Thursday
- Wednesday
- Tuesday
- Monday
- Last Week
- 2 Weeks Ago
- 3 Weeks Ago
- Do we need to go further back and how far with what interface?

115: Fractalizer



115. Fractalizer

Discussion

- Will the battery saver spoil all our hard effort in producing this?
- Still very sexy!
- When should it be shown.

120: Choose Language √

The **Choose Language** screen is accessible from the **Options** menu. It controls the language which the application displays text in. It allows the user to change the language.

Language on Start-up
At start-up the application uses the J2ME variables to check the current language setting of the phone's o/s. If it detects that the language of the phone has been changed from what it was to a new language (which we support) then the language of the application is automatically changed.
Or perhaps we should ask them if they want to do this?

120. Choose Language

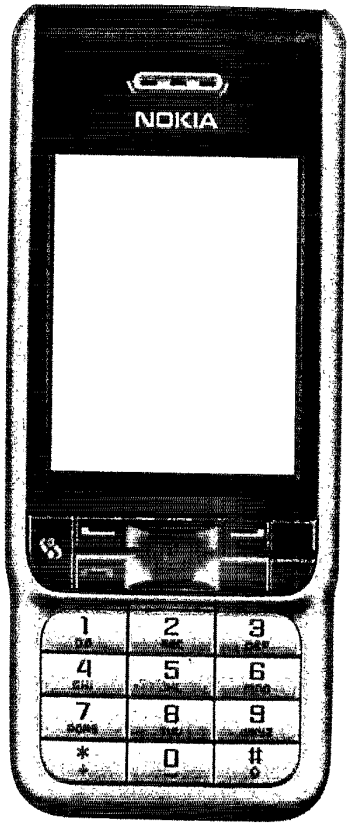
Return to the Options Main screen.

Feature
- Language of the interface is changed dynamically as the language is selected.
- On selecting a language this screen is redisplayed with the title and both softkeys now changed to the newly selected language.
- The language "pack" is either on the client or is downloaded from the network.
- Rob had comments about this and how it should offer options in the current language and the target language (since the user is likely switching from a language they don't know to one they do, or vice versa)

Softkey
- Select (language)
- Now Playing – go to Play Track (if playing)
- Play/Pause (Current Playlist)
- Main menu
- Exit

Navigation	
◀◀	Back/Top menu
◀	Back
▲▲	Up item
▲	Up item
●	Select language
●●	Select language
▼	Down item
▼▼	Down item
▶	No function
▶▶	No function

130: Options: My Account

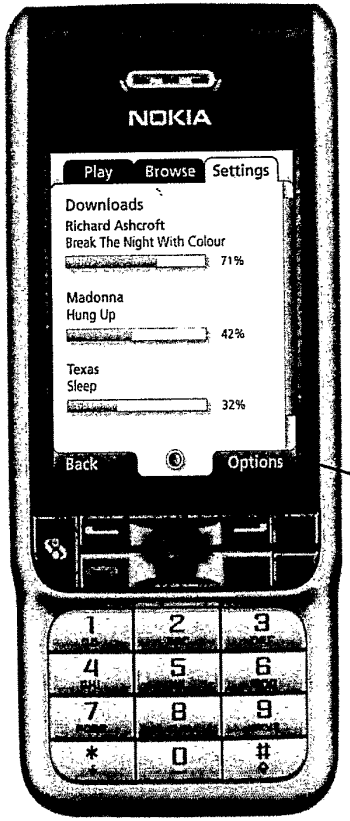


130. My Account

Features

- Presents the member's current status with the serviice using wording like:
- You have not yet upgraded to the full MusicStation service!
- Or, You are subscribed...
- With optoins:
  - Subscribe/Cancel subscription
- More details of what the subscription entails.
- Any identification information like account number/ msisdn/ etc?
- Rob had suggestions for here

140: Options: Downloads



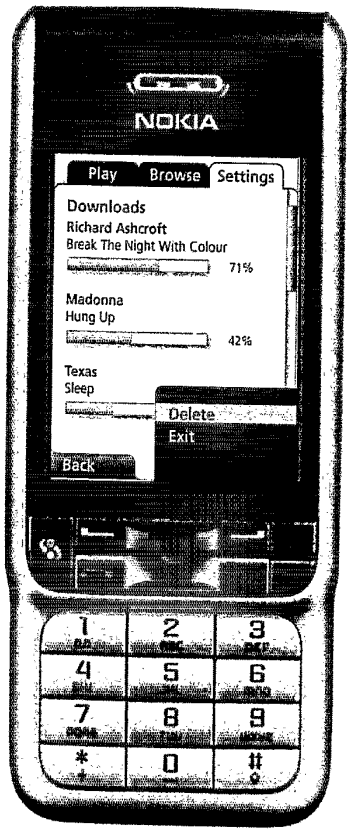
Features

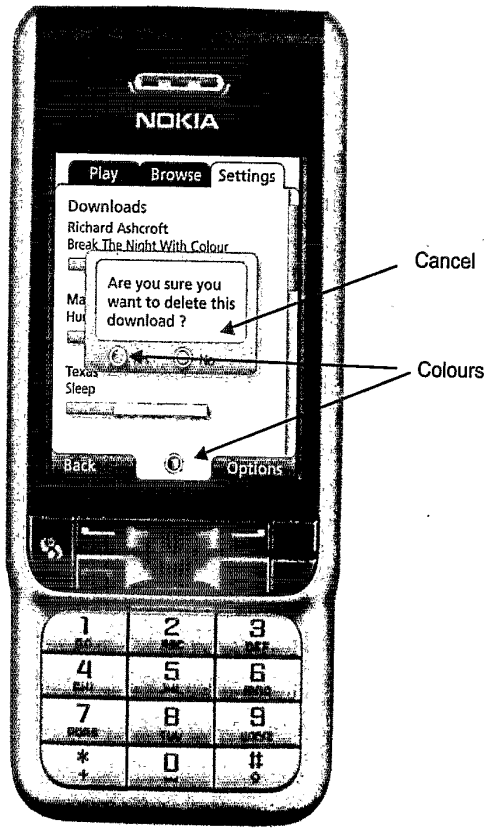
- Ability to control the order of the music queue
- Reprioritise – just like that
- Button/bar colour clash

Softkey

- Rename softkey label to **More.**
- More by Artist
- Stop/Resume
- Stop All/Resume All
- Discard
- Play/Pause (Play Queue)
- Exit

140: Options: Downloads (deleting a download)





TRILLER EXHIBIT 1004-001184

OMNIFONE™

150: Options: Community

Should this be a main menu item?

Blah
Our automated BlueTooth local app carrier detect
Allow the setting of a bluetooth handle

DJ Radio
- Be a DJ for your mates.
- Rate DJs.
- Automated DJs (based off a profile of a virtual DJ who does neighbour matches to play the top stuff for that genre interest.

Users
- Ability to build a profile
- Set up a user handle
- Take a photo and add to profile
- And title so make them sexy.

Community Playlists
- Listen to other playlists
- Rate shared playlists
- Share a playlist
- Playlist charts
- What are other people listening to?
- Opportunity to listen in.
- Top playlists
- Send message with playlist
- Members collections (Their Music)

Artists
- Artist chat

Groups
- Group rate a track or playlist.
- Group screen shows what who is listening to.
- Private groups (invitation only)
- Instant messaging
- Playlist sharing
- Sync listen to music
- Show if share the same music tastes
- Receive playlist
- Send recommendation
- Chat forum
- Voting
- See others albums, playlists & artists
- Make it easy for you to roll over someone else's and take it all or just some of it
- Make it ultracool in the way it does this
- Set people as mates. MyMates

Bluetooth®

OMNIFONE™

160: Options: Help

First Entry

On first entry to the **My Playlists** screen we could try a little education via a popup (right) which also serves to explain why the only item in here is **New Playlist.**

Help popup on first entry to most screens shown on first entry and also off the help (*) hold.

Features

- All help is in the app – or WAP, but not SMS.
- On board help. Audio help with screen control commands embedded. Takes over screen and plays features description audio whilst showing the feature.
- Welcome message audio file. Get a tour. Get going.
- On board help in separate
- Integrated into Options tab.
- Related set of files inside Help in Options. Hidden files unless come through this route or in by context specific.
- Allow retrospective publishing of help patches.
- How Tos when you first start.
- Auto popup if feature not used before and user hesitant.
- One touch help?
- Integrate education into the application. (network based)
- Userbase feedback – how?
- Visual pictures of joystick controls and key controls
- Audio hints on?
- Show picture of keyboard, press button and see what it does.

400: Stream To Device

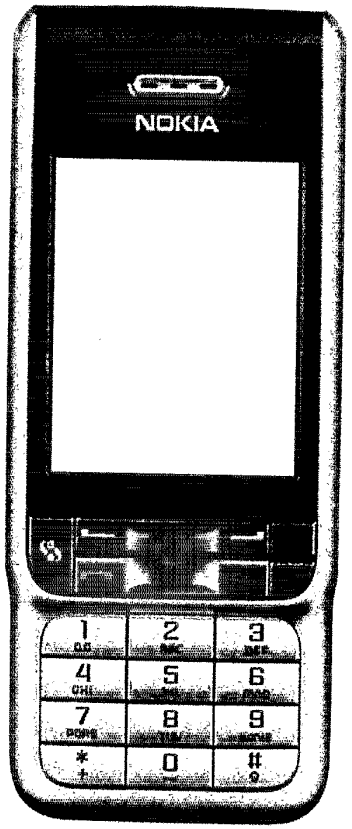
Using the A2DP profile could we stream to a home hi-fi or car stereo?

- Play a stream?
- A2DP BTh streaming

Feature
- Blah

OMNIFONE™

Branding/Whitelabelling



Features

- Background image?

Document made available under the Patent Cooperation Treaty (PCT)

International application number: PCT/GB2007/001675

International filing date: 08 May 2007 (08.05.2007)

Document type: Certified copy of priority document

Document details: Country/Office: GB
Number: 0608932.0
Filing date: 05 May 2006 (05.05.2006)

Date of receipt at the International Bureau: 08 June 2007 (08.06.2007)

Remark: Priority document submitted or transmitted to the International Bureau in compliance with Rule 17.1(a) or (b)

World Intellectual Property Organization (WIPO) - Geneva, Switzerland
Organisation Mondiale de la Propriété Intellectuelle (OMPI) - Genève, Suisse

UK Intellectual Property Office
For Creativity and Innovation

Concept House
Cardiff Road
Newport
South Wales
NP10 8QQ

I, the undersigned, being an officer duly authorised in accordance with Section 74(1) and (4) of the Deregulation & Contracting Out Act 1994, to sign and issue certificates on behalf of the Comptroller-General, hereby certify that annexed hereto is a true copy of the documents as originally filed in connection with patent application GB0608932.0 filed on 5 May 2006.

In accordance with the Patents (Companies Re-registration) Rules 1982, if a company named in this certificate and any accompanying documents has re-registered under the Companies Act 1980 with the same name as that with which it was registered immediately before re-registration save for the substitution as, or inclusion as, the last part of the name of the words "public limited company" or their equivalents in Welsh, references to the name of the company in this certificate and any accompanying documents shall be treated as references to the name with which it is so re-registered.

In accordance with the rules, the words "public limited company" may be replaced by p.l.c., plc, P.L.C. or PLC.

Re-registration under the Companies Act does not constitute a new legal entity but merely subjects the company to certain additional company law rules.

Signed

Dated 30 May 2007

J193660/ 004 D10092 P01/77HOFEE..
05MAY06 0.00 NONE 0608932.0

For Official use only

THE PATENT OFFICE
MC
- 5 MAY 2006
LONDON

Your reference **Testing (UK)**

- 5 MAY 2006

The **Patent Office**

Request for grant of a Patent

Form 1/77 Patents Act 1977

1 Title of invention

MusicStation testing architecture

0608932.0

2. Applicant's details

[X] First or only applicant

2a If applying as a corporate body: Corporate Name

Omnifone Limited

Country

GB

2b If applying as an individual or partnership

Surname

Forenames

2c Address

The Foundry
7 Glenthorne Mews
London

UK Postcode **W6 0LJ**

Country **GB**

ADP Number 8997785001

☐ 2d	Second applicant (if any) Corporate Name Country	
2e	Surname Forenames	
2f	Address UK Postcode Country ADP Number	
3	Address for service	
	Agent's Name	Origin Limited
	Agent's Address	52 Muswell Hill Road London
	Agent's postcode	N10 3JR
	Agent's ADP Number	C03274 727045702

4 Reference Number

Testing (UK)

5 Claiming an earlier application date

An earlier filing date is claimed:

Yes ☐ No ☒

Number of earlier application or patent number

Filing date

15 (4) (Divisional)	8(3)	12(6)	37(4)
☐	☐	☐	☐

6 Declaration of priority

Country of filing	Priority Application Number	Filing Date

7 Inventorship

The applicant(s) are the sole inventors/joint inventors

Yes ☐ No ☒

8 Application fee

The application fee is paid with this form

Yes [ ] No [X]

9 Checklist

		Continuation sheets	
Claims	0	Description	1
Abstract	0	Drawings	0

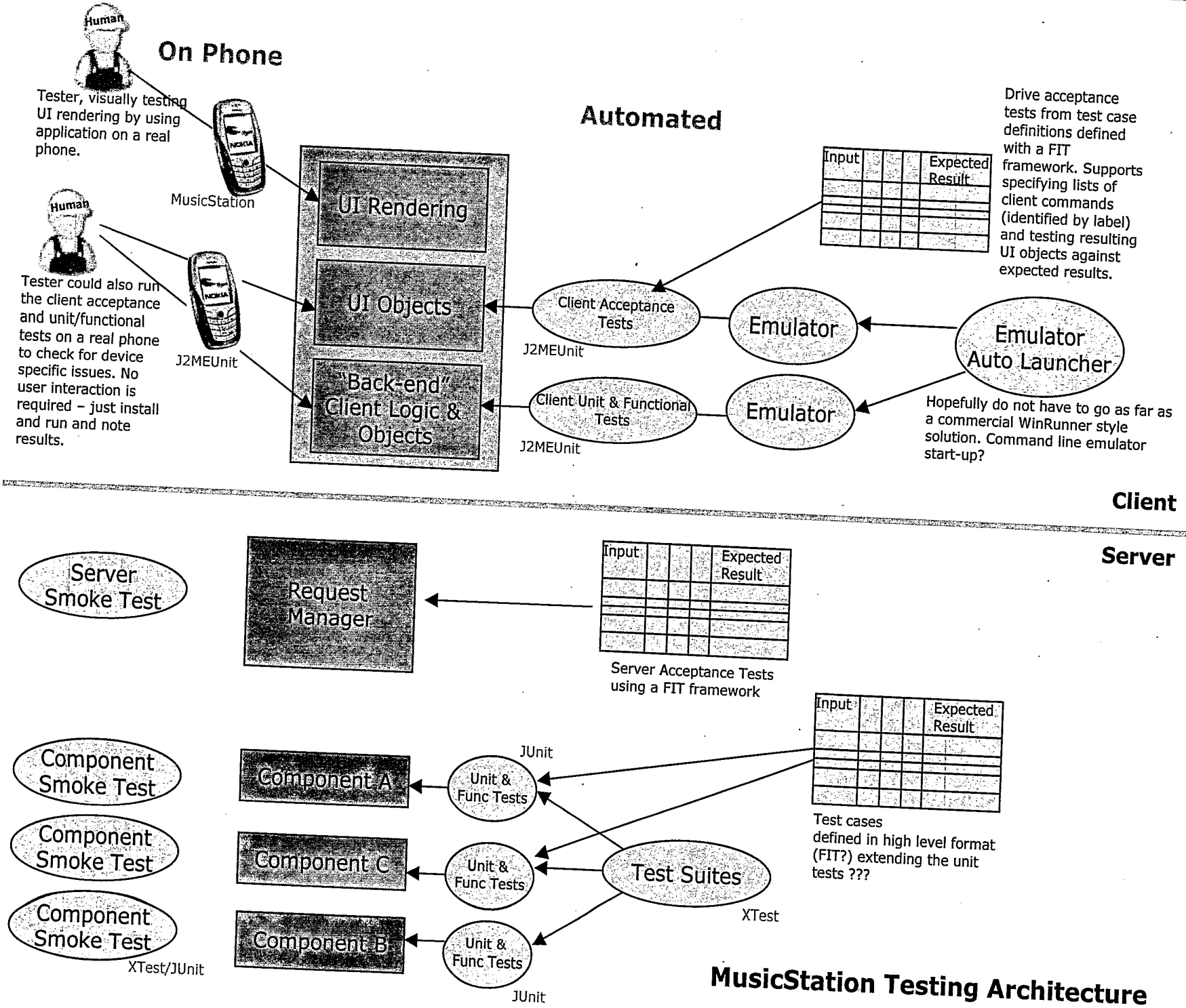
Priority Documents	~~Yes~~/No
Translations of Priority Documents	~~Yes~~/No
Patents Form 7/77	~~Yes~~/No
Patents Form 9/77	~~Yes~~/No
Patents Form 10/77	~~Yes~~/No

10 Request

We request the grant of a patent on the basis of this application

Signed: [signature] Date: 5 May 2006

(Origin Limited)



Document made available under the Patent Cooperation Treaty (PCT)

International application number: PCT/GB2007/001675

International filing date: 08 May 2007 (08.05.2007)

Document type: Certified copy of priority document

Document details: Country/Office: GB
Number: 0608933.8
Filing date: 05 May 2006 (05.05.2006)

Date of receipt at the International Bureau: 30 May 2007 (30.05.2007)

Remark: Priority document submitted or transmitted to the International Bureau in compliance with Rule 17.1(a) or (b)

World Intellectual Property Organization (WIPO) - Geneva, Switzerland
Organisation Mondiale de la Propriété Intellectuelle (OMPI) - Genève, Suisse

UK Intellectual Property Office □□□□
For Creativity and Innovation

Concept House
Cardiff Road
Newport
South Wales
NP10 8QQ

I, the undersigned, being an officer duly authorised in accordance with Section 74(1) and (4) of the Deregulation & Contracting Out Act 1994, to sign and issue certificates on behalf of the Comptroller-General, hereby certify that annexed hereto is a true copy of the documents as originally filed in connection with patent application GB0608933.8 filed on 5TH MAY 2006

In accordance with the Patents (Companies Re-registration) Rules 1982, if a company named in this certificate and any accompanying documents has re-registered under the Companies Act 1980 with the same name as that with which it was registered immediately before re-registration save for the substitution as, or inclusion as, the last part of the name of the words "public limited company" or their equivalents in Welsh, references to the name of the company in this certificate and any accompanying documents shall be treated as references to the name with which it is so re-registered.

In accordance with the rules, the words "public limited company" may be replaced by p.l.c., plc, P.L.C. or PLC.

Re-registration under the Companies Act does not constitute a new legal entity but merely subjects the company to certain additional company law rules.

Signed

Dated 22 May 2007

For Official use only

- 5 MAY 2006

Your reference **Read Me (UK)**

The **Patent Office**

Request for grant of a Patent

Form 1/77 Patents Act 1977

1 Title of invention

MusicStation demonstration pack read me first

0608933.8

2. Applicant's details

[X] First or only applicant

2a If applying as a corporate body: Corporate Name

Omnifone Limited

Country

GB

2b If applying as an individual or partnership

Surname

Forenames

2c Address **The Foundry**
7 Glenthorne Mews
London

UK Postcode **W6 0LJ**

Country **GB**

ADP Number 8997785001

4 Reference Number

Read Me (UK)

5 Claiming an earlier application date

An earlier filing date is claimed:

Yes ☐ No ☒

Number of earlier application or patent number

Filing date

15 (4) (Divisional)	8(3)	12(6)	37(4)
☐	☐	☐	☐

6 Declaration of priority

Country of filing	Priority Application Number	Filing Date

7 Inventorship

The applicant(s) are the sole inventors/joint inventors

Yes ☐ No ☒

MusicStation Demonstration Pack

[FRONT COVER GRAPHICS]

Read Me First

Notices

This version of MusicStation is a 'stand-alone' version designed for the purposes of demonstrating MusicStation's functionality without the requirement for network connectivity. Menu items that are 'greyed-out' represent options that are not currently active, mostly due to the stand-alone nature of this demonstration version.

The music contained on MusicStation is the copyright of EMI. Permission has been obtained for use within this demonstration version of MusicStation.

Contacting Omnifone

If you have any problems or queries relating to using the MusicStation Demonstration Pack, or if you wish to return printed feedback forms by fax or post, then please use the contact details below:

Chris Evans, International Rollout Manager, Omnifone Ltd, The Foundry, 7 Glenthorne Mews, London W6 0LJ, United Kingdom. Tel: +44 (0)20 8846 0970, Fax: +44 (0)20 8846 0971, Email: cevans@omnifone.com

Confidential Information

This MusicStation Demonstration Pack is provided by Omnifone as part of the MusicStation Development Partner Programme evaluation process. All materials and software provided are, and should be treated as, Confidential Information under the terms of the MusicStation Development Partner Programme agreement.

MusicStation Demonstration Pack

Welcome to the MusicStation Demonstration Pack. This pack will allow you to experience MusicStation and enable you to provide feedback into MusicStation's productisation process.

MusicStation is pre-installed on the Nokia N70 provided and ready to run. Installed in the phone already is a pre-pay SIM, a fully charged battery and a memory card.

[Plan view diagram of Demonstration Pack box and contents. Shows box contents out of box slightly and names each of the items.

Items shown include, and are labelled:

- MusicStation Phone (including MusicStation, memory card, battery, & SIM)
- Stereo headphones
- MusicStation Read Me First guide (this document)
- MusicStation Demonstration Pack CD (including feedback form)
- Phone charger]

Providing Feedback

For your convenience a *MusicStation Feedback Form* is supplied in Word format on the Demonstration Pack CD. Simply insert the CD into your CD-ROM drive and copy it to your PC. The locally copied file has a page for each MusicStation screen where you can easily electronically record your feedback. We welcome feedback, so feel free to return as many forms as you like, preferably from a cross-section of different users. Completed feedback forms should be emailed to Chris Evans cevans@omnifone.com. Alternatively manually completed forms should be sent using the contact details on the inside cover of this document.

Getting Started

Connecting the Headphones

The headphones are inserted using the Pop-Port™ connector at the bottom of the phone.

[INSERT 2nd DIAGRAM FROM PAGE 6 OF NOKIA 'GET STARTED' GUIDE – Don't forget to modify the diagram a little]

Starting the Nokia N70

Start the Nokia N70 by pressing and holding the power key on the top of the phone.

[INSERT 2nd DIAGRAM FROM PAGE 8 OF NOKIA 'GET STARTED' GUIDE]

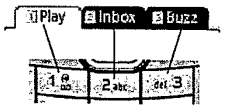
Starting MusicStation

There are two ways to start MusicStation:

- Press the MusicStation key on the right hand side of the keypad. 
- Select the MusicStation shortcut on the Desktop [GRAPHIC OF SHORTCUT].

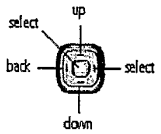
MusicStation Controls

Using Tabs



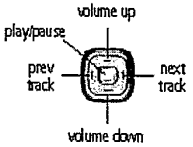
MusicStation has three tab windows, **Play**, **Inbox** and **Buzz**. To change the active tab press keys 1, 2 or 3. Pressing and holding a tab key takes you to the main menu of the tab.

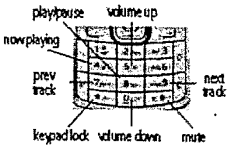
Using Menus



The menus in each MusicStation tab can be easily navigated using the joystick.

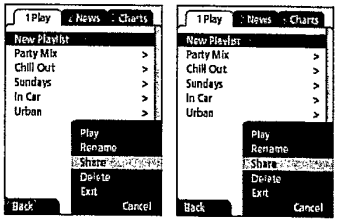
Playing Music





Pressing the 4 key will take you to the **Now Playing** screen if a track is playing. All music playback functions in the **Now Playing** screen can be controlled with the joystick. Keys 5, 7, 8, 9 and 0 perform the same functions as the joystick, allowing music to be controlled from any point within MusicStation. Music playback can be muted or un-muted using the # key and the keypad can be locked or unlocked by holding the * key.

More Options

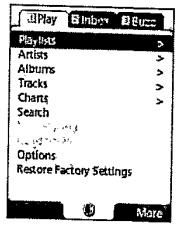


Many screens have extra functions available via the **More** pop-up menu. The menu is accessed by pressing the **More** selection key at the bottom right corner of the screen. It can be operated using the up, down and select keys or closed by pressing the **More** selection key again.

Introduction to MusicStation

Play Tab

The **Play** tab is where music can be browsed, discovered and played. Users can browse by **Playlist**, **Artist**, **Album** or **Track**. The **Charts** menu contains regularly updated relevant music charts (the demo has UK charts). **Search** allows users to quickly locate music in the database using keywords or by a set of advanced search options. **Now Playing** takes the user to a screen of the same name if music is currently being played. **Random Play**, as the name suggests, plays all the tracks on the phone in a random order. The **Options** menu is where various MusicStation settings and information can be found.



Inbox Tab

The **Inbox** is where MusicStation users are kept up-to-date with personalised news and information about the artists they like. Music labels are able to target fans with messages from artists telling them what they are currently up to. **Gigs & Events** keeps fans up to date with the latest information about venues and tour dates. **New Releases** gives users the opportunity to learn about new music from their favourite bands. This feature allows MusicStation users to get new tracks and albums before their friends and potentially affect a track's chart position before the physical release date. The **Inbox** also contains personalised alerts for back catalogue items **Recently Added** and **Hot News** and gossip about the bands they follow.

Buzz Tab

Buzz is where the community thing is happening. It's an area where users can share music and playlists with each other, find people who have similar tastes and see what the coolest MusicStation users are listening to. Users can setup a profile visible to other users, decide which of their playlists they want to share and see what the MusicStation community thinks of their taste in music.

Playlists

Playlists are ways of organising and storing assorted tracks. Any track that is currently being browsed or listened to can be added to a new or existing playlist by selecting the **Add to Playlist** option on the **More** pop-up menu. In future versions of the MusicStation Demonstration it will be possible to:

- Add a photo or image from the phone to represent the playlist.
- Change the position of a Track within a playlist.
- Share a playlist with other MusicStation users.

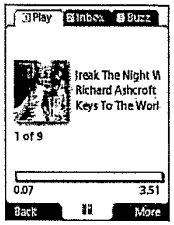
Artists, Albums and Tracks

MusicStation provides quick access to music already on the phone and a convenient method for browsing and locating new music. The **Artists**, **Albums** and **Tracks** menus all work in a similar way to each other. When a user selects the **Artists** menu they are presented with the option to **Get New Artists** and under this they are shown a list of all the artists of music currently on the phone. Selecting **Get New Artists** displays a menu with various options for locating new music by artist.

Now Playing

The **Now Playing** screen is shown whenever music is being played. If the user browses to another part of MusicStation and then doesn't press any keys for a while, the **Now Playing** screen will redisplayed.

In this screen the user is shown the title, artist and possibly the album title and artwork of the currently playing track. Other information includes the position of this track in the **Current Playlist** and the track progress/length. Importantly, whilst music is playing, it is still possible to navigate around MusicStation (e.g. to find other tracks, read news or search for shared music in **Buzz**).



Artist Homepage

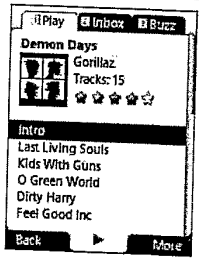
The **Artist Homepage** is the central source for browsing all music by a particular artist. This screen shows all the tracks from this artist currently on the phone and options for getting new tracks and new albums from this artist.

As well as the number of tracks from this artist and the total playtime, each artist has a 5-star overall rating based on user listening habits and explicit rating.

Album Homepage

The **Album Homepage** lists all the tracks from an album in the original published order. From here the user can play the entire album or choose to start playing from a specific track.

Again, a 5-star rating is presented which represents the overall rating for the album based on feedback from users and their listening habits.



You might like...

As users listen to more and more music, MusicStation develops an ever-improving understanding of each user's individual music taste. Users can take advantage of this powerful music discovery capability at various points within MusicStation by using the **You might like...** menu option. In this option users are shown the artists, albums, tracks and playlists that are most popular with people who have similar tastes.

As well as implicitly indicating what they like by playing music, users can explicitly state their preferences by rating artists, albums, tracks or playlists. When a user states whether they love or hate particular music MusicStation takes this rating on board and tailors future recommendations.

Closing MusicStation

MusicStation can be closed by selecting the **Close** item on the **More** pop-up menu. If when they do this music is currently playing the user is asked if they wish to **Close and keep music playing?** or **Close and stop music?**

MusicStation Demonstration Version

Introducing Toni

This MusicStation demonstration version has been configured around the tastes of our imaginary user, Toni, who has been using MusicStation for some time. [PICTURE OF TONI TO RIGHT OF WRAPPED TEXT]

Toni likes Alternative and Indie Rock music and uses MusicStation in 'pay-per-track' mode. She hasn't upgraded to the Unlimited Access subscription service. While using MusicStation Toni has purchased several tracks by Coldplay, Richard Ashcroft and Gorillaz. As a result of these purchases and her listening habits, MusicStation has recommended other tracks and playlists that she might like. Toni has purchased several of these recommended tracks.

Toni's **Inbox** contains news and messages that are relevant to her musical preferences. In the **Buzz** section, Toni has set up a profile under the name **IndiToni**. She has shared some of her playlists and can explore content from other like-minded users.

Exploring MusicStation

We recommend that you listen to some music, build and edit your own playlists. Generally, however, please feel free to browse around and use all the features of MusicStation. We hope you enjoy the experience.

*N.B. The demonstration can be reset back to its original state by selecting **Restore Factory Settings** from the **Options** menu on the **Play** tab.*

Troubleshooting Guide

If you have problems using the MusicStation demonstration version please review this section first. If problems remain please contact Omnifone using the contact details at the start of this guide.

Problem: The Nokia N70 will not start.
Solution: Remove back cover of the Nokia N70. Remove the battery, check the connections are clean, and replace the back cover. If the phone still will not start then try recharging the Nokia N70 using the power adapter supplied.
N.B. The power button requires pressing quite hard and for a few seconds before it responds.

Problem: The Nokia N70 starts but is unusable.
Solution: Turn the phone off. Remove back cover and battery of the Nokia N70. Check that the SIM card is properly fitted and that the connections are clean. Replace the back cover. Try restarting.

Problem: MusicStation will not start, or there appears to be no demonstration content.
Solution: The memory card may have become dislodged. Switch off the phone. Open the memory card port on the right side of the phone. Remove the memory card by pushing it in so that it pops out. Check that the connections on the memory card are clean. Reinsert the memory card with the contacts facing the top. Switch the phone back on. Try restarting MusicStation.

Problem: MusicStation or the phone complains that there is an invalid security certificate.
Solution: Check that the time and date settings are set up correctly on the phone. In order to update it go to Nokia N70 Menu > Clock > Settings > Options.

Document made available under the Patent Cooperation Treaty (PCT)

International application number: PCT/GB2007/001675

International filing date: 08 May 2007 (08.05.2007)

Document type: Certified copy of priority document

Document details: Country/Office: GB
Number: 0702596.8
Filing date: 09 February 2007 (09.02.2007)

Date of receipt at the International Bureau: 26 June 2007 (26.06.2007)

Remark: Priority document submitted or transmitted to the International Bureau in compliance with Rule 17.1(a) or (b)

World Intellectual Property Organization (WIPO) - Geneva, Switzerland
Organisation Mondiale de la Propriété Intellectuelle (OMPI) - Genève, Suisse

The Patent Office
Concept House
Cardiff Road
Newport
South Wales
NP10 8QQ

I, the undersigned, being an officer duly authorised in accordance with Section 74(1) and (4) of the Deregulation & Contracting Out Act 1994, to sign and issue certificates on behalf of the Comptroller-General, hereby certify that annexed hereto is a true copy of the documents as originally filed in connection with patent application identified therein.

In accordance with the Patents (Companies Re-registration) Rules 1982, if a company named in this certificate and any accompanying documents has re-registered under the Companies Act 1980 with the same name as that with which it was registered immediately before re-registration save for the substitution as, or inclusion as, the last part of the name of the words "public limited company" or their equivalents in Welsh, references to the name of the company in this certificate and any accompanying documents shall be treated as references to the name with which it is so re-registered.

In accordance with the rules, the words "public limited company" may be replaced by p.l.c., plc, P.L.C. or PLC.

Re-registration under the Companies Act does not constitute a new legal entity but merely subjects the company to certain additional company law rules.

Signed K Behan

Dated 19 June 2007

For Official use only

J263593/ 006 D10092 P01/77NOFEE....
12FEB07 0-00 ACCOUNT 0702596-8

THE PATENT OFFICE
C D
- 9 FEB 2007

Your reference Big Book One (UK)

0702596.8

The **Patent Office**

Request for grant of a Patent

Form 1/77 Patents Act 1977

1 Title of invention

Big Book One

2. Applicant's details

[X] First or only applicant

2a If applying as a corporate body: Corporate Name

Omnifone Limited

Country

GB

2b If applying as an individual or partnership

Surname

Forenames

2c Address **22 St Peter's Square**
London

UK Postcode **W6 9NW**

Country **GB**

ADP Number 09496662001

DS

☐	Second applicant (if any)	
2d	Corporate Name	
	Country	
2e	Surname	
	Forenames	
2f	Address	
	UK Postcode	
	Country	
	ADP Number	
3	Address for service	
	Agent's Name	Origin Limited
	Agent's Address	52 Muswell Hill Road London
	Agent's postcode	N10 3JR
	Agent's ADP Number	C03274

4 Reference Number

Big Book One (UK)

5 Claiming an earlier application date

An earlier filing date is claimed:

Yes [ ] No [x]

Number of earlier application or patent number

Filing date

15 (4) (Divisional)	8(3)	12(6)	37(4)
[ ]	[ ]	[ ]	[ ]

6 Declaration of priority

Country of filing	Priority Application Number	Filing Date
GB	0608936.1	05 May 2006
GB	0608935.3	05 May 2006
GB	0608934.6	05 May 2006
GB	0608933.8	05 May 2006
GB	0608932.0	05 May 2006

7 Inventorship

The applicant(s) are the sole inventors/joint inventors

Yes [ ] No [x]

8 Application fee

The application fee is paid with this form

Yes [ ] No [x]

9 Checklist

		Continuation sheets	
Claims	0	Description	341
Abstract	0	Drawings	0

Priority Documents	Yes/(No)
Translations of Priority Documents	Yes/(No)
Patents Form 7/77	Yes/(No)
Patents Form 9/77	Yes/(No)
Patents Form 10/77	Yes/(No)

10 Request

We request the grant of a patent on the basis of this application

Signed: Origin Ltd Date: 9 Feb 07

(Origin Limited)

OMNIfONE™

MusicStation Search Whitepaper
v1.2

Omnifone Ltd
Island Studios
22 St Peter's Square
London W6 9NW
United Kingdom

Tel: +44 (0)20 8600 0580
Fax:+44 (0)20 8600 0581

Confidential Information

This information is provided by Omnifone as part of the MusicStation Development Partner Programme. All materials and software provided are, and should be treated as, Confidential Information under the terms of the MusicStation Development Partner Programme agreement.

Table of Contents

1. Introduction

This whitepaper describes the search interfaces, processes and results sets that make up the MusicStation music search. Due to the nature of the mobile working environment the search mechanism within MusicStation has been designed so that it is simple and intuitive to use, whilst at the same time being an extremely powerful feature. Emphasis is placed upon providing relevant and accurate results quickly to the MusicStation customer base.

At the same time as this, it should be remembered that much ongoing automated work is being done in the background to push relevant Artists, Albums, Tracks and Playlists to the customer under the **You might like**, **Recently Added** and **Featured Artists / Albums / Tracks / Playlists** menu options. The contents of these menu options are updated constantly and are based upon a customer's unique tastes and their purchasing and listening habits.

2. Search Interface

2.1 Basic Search

The basic search provides quick but powerful access to the MusicStation music database. The search is performed by the customer entering a **keyword** (or set of keywords) and then further refining their search by one of:

- Artist
- Album
- Track

Additionally it is possible to further restrict the search to **non-classical** music only or **classical** music only, with the default being the selection previously used. Otherwise the system will search both.

2.2 Advanced Search

The Advanced Search screen allows for extensive and finer control to be employed over the search process. Using the **Advanced Search** screen it is possible to filter the result set by:

- Artist, Album or Track
- Genre
- Chart position (highest)
- Minimum customer rating
- Language
- Country

In addition it is possible to search the following fields for **classical music**:

- Work Title
- Album Title
- Composer
- Soloist/Performer
- Conductor
- Orchestra/Ensemble
- Record Label

3. General Principles to Support Searching in MusicStation

There are ten basic principles that have been adhered to in creating the MusicStation search. These principles are provided here with examples where appropriate.

3.1 No dependence on non-alphanumeric characters

Different customers will use non-alphanumeric characters in different ways. For example some may use a hyphen as a separator in an Artist title. Others may simply use a space. In the mobile environment entering non-alphanumeric characters can sometimes be tricky and is prone to error. Therefore, for the purposes of search, there is no dependency on non-alphanumeric characters, and by way of example, the following are all considered to be equivalent:

- **s club 7**
- **s-club-7**
- **sclub7**

3.2 No dependence on character case

This simply means that, for example, the following are considered equivalent

- **s club 7**
- **S CLUB 7**
- **S Club 7**

3.3 International variations of characters are treated as equivalent

Different customers will use non-English characters in different ways. For example a British person may search for:

- **Bjork**

..when what they really should be searching for is:

- **Björk**

In MusicStation such discrepancies do not matter since the search system matches international variation of English letters to their English alphabet equivalents (and vice versa).

3.4 Numerics are treated as the same as their written equivalents (and vice versa)

In an Artist search a customer may enter, for example, "**50 Cent**" or "**Fifty Cent**". Both these cases are handled by the system.

3.5 Abbreviations and different ways of writing words do not matter

Internal mapping tables ensure that commonly used abbreviations and equivalent representations are understood. Thus the following keywords are all be considered by the system as equivalent:

- **Boys to Men**
- **Boys 2 Men**
- **Boys II Men**

In a similar vein "and" and "&" are considered to be equivalent.

3.6 There should be no dependence on correct positioning of "The"

We are not concerned about how "The" is used. For example the following keywords are all be seen as equivalent by the system and the correct result will be returned:

- **The Rolling Stones**
- **Rolling Stones, The**, or simply:
- **Rolling Stones**

3.7 Customers do not always enter the full set of keywords

Someone searching for "**Rage Against the Machine**" may simply enter "**Rage**" as a keyword and expect MusicStation to return a sensible set of results to choose from.

3.8 Customers do not always spell words correctly

Though we are primarily looking for an exact match we recognize that customers often mis-spell words. We use fuzzy-logic and phonetic matching techniques to suggest search Artists, Albums or Tracks to the customer when all else fails.

3.9 If they know what they want then take them there

If a customer searches for "**rage against the machine**" and this results in 1 result being returned then they will be forwarded automatically to the "Rage Against the Machine" Artist Homepage. We will not present them with a result set containing a single Artist that they must then click on.

3.10 We will learn from use of the system and optimize it accordingly

There may be variations of Artist, Album or Track names that customers search for, that are quite different to the one stored in the database. Structures exist to ensure that when we see a new variation in a search keyword, we are able to match it to the intended Artist, Album or Track name, thus ensuring that all future searches using that variation are successful.

Similarly, when sorting the search results will make use of knowledge of the popularity of the results (as played by customers) ensuring that the most popular (and hence the most likely result for the intended search) are nearer the top. When this is occasionally not correct the customer may choose an alpha-sorted view instead.

4. The Search Process

The following is a description of the search process from the point where the customer enters their search keyword(s) for an Artist search.

N.B. The same principles below are also applicable to the Album or Track searches.

1) An exact match for the entered **search keyword(s)** is searched for, but based on the underlying principles outlined in Section 3 - *General Principles to Support Searching in MusicStation.*

2) We then search for instances of the **search keyword(s)** *within* the Artist **names**. For example, given the search keyword "BOB MARLEY", valid matches are:

   a) "**BOB MARLEY ***"

   b) "* **BOB MARLEY** *", and:

   c) "* **BOB MARLEY**"

   (where * is a 'wildcard' representing any sequence of characters)

Matches of type (a) are viewed as higher priority in the returned results list than those of type (b) and (c).

If (1) and (2) returns only 1 match then we go direct to Artist homepage (and the Album Homepage for Albums, and the Now Playing screen for Tracks).

3) Otherwise we list matches from 1), followed by matches from 2) ranked by popularity and then alphabetically.

If we have found matches from the above then we leave the search routine. Otherwise we move on to **approximate matching**:

4) We repeat steps 1) to 4), but this time by making use of phonetic and fuzzy-logic matching to find matches that sound similar to the keyword or are spelt slightly differently. Any matches that are returned from this process are preceded by the header: "**No exact matches found. Did you mean:**" so that it is clear to the customer that the search results are not precise matches. The result set is again ranked by popularity and then alphabetically.

5. Finding in results

In instances where the results list is large the customer may search for more specific items by using the 'Find' option on the 'More popup' menu to navigate through the list looking for a specific string. When the customer submits the first occurrence of it is found. The next result can be moved to quickly by use of the 'Next' option on the left-hand soft-key.

6. Refining the Search

It is possible to refine a search from the results set page using an option on the 'More popup' menu. What this means is that the user may search again (in either the Basic or Advanced Search) but with the search keyword box and all pre-selected filters maintained allowing for them to be quickly refined.

7. Format of the Search-Results Set

When a search results in a search result-set being returned the count of elements in the set will be presented in the top right of the page.

The format of the actual results themselves is different dependent on whether the search was for an Artist, Album or Track. These formats are described in greater detail in this section.

7.1 Artist Search

The top **Artist Name** matches are returned, sorted by popularity of the Artist as measured by the system. These are followed by further matches of similar (but lower) popularity, sorted in alphabetical order.

7.2 Album Search

The Album search will return results in the following format:

Album Name – Artist Name (Year of Release)

Having the 'Year of Release' ensures that, for example, re-releases (which may contain bonus or updated tracks) can be easily distinguished from the original.

The top matches are returned sorted by popularity of the Album as measured by the system. These are followed by further matches of similar (but lower) popularity, sorted in alphabetical order.

7.3 Searching Tracks

The Track search will return results in the following format:

Track Name – Artist Name (Track Length)

Having the 'Track Length' (in **mm:ss**) ensures that Tracks having the same name (but of different length) can be distinguished. This can often occur with re-mixes on different Albums.

N.B. Having the Album Name here is deemed as unnecessary and undesirable due to the overall length of the string that would result in what is a tightly restricted environment. Also, if the same Track occurs on different albums, then it will only be returned once.

The top matches are returned sorted by popularity of the Track as measured by the system. These are followed by further matches of similar (but lower) popularity, sorted in alphabetical order.

8. Find in Playlists

At suitable points in the system, when a Track is being referenced, the customer can search for that Track within Playlists by using the 'More popup' option 'Find in playlists'. A list of Playlists shared by other MusicStation customers (or contained within other system-published Playlists) is returned, sorted by popularity.

OMNIfONE™

MusicStation Recommendations Whitepaper v1.2

Omnifone Ltd
Island Studios
22 St Peter's Square
London W6 9NW
United Kingdom

Tel: +44 (0)20 8600 0580
Fax:+44 (0)20 8600 0581

Confidential Information

This information is provided by Omnifone as part of the MusicStation Development Partner Programme. All materials and software provided are, and should be treated as, Confidential Information under the terms of the MusicStation Development Partner Programme agreement.

Table of Contents

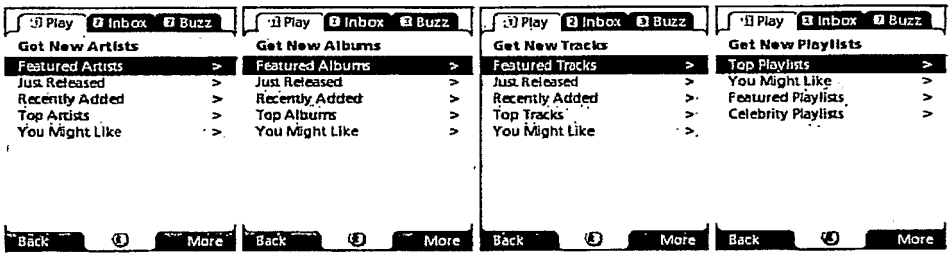
1. Introduction

This document describes the approach taken with making recommendations to customers from within the MusicStation application. Omnifone views the ability to make ever-changing, relevant and up-to-date recommendations as key to the strategy of creating loyalty towards the MusicStation application. Recommendations, properly implemented, encourage exploration and discovery that in turn lead to more purchases of new music. Additionally they allow us to optimise the MusicStation experience in the restricted mobile environment.

2. Recommendations within MusicStation

MusicStation contains several features designed at promoting personalised recommendations to the customer. These features are spread over the **Play**, **Inbox** and **Buzz** tabs and are described in detail in the following sections.

2.1 Recommendations in the Play tab



Whenever a customer selects the **Get New Artists, Albums, Tracks** or **Playlists** option from the **Main Menu**, they are presented with a list of menu options, some of which are personalised suggesting recommendations to the customer based upon their recent listening habits.

The menu items that contain personalised recommendations are described here:

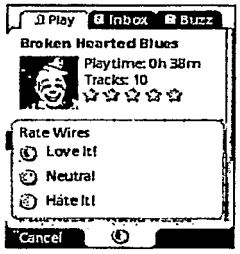
Personalised Menu Item	Recommendations Contained
You Might Like	Artists / Albums / Tracks / Playlists recommend to the customer based on their recent listening habits, and taking into account any explicit music ratings that they have made. Artists / Albums / Tracks / Playlists that the customer has already listened to (under the subscription all-you-can-eat mode) or purchased (under the pay-per-track model) are automatically filtered out.
Recently Added	A list of back-catalogue Artists / Albums / Tracks that have been recently added to the system. Even those that are new to the system could potentially be old back catalogue releases. This list is based on the customer's recent listening and rating habits.
Featured Artists / Albums / Tracks / Playlists	A list of Artists / Albums / Tracks / Playlists that have been editorially pushed for promotion, and personalised for the Artists / Albums / Tracks / Playlists and Genres the customer listens to. Artists / Albums / Tracks / Playlists that the customer has already listened to (under the subscription all-you-can-eat mode) or purchased (under the pay-per-track model) are automatically filtered out.

2.2 Information that influences music recommendations

Music recommendations for the **Play** tab are made based upon the interaction of two factors unique to the customer:

- The **implicit** factor: This is based upon the listening habits of the customer (i.e. the type of music they listen to and the frequency with which they listen to it).
- The **explicit** factor: How the customer actually rates music that they listen to.

Also counting towards the **implicit** factor will be any click-throughs on **Inbox** content that the customer has made (for more information please refer to section 2.4 - *Recommendations in the Inbox tab*)



In terms of the **explicit** factor Customers are pushed recommendations for music similar to other music that they have rated as **Love it**, and are not recommended any content that is defined as similar to music they have rated with **Hate it**.

2.3 Making music recommendations

These **implicit** and **explicit** factors for each customer are combined, and mixed with known relationships between Artists and other Artists, Tracks and other Tracks and so on. The outcome is a list of personalised recommendations to the customer.

For version 1.0 of MusicStation, the 'known relationships' mentioned have been purchased from an industry-leading third-party supplier, are maintained regularly to a highly professional standard, and will be updated on a weekly basis (with new release information coming into the system prior to it going live). Overall this ensures that MusicStation will hit the ground-running in terms of being able to make recommendations and *go-live.*

The information provided covers relationships on Artist **influences**, **contemporaries** and **followers**, as well as key Tracks and definitive Albums.

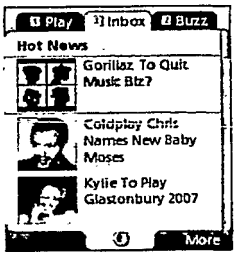
Over time, as we collect information on which Artists, Albums, tracks and Playlists are popular (or not), these recommendations will become even more tightly geared to what the customer is most likely to find relevant. The system will automatically push to the customer the most popular Artist, Albums or Tracks that have direct relationships with the top Artists / Albums and Tracks that the customer listens to or purchases.

2.3.1 The importance of 'recency'

It is important that recommendations are only made based upon the customer's recent listening habits, and not their listening habits for all time. This ensures that the suggestions are the most relevant to the customer at the time of creation, and do not consist of a clouded swathe of very broad suggestions that are influenced by a customer that may have, say, very changeable and diverse tastes.

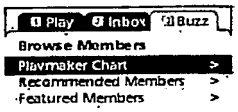
In terms of MusicStation, **recency** is defined by the **last N** Artists / Albums / Tracks or Playlists that the customer has listened to or purchased. The actual value of **N** is configurable based on observations, enabling fine-tuning over time of the recommendation process.

2.4 Recommendations in the Inbox tab

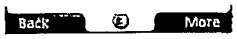


All **Inbox** content (news stories, events notifications, promotions for particular Artists etc.) is personalised to the customer based upon the same **implicit** and **explicit** factors described for the **Play** tab. Additionally, as described previously, where users click-through on **inbox** content, e.g. a promotion linking to an Artist homepage, this event is tracked, and is then used as a 'positive vote' for that Artist in the overall recommendation process.

2.5 Recommendations in the Buzz tab



The **Buzz** tab contains two elements that contain recommendations directed at each individual customer. These are described in the table below:



Personalised Menu Item	Recommendations Contained
Recommended Members	Customers who are similar to the current customer based on their recent listening and rating habits.
Featured Members	Editorially controlled members personalised to the individual customer based upon their recent listening and rating habits.

Recommendations for members (i.e. MusicStation customers) are made by linking customers whose listening and rating history for music is similar (internally the system measure the 'affinity' of customers to all other customers, and select those with the highest level of affinity for the customer in case).

If a customer selects a recommended member then they are able to listen to, and rate, their Shared Playlists.

3. Next Stages for MusicStation Recommendations

In a later version of MusicStation a **collaborative filtering** approach will be introduced for making both music and customer recommendations, whereby the system will:

- Look for customers who share the same listening and rating patterns with the active customer (the customer who the prediction is for).
- Use the listening and rating patterns from those like-minded customers found to calculate a prediction for the customer in question.

This information can then be used to create accurate, and ranked, music predictions for the user.

We are actively evaluating the offerings of several key industry players for partnership on the collaborative filtering system, the potential partners being:

- MusicStrands
- MusicIP
- MediaUnbound
- Gracenote
- MyPlaylists
- Last.FM

By using a collaborative filtering system, music and customer relationships will evolve organically and in real-time, allowing for an even more powerful recommendation system based upon customers' ongoing interactions and tastes.

OMNIfONE™

MusicStation: Searching, List Handling and Data Quality

ver 1.4

26th June 2006

Chris Evans

Table of Contents

Introduction

This document describes a complete methodology for handling fast, accurate, and forgiving searches within the MusicStation environment. It has been put together on the basis that mobile users want easy, but powerful access to their music, without any speed penalty in what is already a relatively cumbersome networked environment.

1. Search (general)

The search should be available from the right-hand menu from ALL three tabs (i.e. Play, Inbox and Buzz). This is logical since you should be able to perform searches in all 3 areas. The name of the Search option should change depending on the context, i.e.:

- "Search " (better than "Search Play" for this option)
- "Search Inbox" (possibly in v2)
- "Search Buzz" (i.e. search for Users, though in V1 this is essentially a shared Playlist Search) (possibly in v2)

2. Basic Search

Within the basic search it should be possible to search by:

- Artist
- Album
- Track

DISCUSSION: Perhaps we should have an option under Options that says "Only search classical music" and "Never search classical music" for the basic Search. We should remind people of the status of this when they perform the search.

3. Match-keys

For the purposes of quick and effective matching:

1) on search terms, and
2) across different data sets e.g. two external databases, that have no common IDs (such as UPC codes) and that we can only match on Artist, Album and Track names.

we should set up pre-prepared indexed ***match-keys*** within the database based on Artist, Album and Track names. There should be 2 keys for each Artist, Album and Track name, the **main match-key**, and an **approximate match-key**.

The main match-key allows for accurate matching on similarly transformed user search terms and guarantees a good level of confidence that the result from the search is what was intended.

The **approximate match-keys** can be used for matching search terms entered without any whitespace, e.g. you can easily imagine someone entering "sclub7" we they should really write "S Club 7".

3.1 Examples of match-keys and their original Artists names:

Original	Match-key	Approx. Match-key
Boys II Men	BOYS 2 MEN	BOYS2MEN
Boyz 2 Men	BOYZ 2 MEN	BOYZ2MEN
Boyz To Men	BOYZ 2 MEN	BOYZ2MEN

Examples of Artist name duplicates, that exist in Sandy's database, and that our match-key should enable us to consolidate:

Agnostic Front & Discipline
Agnostic Front: Discipline

All Natural Lemon & Lime Flavors
All Natural Lemon & Lime Flavours

Allen,Pete: Jazz Band
Allen,Peter: Jazz Band

Andrew Lloyd Webber
Andrew Lloyd-webber

C&c Music Factory
C+C Music Factory

(query used was *select distinct artist from release order by artist*)

3.2 Match-key creation

The match-keys for the main data set should be created only when new data is added to the system, The real-time application occurs when a user enters a search term. The search term is first converted into a match-key, following the same rules, before any comparison is made.

3.2.1 Creating the *main match-key*

1) Rewrite word "NO." or "#" as "NUMBER". By ***word***, we mean instances of "NO._", "_NO._" and "_NO.", where underscore represents a space.
2) Remove all non-alpha and non-numeric chars. Replace foreign chars with English equivalent.
3) Make all upper case.
4) Remove the **word** "THE". By ***word***, we mean instances of "THE_", "_THE_" and "_THE", where underscore represents a space. Remove the **words** "A" and "AN".
5) Remove any instances of the **words** " AND ", "n" ("&" has already been removed due to preceding rule).
6) Rewrite **words** "to", "two", "II", "too" as "2". Rewrite **words** "for", "fore", and "four" as "4 ".
7) Rewrite word "Pt" as "Part" (particularly important in classical music).
8) Take each space-separated "word" and perform the following:
    i) Rewrite **words** "ONE" as "1", "TWO", as "2", ,,, "NINETEEN" as "19"....
    ...(requires lookup table)
    ii) Rewrite "FIRST" as "1ST", SECOND as "2ND" etc...(requires lookup table)

3.2.2 Creating the *approximate match- key*

9) The approximate match-key will simply be the same as the normal match key but with all whitespace removed.

N.B. when a process involves removing words, it does not run if the removal process would result in the creation of an empty string (or just whitespace).

3.2.3 Example - creating the match-key

	Transformation Description	String Result Example 1	String Result Example 2
		Siouxsie & The Banshees	Three Lions (Football's Coming Home) (Party Anthem 2006)
1	Rewrite word "NO." or "#" as "NUMBER". By ***word***, we mean instances of "NO._", "_NO._" and "_NO.", where underscore represents a space	Siouxsie & The Banshees	Three Lions (Football's Coming Home) (Party Anthem 2006)
2	Remove all non-alpha and non-numeric chars. . Replace foreign chars with English equivalent.	Siouxsie The Banshees	Three Lions Footballs Coming Home Party Anthem 2006
3	Make all upper case.	SIOUXSIE THE BANSHEES	THREE LIONS FOOTBALLS COMING HOME PARTY ANTHEM 2006
4	Remove the **word** "THE". By ***word***, we mean instances of "THE_", "_THE_" and "_THE", where underscore represents a space.	SIOUXSIE BANSHEES	THREE LIONS FOOTBALLS COMING HOME PARTY ANTHEM 2006
5	Remove any instances of the **words** " AND ", "n" ("&" has already been removed due to preceding rule	SIOUXSIE BANSHEES	THREE LIONS FOOTBALLS COMING HOME PARTY ANTHEM 2006
6	Rewrite **words** "to", "two", "II", "too" as "2". Rewrite **words** "for", "fore", and "four" as "4 ".	SIOUXSIE BANSHEES	THREE LIONS FOOTBALLS COMING HOME PARTY ANTHEM 2006
7	Rewrite word "Pt" as "Part" (particularly important in classical music).	SIOUXSIE BANSHEES	THREE LIONS FOOTBALLS COMING HOME PARTY ANTHEM 2006
8	Take each space-separated "word" and perform the following: o Rewrite **words** "ONE" as "1", "TWO", as "2", ,,, "NINETEEN" as "19".... ...(requires lookup table) o Rewrite "FIRST" as "1ST", SECOND as "2ND" etc...(requires lookup table)	SIOUXSIE BANSHEES	3 LIONS FOOTBALLS COMING HOME PARTY ANTHEM 2006
9	Remove any superfluous whitespace	**SIOUXSIE BANSHEES**	**THREE LIONS FOOTBALLS COMING HOME PARTY ANTHEM 2006**

4. AKA Fields

We need to be able to associate multiple variations of Artist, Album and Track names with a particular Artist, Album, or Track. These names will be managed by our content controller based on monitoring of the types of search terms people are using, i.e. if many people are searching for "Bonn Jovy" (as opposed to "Bon Jovi"), our content manager will associate "Bonn Jovy" with "Bon Jovi" by adding a record to an artist_aka table. Subsequent searches for "Bonn Jovy" will then match "Bon Jovi".

5. Searching Artists

5.1 Napster Use Cases for Artist Search

From experience, Napster handles music searching in a simple but powerful way. As such there is much we can learn from a set of use cases that test the way in which Napster utilises the entered search-term on order to match with the desired result. Nine such use cases are listed below.

Napster comparison 1: "50 Cent"
"50 Cent" – returns list of matching Artists, "50 Cent" ranked first... i.e. returns one exact match
"Fifty Cent" – goes direct to music homepage of "50 Cent"
"FiftyCent" – returns no results
"50Cent" – goes direct to home page of "50 Cent", i.e. returns one exact match.

Therefore we need to maintain a list of alternative naming for Artists, Albums, Tracks etc...

Napster Comparison 2: "S club 7"
"sclub7" – goes direct to "S Club 7" homepage
"s-club-7"– goes direct to "S Club 7" homepage
"s club"7"– goes direct to "S Club 7" homepage
"s-club seven" – goes direct to "S Club 7" homepage
"s-club" – give list of Artists with "s club" in..
"s clb 7" – give list of Artists with "s club" in..

Napster Comparison 3: "U2"
"U2" – list of Artists with "U2" in
"u 2" – goes direct toU2 homepage (there fore appears to do an **exact match** on Artist name once whitespace and other chars removed)
"u two"– does not resolve to "U2"
"you 2"– returns Artist list with "U2" at the top!!!
"you2" – no results found

Napster Comparison 4: "Franz Ferdinand"
"franzferdinand" – goes direct to to "Franz Ferdinand" homepage
"Franz c" – gives long list of Artists, many starting with "Franz", but with Franz Ferdinand at top (probably due to popularity being taken into account?). Interestingly one of the surnames starts with the "c" ("Franz Capone").
"Franz ferdinands" –gives much shorter list with "Franz Ferdinand" at the top.
"franz ferfdg" – returns no results
"franz ci" – returns no results
"f Ferdinand" – returns no results

Napster Comparison 4: "Everything But the Girl"
"everything but the" – returns results with "Everything But the Girl"
"everything but t" – returns results with "Everything But the Girl"
"but t" – returns results with "Everything But the Girl"
"butth" – Artist list containing only "The Butthole Surfers" !!!

Napster Comparison 5: "Björk" (*International character handling*)
"bjork" – returns results with "Björk"
"Björk" – returns results with "Björk"
"jöhn" – returns matches for "john" – this is good!!!

Napster Comparison 6: "1"
"1" returns the following list of Artists:
"KRS-One"
"Atomic #1"
and many more....

Napster Comparison 7: "2"
"2" returns the following list of Artists:
"Boyz II Men"
"Soul II Soul"
"Bad Boys 2"
"Face to Face"

Napster Comparison 8: "Kooks"
"The Kooks" returns "Kooks", therefore we should remove preceding "the_"'s from strings.

Napster Comparison 9: "Boys to Men"
"Boys to men" returns the following list of Artists:
"Boyz II Men"
"Boyz 2 Men"
"Boyz To Men"

This is very bad since they are all one and the same Artist! This is an area in where we should be seeking to improve on Napster's approach, although this is most definitely an issue of BAD DATA as opposed to a flaw in Napster's search algorithm.

Napster Comparison 9: "Martha and the Muffins"
The following should be treated as equivalent:

"Martha and the Muffins"
"Martha & the Muffins"

5.2 Basic Search Description for Artists

The following is a description of how we can emulate Napster's Artists search. The process is based on the real-life Napster use-cases described previously.

N.B. The same principles below will also work nicely for the Album, Track and Playlists searches described further on in this document.

1) **Transform** entered user string to a **user match string** following exactly the same rules as described in section 3.2.1. Perform the same match on the Artist AKA field (editorially controlled field).

2) Find instances of the **user match string** (e.g."3 LIONS" or FRANZ FERDINAND") **within** the Artist **match-keys**. Valid matches are "3 LIONS_%", "%_3 LIONS_%", and "%_3 LIONS".

3) If (1) and (2) returns only 1 match then go direct to Artist homepage (and Album Homepage for Albums, and the Now Playing screen for Tracks)

4) Otherwise list matches from 1), followed by matches from 2) ranked by popularity and then alphabetically.

5) If we get this far then: Attempt to make match the user string against the **approximate match key.** If we find an exact match then take the corresponding **match key** (not the approximate one) and run steps 1) thru 4) again (obviously skipping this stage after...). This approach ensures, for example, that if the person enters "sclub7" then we can find out what it really should be in terms of the match key, i.e "S CLUB 7" and perform the search again.

6) Look for matches where the user might have entered:

- The Album and Artist name in combination (where we are in ALBUM SEARCH mode). We do this by joining the match keys for both Album and Artist and comparing against the user search string. This will help us find examples of, say, "Revolver The Beatles". If first match attempt fails then we need to join match keys both ways round in order to accommodate the possibility of "The Beatles, Revolver "

- The Track and Artist name in combination (where we are in TRACK SEARCH mode). We do this by joining the match keys for both Track and Artist and comparing against the user search string. This will help us find examples of, say, "Yesterday The Beatles". If first match attempt fails then we need to join match keys both ways round in order to accommodate the possibility of "The Beatles, Yesterday"

- **N.B. There combinations are considered the most likely ones that a user would enter. We could also consider combinations of Track and Album name but I don't think it likely that the user would do this. We will monitor the situation.**

- **N.B. The system should be also able to handle situations where the user types in just "beatles" when searching for ALBUMS or TRACKS. The net result should be a list of Beatles Albums or Tracks respectively. This can be done using existing algorithms once we have sorted out the "beatles yesterday" and "yesterday beatles" situation.**

If we have found matches from the above then we leave the search routine. Otherwise we move on to **approximate matching**:

7) At this point we need to track the user search string in the DB with a marker of "NO EXACT MATCH FOUND". The MusicStation content controller can then look at this and, if necessary, modify the AKA tables to accommodate the search string. Often the case night be that we currently do not have content for that Track, Artist or Album. This information will also be very useful to the content controller.

   We then repeat steps 1) to 4) further converting the **user match string** to an **approximate user match string**, and comparing against the approximate match-keys. Return results by popularity then by most recent, and under the heading "**No exact matches found. Did you mean:**"

Very Important Notes:

- **It must be remembered the same track can occur on many different Albums. Since we only return Track name and then Artist Name in the result set for Tracks we need to filter out occurrences of exactly the same Track. One way of doing this might be to filter out Tracks with exactly the same name (or match key) and length (though I have seen the same track on different albums with slightly different lengths). The best way to approach this problem would be to use acoustic fingerprint matching when the files are imported in order to confirm that the track is a duplicate. We would then maintain a table to control these relationships. (N.B. Exactly the same problem will arise with recommendations – we do not want to recommend the same Track more than once).**

- Where popularity is equal, tracks should be ranked in Alpha order. We should also place Non-Classical before Classical in separate sections. The list of classical matches should be entitled "Classical matches". The main list does not need a title.

6. Searching Albums

The Album search should return the following:

Album Name – Artist Name (Year Of Release)

The process behind the search is the same as for Tracks, but instead working off the Album field, i.e. return exact matches first, ranked by popularity... followed by LEFTSTR wildcard matches ranked by popularity... followed by completely wild-carded "%_ONE_%" type entries

Notes:
DISCUSSION: We should consider using more than one line for this information or the user will have to scroll for practically every track returned to get album and artist info.

DISCUSSION: We need to consider how to represent explicit content. This is a wider issue than just for search results.

7. Searching Tracks

The Track search should return the following:

> **Track Name – Artist Name (Track Length)**
> and only return Tracks of the same name with lengths that are different.

DISCUSSION: We should consider using more than one line for this information or the user will have to scroll for practically every track returned to get Album and Artist info.

Return matches as per Albums.

8. Advanced Search

We should remove the 'Classical' and 'Popular' and 'Not Popular' from the Popularity menu. Perhaps even remove the 'Popularity' section altogether (unless it can be defined as a threshold number of users who have downloaded that Artist, Album , Track or Playlist).

In addition we should add the following drop-down list (based on feedback from GlobeTel).:

Type of results:
- Most relevant (default)
- Most recent
- Most popular
- Most rated
- You might like

8.1 Advanced Classical Search

This needs to be a new option on the right-hand menu of the search screen.

If we use the Muze meta-data model for classical music then it is likely that we will need to accommodate the following free-text search fields.

- Work Title:
- Album Title:
- Composer:
- Soloist/Performer:
- Conductor:
- Orchestra/Ensemble:
- Record Label:

OMNIfONE™

MusicStation Recommendations & Ratings

ver 1.1

16th November 2006

Chris Evans

Table of Contents

Introduction

This document describes a very feasible, and well-though-out, method for running and hosting a recommendations system for MusicStation.

Important to note, that not when explicitly stated, a **full-play** uses the same criteria as that used for subscription licensing with the content owners. It is expected to be a play of either a certain minimum number of seconds of a track or percentage of a track.

1. Recommendations

Supporting systems are required to support the following personalised customer recommendations:

- "More like this" Track, Album or Artist
- Tracks "You might like"
- Albums "You might like"
- Artists "You might like"
- Playlists "You might like"
- "Recommended Members" as listed on the Buzz Cool Members screen
- Recommended Playlists as listed on the Buzz Cool Playlists screen – is this the same list as Playlists you might like?
- "Find in Playlists?"
- Inbox – editorial and promotional

2. Supporting Logical Structures for Making Recommendations

We will have three main structures to support the making of these recommendations.

- Associated Tracks Matrix
- Associated Artists Matrix
- Associated Customers Matrix

We will discuss the physical infrastructure of systems in a later section. For the moment it is enough to consider that these structures will be frequently refreshed, perhaps every 24 hours.

2.1 Supporting Structure 1 - Associated Tracks Matrix

The Associated Tracks Matrix is a matrix of correlations representing how strongly associated pairs of Tracks are in the system, based on ratings, and customer plays.

2.1.1 Stage 1 - Produce counts of Track associations

For Tracks we will build a matrix like the one above, representing:

Counts of customers who have either/or fully played, or have rated as **Love It!**, the Tracks in the pair.

	a						
	Track1	**Track2**	**Track3**	**Track4**	**Track5**	**Number of Correlations**	
Track1		12	0	23	78	3	
Track2	12		27	0	0	2	
Track3	0	27		5	0	2	
Track4	23	0	5		10	3	
Track5	78	0	15	10		3	
Number of Correlations	3	2	3	3	2		

Represents the number of customers who have fully listened to Track1 and Track2 at least twice.

Represents the total number of correlations for Track3 (I.e. non-zero cells).

Important Notes and Rules

The matrix above only considers a universe of 5 Tracks. We are likely to be considering 500,000 for go-live.

In order to be included as a count in 1), the user in question must have listened fully (as defined by the licensing agreements) AT LEAST TWICE. The rationale behind this is that, if a customer listens to a Track more than once, then they probably like it. If they only listen to the Track once then they may only be exploring new music, but not be impressed enough to ever go back to it.

If a customer rates two Track pairs highly, and listens to both more that twice, then this will have the effect of adding 2 to the corresponding intercept in the matrix. This is the maximum influence that one user can ever have on a Track intercept pair.

A Track that has been rated as **Love It!**, but never played, still counts towards an association.

This matrix covers all Tracks, and all ratings and plays, across all services, within the global MusicStation offering. The same applies to the Artists Associations Matrix described further on.

You will note that half the matrix is duplicated across the diagonal. Therefore, in theory, only half of the matrix is needed.

2.1.2 Stage 2 - Weight the Track associations

We now need to take the matrix from Stage 1 and apply weightings and produce correlations that take account of the fact that some Tracks might just simply be popular to ALL customers (and hence are not necessarily highly correlated for individual associated pairs).

The formula that we apply to do this is known as a TF•IDF formula.

A description of how the TF•IDF formula works, in the context of keywords belonging to a document or web search, is outlined here:

TF = Term Frequency

A measure of how often a term is found in a collection of documents. TF is combined with inverse document frequency (IDF) as a means of determining which documents are most relevant to a query. TF is sometimes also used to measure how often a word appears in a specific document.

IDF = inverse document frequency

A measure of how rare a term is in a collection, calculated by total collection size divided by the number of documents containing the term. Very common terms ("the", "and" etc.) will have a very low IDF and are therefore often excluded from search results. These low IDF words are commonly referred to as "stop words".

$$Weighting = frequency \times \log_2\left(\frac{1}{p(T_1)p(T_2)}\right)^3.$$

Notes on this equation:

- The TF = *frequency* (or the intercept value in the Stage 1 matrix).
- The IDF is represented by the latter (*log*) part of the equation, and is a *base-2 logarithm.*
- $P(T_1)$ represents the overall probability of Track 1 appearing at least once in the different pairings in the matrix (i.e. it is simply how many times it occurs at least once in a pairing, divided by the total number of Tracks).
- The IDF is raised to the power of 3. This is not a fixed constant, but is something that can be experimented with in order to refine the recommendations. A well-known online music-recommender uses the value of 3 for this constant, and so we would be wise to follow their knowledge and lead.

As an example of the equation's use, if we wish to calculate a weighting for **Track 1** and **Track 2** from the Stage 1 matrix, then we would perform the following calculation

$$Weighting(T_1, T_2) = 12 \times \log_2 \left(\frac{1}{\frac{3}{4} \times \frac{2}{4}} \right)^3$$

This gives a weighting for **Track 1** and **Track 2** of **34.**We can now produce a new Weightings Matrix, including the sum of all the weightings at the end of each row and column: .

		a							
		Track1	**Track2**	**Track3**	**Track4**	**Track5**	**Weighting Sum**		
b	**Track1**		34.00	0.00	13.15	44.61	91.77		
	Track2	34.00		76.50	0.00	0.00	110.50		
	Track3	0.00	76.50		2.86	8.58	87.94		
	Track4	13.15	0.00	2.86		5.72	21.73		
	Track5	44.61	0.00	8.58	5.72		58.91		
	Weighting Sum	91.77	110.50	87.94	21.73	58.91			

2.1.3 Stage 3 - Normalize the weightings

We now need to normalize the weightings. Essentially all this means is that we create a new matrix where every weighted correlation in the matrix is divided by the overall sum for the correlations in that row or column.

Using the example of **Track 1** and **Track 2** again, we would simply **divide 34 by 110.5**, providing a **normalised weighting of 0.31**.

The result of this is that we now have a set of normalized weightings lying between 0 and 1:

		a						
		Track1	**Track2**	**Track3**	**Track4**	**Track5**		
b	**Track1**		0.31	0.00	0.61	0.76	PCAs	
	Track2	0.31		0.87	0.00	0.00		
	Track3	0.00	0.87		0.13	0.15		
	Track4	0.61	0.00	0.13		0.10		
	Track5	0.76	0.00	0.15	0.10			

In the resulting table, the nearer the value is to 1, then the higher the correlation between the Tracks.

In the world of recommendations, the values in the table are now called **Pre-Computed Associations (PCAs)**, by virtue of the fact that they are correlations, at that they are reproduced on a regular basis (but generally not updated in an ongoing manner due to the amount of number crunching involved).

An example of the Tracks matrix in action can be seen in the attached active Excel worksheet:

2.2 Supporting Structure 2 - Associated Artists Matrix

The Associated Artists Matrix is a matrix of correlations representing how strongly associated pairs of Artists are in the system, based on ratings, and customer plays.

a

b		Artist1	Artist2	Artist3	Artist4	Artist5
	Artist1		0.31	0.00	0.61	0.76
	Artist2	0.31		0.87	0.00	0.00
	Artist3	0.00	0.87		0.13	0.15
	Artist4	0.61	0.00	0.13		0.10
	Artist5	0.76	0.00	0.15	0.10	

The Associated Artists Matrix of PCAs will essentially be built in exactly the same way as that for Tracks.

The criteria for inclusion in the Artist Plays Matrix is that the customer must have fully played at least one track from that Artist at least twice. Again, the maximum influence a single customer can have on the matrix is a an additional value of 2 (in the instance where they have both rated a pair of Artists as **Love It!** And have fully listened to at least one Track from both Artists at least twice.

N.B. Ratings for Tracks or Albums by this Artist have no influence on the Associated Artists Matrix.

2.3 Supporting Structure 3 - Associated Customers Matrix

The Associated Customers Matrix is a matrix of correlations representing how strongly associated pairs of Customers are in the system, based on ratings, and customer plays.

		a				
		Cust1	**Cust2**	**Cust3**	**Cust4**	**Cust5**
	Cust1		0.31	0.00	0.61	0.76
	Cust2	0.31		0.87	0.00	0.00
b	**Cust3**	0.00	0.87		0.13	0.15
	Cust4	0.61	0.00	0.13		0.10
	Cust5	0.76	0.00	0.15	0.10	

The Associated Customers Matrix of PCAs can be built as part of the same process for generating the Associated Artists matrix.

The criteria for inclusion in the Associated Customers Matrix is that the customer must have fully played at least one track from the same Artist* at least twice. Again, the maximum influence a single customer can have on the matrix is a an additional value of 2 (in the instance where they have both rated THE SAME pair of Artists as **Love It!**, and have fully listened to at least one Track from both Artists at least twice.

- N.B. Choosing common Artists here is likely to be beneficial over choosing common Tracks since the implications for calculations and processing power will be lowered.

3. Making Recommendations

This section describes how the described structures are used to generate recommendations fro:

- "More like this" Track, Album or Artist
- Tracks "You might like"
- Albums "You might like"
- Artists "You might like"
- Playlists "You might like"
- "Recommended Members" as listed on the Buzz Cool Members screen
- Recommended Playlists as listed on the Buzz Cool Playlists screen – is this the same list as Playlists you might like?
- "Find in Playlists?"
- Inbox – editorial and promotional

All the functionality described runs at run-time on a per-request basis*, based upon the calculated PCAs.

*I.e. We are not calculating recommendations for all customers. We only produce them when requested from the PCAs.

Functionality	Description	Associations Matrix based on?	Input(s) to recommendation process	Results mechanism*
"More like this" Track	In the **More like this** scenarios, a seed Track, Album or Artist is selected by the customer. MusicStation then provides a sequence of Tracks, Albums or Artists based on the seed, which can optionally be used as a Playlist by the customer.	Track Associations Matrix	The seed Track	Sequence of 10 distinct recommended Tracks in descending of order of closeness of fit (i.e. PCA correlation value). Filtered out of this returned list are: ▪ The seed Track. ▪ Tracks that are not available on the customer's service. ▪ Tracks that the customer already owns or has fully listened to. ▪ Tracks, or Tracks from Albums or Artists, that the customer has rated as **Hate it!** Returned Tracks should be from a variety of Artists. No more than 2 Tracks should be for the same Artist.
"More like this" Artist	Ditto above.	Artist Associations Matrix	The seed Artist.	Sequence of 10 distinct recommended Artists in descending of order of closeness of fit (i.e. PCA correlation value). Filtered out of this returned list are: ▪ The seed Artist ▪ Artists that are not available on the customer's service. ▪ Artists for which the customer has already fully listened to 50% of their catalogue. ▪ Artists that the customer has rated as **Hate it!**
"More like this" Album	Ditto.	Artist** Associations Matrix ***N.B. this solution avoids the creation of an*	The seed Artist of the Release.	Sequence of 10 distinct recommended Albums (Releases), based on a cross-section of the Albums produced by the Artists with the highest closeness of fit (i.e. PCA correlation value). Filtered out of this returned list are: ▪ The seed Album. ▪ Albums that are not available on the customer's

		Album Associations Matrix, though we should consider pros and cons.		service. ▪ Albums for which the customer has already fully listened to at least 50% of the Tracks. ▪ Albums that the customer has rated as **Hate it!**, and Albums from Artists that the customer has rated as **Hate it!** Returned Albums should be from a variety of Artists. No more than 2 Albums should be for the same Artist.
Tracks "You might like"	Tracks, Albums, Artists, and Playlists that "You might like" is a sequence of 10 recommended Tracks, Albums, Artists, and Playlists that are based upon your recent listening habits. The list is available form the Artist/Album/Track/Playlist screens.	Track Associations Matrix	10 distinct Tracks made up of: ▪ The most recent Tracks rated a **Love It!** (up to a maximum of 5) ▪ The most recent Tracks that have been fully-listened to at least twice, to make up the remaining Tracks. Excluded as inputs are Tracks the customer had flagged as **Hate It!** Regardless of how many times listened to.	Sequence of 10 distinct recommended Tracks in descending of order of closeness of fit (i.e. PCA correlation value). Filtered out of this returned list are: ▪ The seed Tracks. ▪ Tracks that are not available on the customer's service. ▪ Tracks that the customer already owns or has fully listened to. ▪ Tracks, or Tracks from Albums or Artists, that the customer has rated as **Hate it!** Additionally the system will seek to return the best correlated two new Tracks that are one-week old new releases in the system, replacing the least highly correlated Tracks in the 10 returned. Returned Tracks should be from a variety of Artists. No more than 2 Tracks should be for the same Artist.
Artists "You might like"	Ditto above.	Artist Associations Matrix	10 distinct Artists made up of: ▪ The most recent Artists rated a **Love It!** (up to a maximum of 5) ▪ The most recent Artists for which Tracks have been fully-listened to at least twice, to make up the remaining Artists. Excluded as inputs are Artists the	Sequence of 10 distinct recommended Artists in descending of order of closeness of fit (i.e. PCA correlation value). Filtered out of this returned list are: ▪ The seed Artists. ▪ Artists that are not available on the customer's service. ▪ Artists for which the customer has already fully listened to 50% of their catalogue.

			customer had flagged as **Hate It!** Regardless of how many times listened to.	▪ Artists that the customer has rated as **Hate it!** Additionally the system will seek to return the best correlated two new Artists that are one-week old new releases in the system, replacing the least highly correlated Artists in the 10 returned.
Albums "You might like"	Ditto.	Artist Associations Matrix	Ditto above.	Sequence of 10 distinct recommended Albums in descending of order of closeness of fit (i.e. PCA correlation value). Filtered out of this returned list are: ▪ The seed Albums. ▪ Albums that are not available on the customer's service. ▪ Albums for which the customer has already fully listened to at least 50% of the Tracks. ▪ Albums that the customer has rated as **Hate it!**, and Albums from Artists that the customer has rated as **Hate it!** Additionally the system will seek to return the best correlated two Artists that are one-week old new releases in the system, replacing the least highly correlated Artists in the 10 returned. Returned Albums should be from a variety of Artists. No more than 2 Albums should be for the same Artist.
"Recommended Members"	Recommended members are customers that are similar to the source customer based upon rating and listening habits. They are listed on the Cool Members screen on the Buzz tab.	Customer Associations Matrix	The current customer.	Sequence of 10 distinct recommended Customers in descending of order of closeness of fit for the customer's service. Filtered out of this returned list are: ▪ Customers who are already friends. ▪ Customers who have been blocked. Additionally the system will seek to return the best correlated two new Customers that are one-week old new releases in the system, replacing the least highly correlated Customers in the 10 returned.

Playlists "You might like"	Playlists "You might like" are Playlists that have been shared by other MusicStation customers within the same service. Accessed from the Playlists screen and also listed on the Buzz tab Cool Playlists screen.	Customers Associations Matrix	The current customer.	Sequence of 10 distinct recommended Playlists taken from the customers who are closest to the current customer, in descending of order of closeness of fit. Filtered out of this returned list are: ▪ Playlists from customers who are already friends. ▪ Playlists from customers who have been blocked. ▪ Playlists that the customer has already listened to at least once. ▪ Playlists for which the customer has already fully listened to at least 50% of the Tracks. ▪ Playlists that the customer has in their library. ▪ Playlists that the customer has rated as **Hate it!** Additionally the system will seek to return the best correlated two new Customers that are one-week old new releases in the system, replacing the least highly
Inbox – editorial and promotional.	Inbox items are directed to the customer based on what they listen to. An extension of this would be to supply news on Artists that we THINK they may be interested in based upon our Artists correlations matrix.	Artist Associations Matrix	As per "Artists "You might like".	As per "Artists "You might like".

*** in all cases, if no recommendations are available (e.g. all correlations are zero), then simply return the most popular entities (but conforming to the post-filters like not including Hate-it! Material.)**

4. Supporting Infrastructure for Recommendations

Since the Track PCA matrix will be by far the biggest (remember the Customer Associations Matrix is on a per-service level, and likely to be spread across different servers), we can take the Track Associations Matrix as an example we can get an idea of the amount of storage required to accommodate our PCA structures.

Assuming that we have 500, 000 Tracks, and are using a 16-bit 4-decimal place* floating-point representation for each PCA (could be 10-bit id the underlying stack allows this), then the total number of PCAs required to store is:

$5 \times 10^5 \times 5 \times 10^5 = 25 \times 10^{10}$ correlations.

However, since the matrix is duplicated across the diagonal, we can halve this giving:

12.5×10^{10} correlations.

Since each PCA takes 2 bytes to store then the total memory required is:

$2 \times 12.5 \times 10^{10} = 25 \times 10^{10}$ bytes.

Or approximately 240 GB.

**more decimal places may be required since some of these correlations could be <> 0 but still very small. We need to set up a test.*

Notes

If an 8-bit floating-point representation was used then we could halve the memory requirement (though we would loose accuracy)

With a **million Tracks** the implication for storage is almost up to **1 Terabyte.**

Refer to section 4.4 for more discussion on how space can be saved.

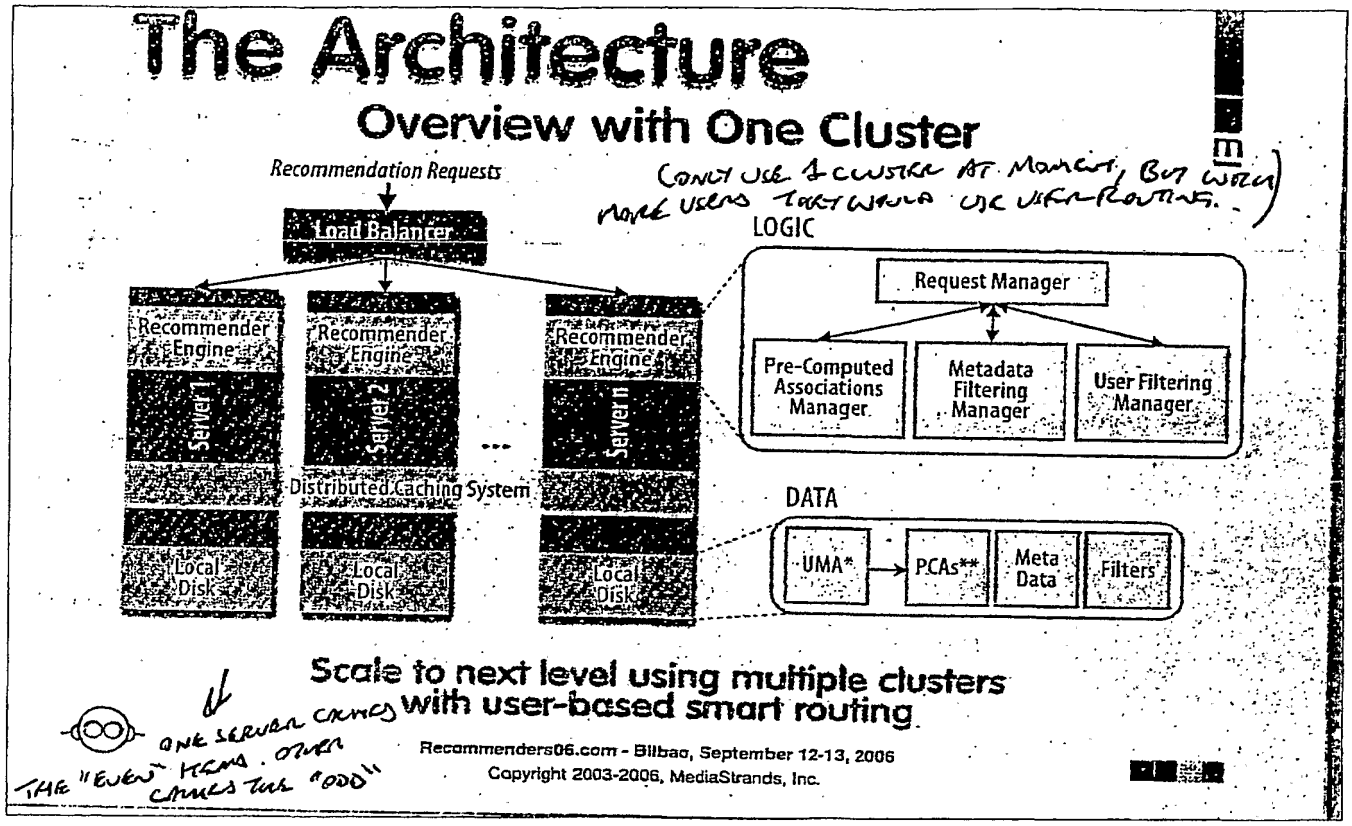
4.1 Architecture

The following is recommended as a minimum:

- **PCA generation server.** Creates and stores the PCA matrices. Is effectively a dumb, but powerful server, with plenty of disk capacity.
- **Recommendation broker.** Requests and responds to recommendation requests. Contains the intelligence to build the recommendation set based on the PCA tables on the PCA server. The PCA matrices could sit on this server once created, or alternatively be located on another server.

4.2 Sample architecture from MyStrands

MyStrands.com is an online community –based music recommendation service. What is illustrated below is a basic representation of the physical architecture of their recommendations system. The system runs on Apache / Tomcat / Servlets.



N.B. My two scribbled comments from the presentation:

> "[MyStrands] only use 1 cluster at [the] moment, but with more users they would use user-routing"'
>
> "One server caches the 'even items'. Another caches the 'odd'",

4.3 Update frequency

Two recommended approaches, either:

1) The PCA matrices be totally re-generated from raw-data every 24 hours, or
2) The Stage 1 matrices are maintained and updated in real-time, and blown into the PCA matrices at regular intervals.

Approach 2) may take less time and be more efficient, though it does rely on the Stage 1 data always being accurately maintained.

MyStrands approach is to perform "continuous synchronous updating of PCAs via harvest process"

4.4 Storage

Should we store the PCA matrices in a DB? Or is a flat-file good enough? Interestingly MyStrands use the latter, but do recognise that a DB could be used.

Whatever physical storage mechanism is used, the likely structure will be:

Track1_ID	Track2_ID	PCA
12345	12346	0.0023
12345	12347	0.2040
12345	12348	0.0002
12345	12349	0.0001

IMPORTANT: We can save space by not storing PCAs that are equal to 0. Basically, if there is no association of two Tracks in the table, then the PCA will be assumed to be 0.

4.5 Caching

Consideration should be given as to cache intelligently – for example; MyStrands find that just keeping the top 250,000 most-recently-used PCAs in memory still provides a 93% hit-rate from customer requests.

4.6 Distribution of recommendation servers across MusicStation services

Much to be discussed here. More info required from Phil.

5. Solving Cold Start Issue

At initial go-live we will have no usage or rating date with which to compute PCAs. This section seeks to address this issue.

5.1 Incorporating Muze data

Muze can supply information linking related Artists (coverage of 1,400 Artists). Additionally they can supply Sub-Genre information for many Tracks.

To solve our cold-start issue we can create an initial set of PCA matrices in which we have placed associations based on the Muze data.

For example, for the **Artist Associations Matrix**, we can simply insert an initial starter-value of 10 into the Stage 1 creation process for all Artists that Muze tell us are related, and a value of 5 if they share the same Sub-genre.

Similarly for the **Track Associations Matrix**, we can simply insert an initial starter-value of 10 into the Stage 1 creation process for all Tracks by Artists that Muze tell us are related, and a value of 5 if they share the same Sub-genre.

For the **Customer Associations Matrix**, we can simply insert an initial starter-value of 10 into the Stage 1 creation process for all Tracks by Artists that Muze tell us are related, and a value of 5 if they share the same Sub-genre.

Should we consider how to incorporate NEW Muze association later on? Personally I think we do not need to be concerned about this as such associations should creep into the system anyway.

5.2 Audioscrobbler and similar

We can use related Artist and Track info from Last.FM (and similar services) to pre-populate the Stage 1 PCA matrices in a similar way to that described for the Muze data (e.g. by adding another 10 to the correlated value).

5.3 How we present recommendations on first use of MusicStation

When a customer first uses MusicStation there will be no usage or rating data available for that customer to base recommendations on. There are tow options to address this:

1) Display a message to the customer in the "You might like" sections explaining something like **"Rate or listen to some music and MusicStation will recommend other Artists/Albums/Tacks/Playlists/Members that you might like."**

2) Because, the system always returns the most popular entities as defaults when no other customer input data is available (refer to the starred comment after the table in section 3), the system will simply return the 10 most popular entities (i.e. Tracks/ Artists / Albums etc,).

If we decide to go with 2) then we would need to ensure that we have set up some initial popularity data in the database so that the very first MusicStation users receive some recommendations.

My preference, however, is to use the approach in 1), since:

1) First impressions last, and customers might be put off when being presented with recommendations that are blatantly of the mark.
2) It is a good introduction to the customer on how the "You might like" sections work.

6. 'Definites' for Consideration (not just *nice-to-haves!*)

6.1 Randomizing output to allow for refresh of recommendations

If we randomized the output of the recommendations system somewhat, then we could allow for the customer to request a new set of "You might like" recommendations.

For example, the recommendation system internally could actually return 100 entities, of which 10 are randomly chosen for return back to the client.

6.2 Keeping recommendations current

In order to keep recommendations current (i.e so that they shift over time with customers' tastes), it would be a good idea to keep 2 sets of PCA matrices being updated concurrently, with the second set of matrices being, for example, staggered 1 month behind the first in terms of the data used. At a certain point (say once a month) the reserve matrix could be switched into 'live', ensuring that fresh associations are available based on current trends. At the same time we would begin calculating PCAs for a new reserve table.

6.3 Filtering recommendations

It would be useful of recommendations could be post-filtered by Era, Genre, Rating and Mood (if available). Stevie G, you may have some input here.

6.4 Moods

It would be a god idea to allow customers, or editorial personnel, to associate Artists, Albums, Tracks or Playlists with a pre-defined set of moods. These moods could then be used as the basis for making recommendations (e.g. show me Happy music that I might like), and for post-filtering the results (as described in the previous section. This functionality would be a good v1 for Tags.

6.5 Supporting Structure 4 - Associated Web-Artists Matrix

A duplicate structure as that described for the Associated Artists Matrix ("Associated Web-Artists Matrix" could be built from information crawled from the Internet.

Whenever 2 Artists are found on the same page, then we could assume that this is a positive association.

6.6 Explaining recommendations

From conferences I have been to, customers like to gain an understanding of how recommendations have been created for them. For this reason we could have a menu option similar to "How did I get these?"

7. Generating Starred Ratings

This section explains how we generate the 5-star ratings for Artists/Albums/Tracks/Playlists.

7.1 Inputs to the rating system

There will be two inputs to the star-ratings system - **explicit ratings** (i.e. **Love It!** and **Hate it!**), and **implicit ratings** (i.e. number of listens to Artists / Albums / Tracks, specifically the number of times a customer has **fully-listened** to that Artist / Album or Track, and **at least twice**).

It is recommended that, where possible, the ratings be mad up of a 50/50 split of **explicit** and **implicit** measures.*

* This will also have the advantage that customers cannot simply abusively rate stuff to get it to appear with a higher or lower star rating.

7.2 Calculating the 5-star rating for Artists/Albums/Tracks/Playlists

7.2.1 Calculating the explicit rating value

The explicit rating for an Artist/Album/Track/Playlist is simply based upon the proportions of customers who rated the Artist/Album/Track as **Love It!** against those who rated it as **Hate It!**. It is calculated as follows:

1) Take the number of customers who have rated the Artist/Album/Track/Playlist as **Love It!.**
2) Divide the value in (1) by the overall number of customers who have rated the Artist/Album/Track/Playlist (i.e. either as **Love It!** or **Hate It!**)
3) Multiply by 5 to provide a rating value out of 5.

For example, consider that for **Angels - Robbie Williams**, we have **45 Love It!** ratings and **18 Hate It!** ratings. The rating value is then:

$$Rating\_value = \left(\frac{45}{45+18}\right) \times 5 = 3.57$$

7.2.2 Adjusting the rating value to handle low number of ratings

I order to avoid abuse, and to prevent lots of 0 or 5 star ratings appearing in the system in situations where only a few customers have rated an Artist/Album/Track/Playlist, we should always include two phantom ratings of **Love it!** and **HateIt!** in the calculation. Thus the final calculation becomes:

$$Rating\_value = \left(\frac{45+1}{(45+1)+(18+1)}\right) \times 5 = 3.53$$

7.2.3 Calculating the implicit rating value

For calculating the implicit rating value we need to create a baseline for comparison.

The most sensible baseline is one that represents the average number of plays per customer for all Artists/Albums/Tracks/Playlists that **have been fully played at least once by each individual customer** (i.e. it is not fair to include Artists/Albums/Tracks/Playlists that have never been listened to within the calculation). We can that take this baseline to represent a **2.5 rating** within the system, and adjust all other ratings up or down accordingly by normalising the distribution to around the **2.5 rating value.**

As an example, if the average* number of plays per customer for the Track: **Angels - Robbie Williams is 12.90**, and the average number of plays for all Tracks (that have had at leas one full play) per customer is **4.66**, with a standard deviation of 4.23, then we would do the following:

Average plays per customer for **Angels - Robbie Williams = 12.90**

Normalized plays (around a mean of 0) = (AV. PLAYS – OVERALL AV. PLAYS) / (STDEV)

Therefore, **normalized plays (around a mean of 0) = (12.90 – 4.66) / 4.23 = 1.95**

Therefore, **normalized plays (around a mean of 2.5 stars) = 2.5 + 1.95 = 4.45**

(N.B. It is feasible that, in very extreme circumstances, this value could be < 0, or > 5. In this case we will cap the value at 0 or 5 accordingly)

The overall representation of how this works in a universe of 6 Tracks is represented below.

	Average plays per customer	Normalized Plays (X - MEAN) / STDEV	Rating Value (2.5 + NORMALISED PLAYS)
Angels - Robbie Williams	12.90	1.95	**4.45**
Country House - Blur	4.60	-0.01	**2.49**
Life on Mars - David Bowie	3.30	-0.32	**2.18**
Yellow - Coldplay	1.23	-0.81	**1.69**
Bohemian Rhapsody - Queen	4.01	-0.15	**2.35**
I Luv Ya - Atomic Kitten	1.89	-0.65	**1.85**
Average overall plays per customer	**4.66**		
Standard Deviation	**4.23**		

* N.B. Use the MEAN average initially, but we should also experiment with the MEDIAN average since the latter will have the effect of removing the influence of individual customers who just play one Artist/Album/Track/Playlist in an obsessive manner (!)

7.2.4 Calculating the overall rating value

The overall 5-Star rating is calculated by simply taking the average of the implicit and explicit ratings, and rounding up to the nearest half star (round up since we want to be positive in what we present!).

Thus the overall rating for **Angels - Robbie Williams** = (3.53 + 4.45) / 2 = **3.99**

Therefore **Angels - Robbie Williams** receives a **4-star rating.**

7.3 Calculating ratings for Customers

The ratings for customers will be based upon the ratings and number of listens that a customer has had to their shared Playlists. It is calculated in a similar manner to that described in section 7.2, and likewise, for the implicit part, only considers Playlists that have been **listened to by other customers and at least twice.**

Once we have the overall ratings for all the customer's playlists then we will simply take an average of all of them to produce a final rating (5 star or other more desirable representation).

7.4 Ratings and the cold start

At go-live, or when any new Artists/Albums/Tracks/Playlists/Customers come into the system, that their **initial rating defaults to 3**. Additionally we will have editorial tools that will allow us to increase or decrease this value for certain Artists/Albums/Tracks/Playlists/Customers prior to go-live, or when new Artists/Albums/Tracks/Playlists/Customers are entered into the system.

Appendix A - Database Fields Used

The following CUSTOMER-CENTRIC database fields will be used to implement the above logic.

Table: **CUSTOMER_TRACK**

Field	Example Recommendation Use(s)
rating	REC INPUT: Customers loves this track. REC OUTPUT: Filter out tracks that this customer hates. RATING CALC: Explicit rating calculation.
play_count	REC INPUT: Only input track if play_count >= 2 (assuming that the play recorded is a FULL play as per the definition). N.B. The Track play_count is used as an input to all of Track, Album and Artists recommendations. REC OUTPUT: Filter out tracks that this customer has already fully played (>= 1). REC BUILD: Correlate 2 tracks in Stage 1 track recommendation matrix provided both tracks have play_count >= 2. Similarly the Artist and Customer Stage 1 matrices are based upon this track field. RATING CALC: Implicit rating calculation.
purchase_count	REC OUTPUT: Filter out tracks that this customer already owns.
last_play_date	REC INPUT: Select the last 10 tracks that have been fully-played at least twice by distinct artists.

Table: **CUSTOMER_RELEASE**

Field	Example Recommendation Use(s)
rating	REC INPUT: Customer loves this album. REC OUTPUT: Filter out albums that this customer hates. RATING CALC: Explicit rating calculation.
play_count	REC OUTPUT: Filter out albums or playlists for which this customer has already listened to >= 50% of the tracks. REC OUTPUT: Filter out playlists that the customer has fully listened to at least once. RATING CALC: Implicit rating calculation.
purchase_count	REC OUTPUT: Filter out albums for which this customer has already owns>= 50% of the tracks.

Table: **CUSTOMER_ARTIST**

Field	Example Recommendation Use(s)
rating	REC INPUT: Customer loves this artist. OUTPUT: Filter out artists that this customer hates. RATING CALC: Explicit rating calculation.
play_count	REC INPUT: Only input artists for which there are tracks with play_count >= 2 (assuming that the play recorded is a FULL play as per the definition). N.B. Inputted artists scenario applies for both recommended artists and albums. REC OUTPUT: Filter out albums for which this customer has already listened to >= 50% of the tracks. RATING CALC: Implicit rating calculation.

Important Notes

1) Other non-CUSTOMER_CENTRIC fields will need to be referenced in order to complete the implementation (e.g. fields that identify the release date of a new Track or Album, so that new releases can be returned as part of the recommendation mix).

2) We also assume that, in order to recorded in these tables, a play for a track represents a FULL play of that track, according to the definition used for payment according to subscription licensing (e.g. this is currently spoken of as being the first 30 seconds of the track). If this is NOT the case then other fields will need to be referenced, and at a deeper level than the information that these tables can supply (e.g. the start and end time for each track play).

Client Exception Handling

Mark Sullivan – 3 Jan 2007

Table of Contents

1. Introduction

The MusicStation client is regularly downloading and updating files in the background whilst the customer is using the application. When an error occurs we may want to retry, inform the user or do nothing depending on the task that is being performed and the error that was thrown. This document describes how we will decide what action to take when an error occurs.

2. Exception Listener

There are 3 main threads that control the MusicStation client. The UI Thread handles all key presses, the Paint Thread handles all screen redraws and the Task Thread handles loading data. Exceptions can be thrown in any of these threads but they are always passed to the ExceptionListener exceptionThrown() method.

The ExceptionListener then decides how to handle the exception based on:

- The Exception that was thrown
- The Event that caused the Exception
- The priority of the Event
- The super class of the Exception

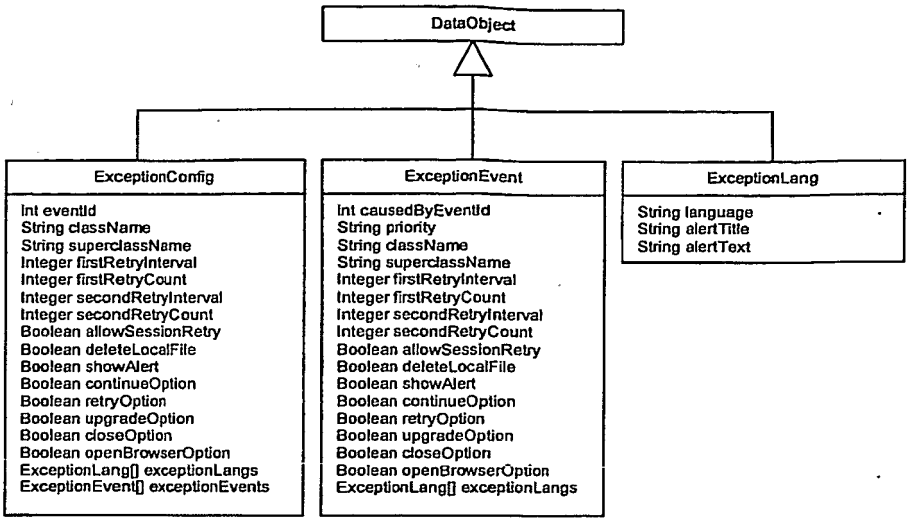
These parameters are used to lookup the ExceptionConfig for this exception. The ExceptionConfig contains all of the information needed to decide how to handle the exception.

3. Exception Config

The ExceptionConfig is used to decide whether to automatically retry the event that caused the exception or whether to show an error message to the user.

The following objects are used to configure exceptions:

- ExceptionConfig: Contains the default behaviour for this exception
- ExceptionEvent: Overrides the default behaviour for a specific Event and priority
- ExceptionLang: Contains the error messages in each language supported by the device



Only exceptions that occur in the Task thread cause a retry. The following attributes are used to decide whether and how to retry the Task:

- firstRetryInterval: We may want to initially retry the request quickly
- firstRetryCount: The number of times to retry or 0 to not retry
- secondRetryInterval: We may then want to back off and leave a longer period between retries
- secondRetryCount: The number of times to retry or 0 to not retry
- allowSessionRetry: If there is a server error or the file was not found we may want to disallow the same request to the server for this session.
- deleteLocalFile: If the file is corrupted we may want to delete the local file and retry loading the file from the server.

Any exception can display an error message to the user. The following attributes are used to decide whether and what to display to the user.

- showAlert: If true, displays an error message to the user with one or more options
- continueOption: Goes back to the last screen
- retryOption: Retries the Task
- upgradeOption: Installs a new version of the application
- closeOption: Closes the application
- openBrowserOption: Retries the request in the mobile's WAP browser
- exceptionLangs: The error message in each supported language

Only the Continue option has been implemented on the client

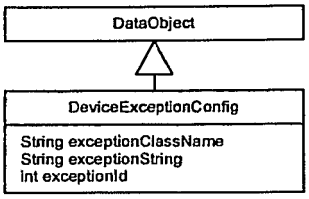
For any exception, these values can be overridden for a particular event or we can fall back to the values defined for the exception's superclass.

4. Device Specific Exceptions

Some devices do not throw the expected exceptions. For example the Nokia N70 throws an IOException with message "-34" when the server does not respond rather than the more specific ConnectionNotFoundException. The DeviceExceptionConfig object allows us to specify mappings between device specific exceptions and the expected exceptions.

The following fields map a device specific exception to and expected exception:

- exceptionClassName: The Exception thrown by the device
- exceptionString: The result of the Exception.toString() method
- exceptionId: The known Exception this maps to



5. Database Requirements

client_build

event_type_set_id	FK	number(10)	not NULL
exception_set_id	FK	number(10)	not NULL

The event and exception sets are created at build time and the indexes are used at runtime to map events and exceptions sent between the client and server.

event_type

priority		varchar(12)	DEFAULT NORMAL, in (MIN, NORMAL, MAX)

The priority is used to determine which events are sent from the client to the server first. Both the priority and severity level can be updated on the client by the server.

event_type_set

id	PK	number(10)	
automatic		number(1)	not NULL, default 0
count		number(12)	not NULL, default 0
guid		varchar(32)	not NULL
name		varchar(96)	not NULL
data_classification			
created			
inserted			
modified			

The set of events created for a client build. This set is used at runtime to map events sent by the client to event types in the database.

event_type_set_item

event_type_set_id	PK	number(10)	not NULL
event_type_id	PK	number(10)	not NULL
event_type_name		varchar(96)	not NULL

event_type_index		number(10)	not NULL, UNIQUE INDEX
data_classification			
created			
inserted			
modified			

The index is the mapping between a client event and an event type on the server. The index will be defined as a constant in the EventType data object. All references to an event in the client code will use this constant.

exception_set

id	PK	number(10)	
automatic		number(1)	not NULL, DEFAULT 0
count		number(12)	not NULL, DEFAULT 0
guid		varchar(32)	not NULL
name		varchar(96)	not NULL
data_classification			
created			
inserted			
modified			

The set of exceptions created for a client build. This set is used at runtime to map exceptions sent by the client to exceptions in the database.

exception_set_item

exception_set_id	PK	number(10)	not NULL
exception_id	PK	number(10)	not NULL
exception_name		varchar(96)	not NULL
exception_index		number(10)	not NULL, UNIQUE INDEX
data_classification			
created			
inserted			
modified			

The index is the mapping between a client exception and an exception on the server. The index will be defined as a constant in the ExceptionConfig data object. All references to an event in the client code will use this constant.

exception

id	PK	number(10)	not NULL
guid		varchar(32)	not NULL
name		varchar(96)	not NULL
event_type_id	FK	number(10)	not NULL
class_name		varchar(128)	not NULL
superclass_name		varchar(128)	
first_retry_interval		number(10)	
first_retry_count		number(10)	
second_retry_interval		number(10)	
second_retry_count		number(10)	
allow_session_retry		number(1)	
delete_local_file		number(1)	
show_alert		number(1)	
continue_option		number(1)	
retry_option		number(1)	
upgrade_option		number(1)	
close_option		number(1)	
open_browser_option		number(1)	
description		varchar (256)	
comments		varchar (256)	
data_classification			

created
inserted
modified

Contains fields that control how an exception that is thrown on the client is handled. The exception configuration is included in the JAR at build time and can be updated at runtime on the client by the server.

exception_lang

exception_id	PK	number(10)	not NULL
language	PK	varchar(2)	not NULL
alert_title		varchar(96)	
alert_message		varchar(256)	

MSullivan 4/4/07 9:24 AM
Comment: Is this large enough? The current largest value for a message is 177 chars

Contains language specific fields for exception handling.

exception_event

exception_id	PK	number(10)	
caused_by_event_type_id	PK	number(10)	
event_priority	PK	number(10)	in (ALL, MIN, NORMAL or MAX)
first_retry_interval		number(10)	
first_retry_count		number(10)	
second_retry_interval		number(10)	
second_retry_count		number(10)	
allow_session_retry		number(1)	
delete_local_file		number(1)	
show_alert		number(1)	
continue_option		number(1)	
retry_option		number(1)	
upgrade_option		number(1)	
close_option		number(1)	
open_browser_option		number(1)	
data_classification			
created			
inserted			
modified			

Exception handling can be overridden for specific events and specific event priorities.

exception_event_lang

exception_id	PK	number(10)	
caused_by_event_type_id	PK	number(10)	
event_priority	PK	number(10)	in (ALL, MIN, NORMAL or MAX)
language	PK	varchar(2)	
alert_title		varchar(96)	
alert_message		varchar(256)	

Localised fields overridden for specific events and specific event priorities.

device_exception

device_id	PK	number(10)	
exception_class_name	PK	varchar(128)	
exception_string	PK	varchar(256)	
exception_id	FK	number(10)	
automatic		number(1)	not NULL, DEFAULT 0
data_classification			
created			
inserted			
modified			

Maps device specific exceptions onto known exceptions. This table is populated by the detective during device commissioning.

OMNIfONE™

MusicStation

Client Overview Jan 07 update

Jan 2007

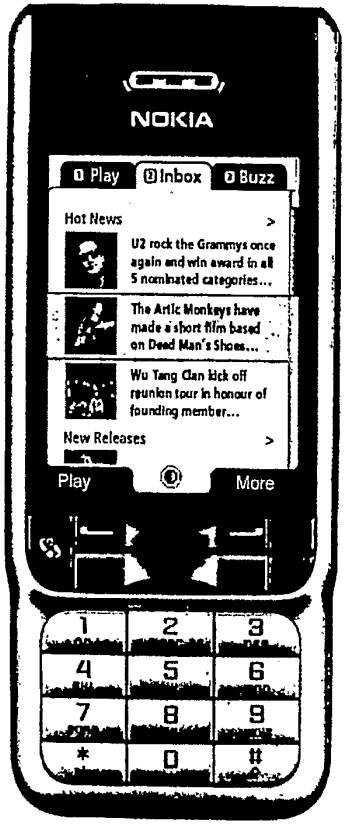
Table of Contents √

Tab Overview

Tab Overview

Potential Contents

- Video
- History
- Genre



Potential Contents

- Merchandise
- Artist/Genre News
- Artist Interviews
- Alerts
- Music Pics
- Editorial
- Did You Know?
- Ringtones/Master tones etc
- Prizes?? VIP tickets/backstage passes etc
- Gigs/Events/Festivals
- Tickets
- Biz
- News & Dates
- Gossip
- Hot News
- Inside Music

Potential Contents

- Friends (Mates too English?)
- What's Hot

OMNIFONE™

Intro: Using Tabs √ (#)

MusicStation splits its main functional concerns into tabs. Tabs are navigated using joystick left and right. Originally, it was considered to use the left & right joy actions, but there were problems. People have come to expect full control of the play screen's playing functions (prev, review, next, ffwd, vol up, vol down, play/pause) from the joystick. For this reason the left & right joy could not be used for tabs.

Numbered Tabs

Should the numbers be ever present? I had originally suggested that after we could tell they had got the hang of it, we could auto turn the numbers off (controlled via an **Options** item). Next time they came in it would look all nice (my main goal here is to make it look nicer by removing the numbers).

There's a suggestion that this is a bad idea if someone who has never used MusicStation before picks up the device of a friend and does know how to tab about.

We are going to try the interface with a graphic for the number.

Context on Tab Switching

When the user switches from one tab to another they had previously been on we could either:

- Display that new tab with the same screen and line item as the user was last on, on that tab
- Return to the user to the home screen of that tab and place the selection at the top of that screen.

For the Demo we have made this behaviour configurable through our configuration file to let people assess the two behaviours.

1
Single click shows left hand tab. Should holding 1 go to the top of the menu?

2
Single click shows left hand tab. Should holding 2 go to the top of the menu?

3
Single click shows left hand tab. Should holding 3 go to the top of the menu?

Multiple Tabs

We could support more than 3 tabs, though 3 only ever displayed at one time. 1, 2 & 3 are the only keys ever. However double click of three will move the tabs one to the left. And again etc. Holding 3 will take you to the far right tab. Holding 1 will take you to the far left tab. At any point pressing 1, 2 or 3 will show the left, middle or right tab respectively.

"Off Tab" Items

The 70. Now Playing screen is treated a little specially in that it can display on any tab, but it will not be considered as having context on that tab.

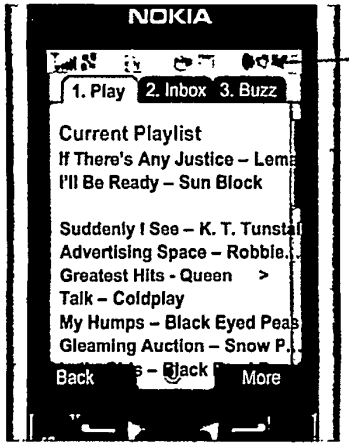
If you are on Tab 3 and playing a track then Now Playing screen will appear as if on tab 3. When you press [1] on this Now Playing screen, The last screen that you were in in tab 1 (other than Now Playing) should be the screen you go back to.

If the Play Tab was Split

- MyStuff/MyCollection
- MyMusic/My Top Stuff
- Find Music
- Play Now/Get More Music/Latest News
- Play Now!/Get More Music!/Latest News!

OMNIFONE

Intro: Screen General 1 √

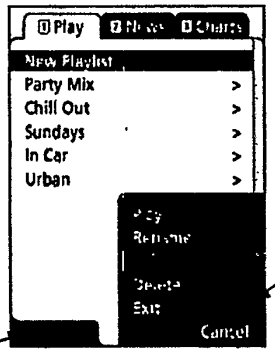


Phone Status Bar
Build the MusicStation app such that if in future it is possible to keep the phone status bar showing then that is possible.

Screen Titles
Each screen (beneath a main tab menu) has a screen title so as to give the user context of where they are. These titles are at the top of any information listed in the page. So that MusicStation can use the most space possible on a screen, these titles actually scroll with the contents of the screen. See below where scrolling down shows the title disappear up off the top of the screen.

Joystick Operation
Each screen design has a table like the one below to show what the joystick actions do. Here's the key to the symbols used.

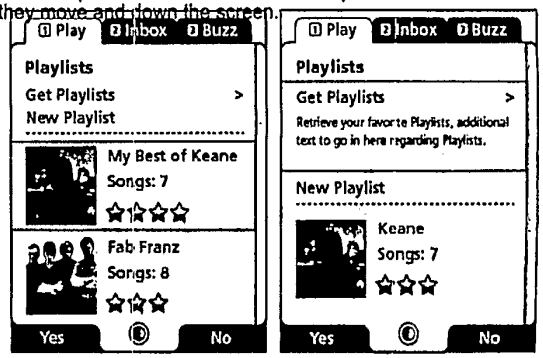
Navigation	
◀◀	Hold joy left
◀	Joy left
▲▲	Hold joy up
▲	Joy up
●	Joy in
●●	Hold joy in
▼	Joy down
▼▼	Hold joy down
▶	Joy right
▶▶	Hold joy right



This should be blank.

Softkey Operation
The softkey menu is accessible by pressing the More softkey. This pops up a softkey menu as shown left. The menu can be closed using the same softkey that launched the menu. This is shown with a Cancel command.

Picture Lists
Some screens like the Inbox use a list with a pictures on each item. On these screen, function options are displayed by default as single line items, but when highlighted expand to include a brief description. If we do not do this the user jumps from single line to multi-line items and the current line highlight changes size as they move and down the screen.



OMNIFONE™

Intro: Screen General 2 √

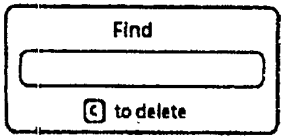
Horizontal Scrolling Extra Long List Items

List items which are too long should be displayed with an elipsis ... at the end.

When such an item is highlighted, the elipsis are removed and the text will scroll right to left. It will wrap around with a few spaces between the end of the string and the start of the string.

There will be a pause of 2 seconds before the scrolling starts.

Everytime the text has scrolled back around to the beginning again, there will be the same pause before it starts scrolling around again.



A text entry pop-up showing help for the delete key

Long List Scrolling

Pushing the joystick up or down and holding will scroll through the list quickly and gradually accelerate up to a higher speed. This will allow the user to mov e more easily through very long lists, while retaining control in shorter lists. There should be consideration given to having the length of the list control the rate of acceleration.

On reaching the top or end of the list, the selection will not automatically wrap around to the other end of the list. Only when the user explicitly pushes the joystick up or down at the start or end will the selection wrap around. This prevents the selection just zooming around and around screen.

Snapback to Now Playing

We will snap back to the Now Playing from any screen where there is no user activity.

Every screen will be assigned a time limit.

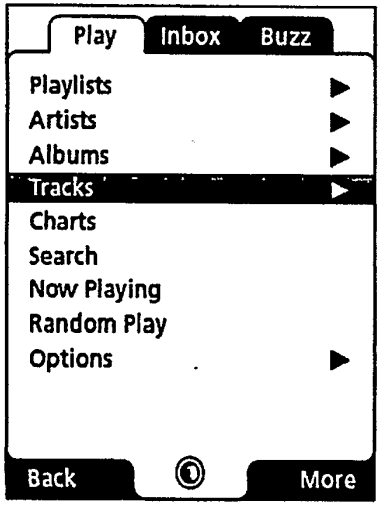
- 0 = no snap back to Now Playing
- Otherwise the count is the number of seconds to wait before snapping back to Now Playing
- When a track is playing and the user enters a screen the countdown will start
- Keypresses or joystick movement (including scrolling down) within the screen restart the timer
- At the end of this time the screen will snap back to the Now Playing screen, remaining in the current tab.

Specific rules for timings for each screen will be defined, but general rules are:

- Any screen within the Play tab should have a 16 second timeout
- Any screen in Inbox or Buzz should have a 1 minute timeout
- While a popup is being displayed there will be no snap-back. Popups are general modal and the user must deal with them.

OMNIFONE™

Intro: Screen General 3



New style solid arrows on lists

New style selection on picture lists

Dynamic Population of Lists

In the prototype we added rows to the screen as they became available, adding 1 item, then adding 2 more items, then adding 4 items, and then 8 items and so on. This causes the visible list to be populated in front of the user eyes.

We likely want to make this more intelligent so that the list is not displayed until there is sufficient data to fill the visible screen, and then the remaining off-screen data is added in the background.

Per Screen Help

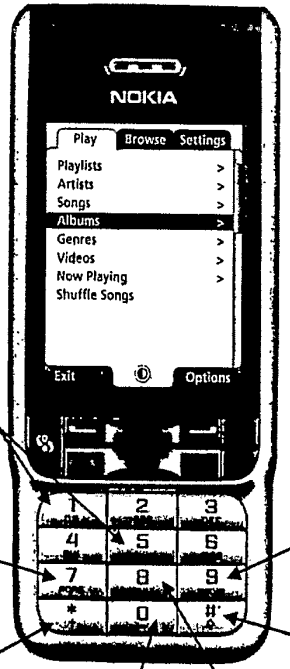
Consider the model used by the SE K800i when in camera mode. There if you press the 0 key it displays a popup with the whole keyboard shown with keys which have specific camera functions shown with an icon on them, and all other keys shown with their normal appeance.

On pressing a key the help picture disappears again and the function that was pressed is performed.

We would display such keyboard context sensitive help for every screen. There would also be an option on this help to see more hel p which would cause a WAP page of help content to be displayed in the browser.

Intro: Dedicated Keys √ (#)

These keys are nearly always available. Because they are hold functions (not just clicks) they are even active when the key they are on is available for use in a popup for example.
Now Playing doesn't work on a modal popup because holding 0 will deliver a 0 into the popup. The other two do.



[1] Display Help
Display the help popup.

[5] Volume Up
Volume up as if user had pushed joystick up on Now Playing screen

Previious Track
Previous track in playlist as if user had pushed joystick Left in Now Playing screen

Lock keypad.
When pressed locks the keypad. When held again unlocks the keypad. While the screen is locked and the user presses another key a popup appears to tell the user how to unlock the keypad

When the keyboard is locked a Locked icon will be displayed above the joystick.

Volume Down
Volume down as if user had pushed joystick down on Now Playing screen. If muted, changing the volume switches off mute

Pause/Play
Pause/Play as if user had pressed Joystick O on Now Playing screen

Next Track
Next track in playlist as if user had pushed joystick Right in Now Playing screen

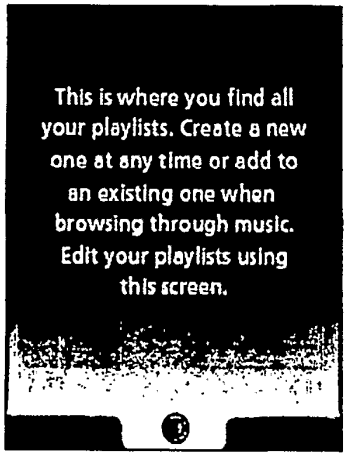
Close
(1) Single press while music playing – Displays the standard Close confirmation popup
(2) Press and hold while music playing - Closes MusicStation and keeps music playing
(3) Press while no music – Does nothing
(4) Press and Hold while no music playing – Exits MusicStation

Mute
Pressing this mutes the sound (but track still keeps playing). It is switched off again by the user either pressing Mute again or changing the volume.

Queries
- Do these functions and keys need to be abstracted and data driven by phone?
- On SEs the # is the mute
- Shall we also dedicate others such as 4/7 for page up/page down, 6/9 start/end of list, 5 for select
- Can double click be used for anything?
- Do we need a keylock?
- Should we graphically show when the app is muted?
- Is Keylock more important than mute & mute can be achieved by pause anyway?
- Should we dedicate 5 to pause/play?
- Should we dedicate 4 & 6 to prev/rewind/next/fast forward?

Intro: New Style Popups √ (#)

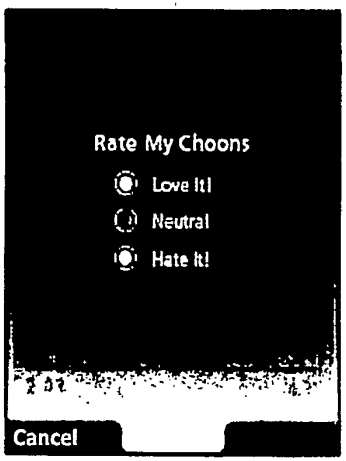
As of Oct 2006 we are introducing a new style full screen, semi-transparent popup screen. This screen type will be used in all cases throughout the interface where popups are displayed.



General information/help popup.

Text input popup.

General information popup.



Simple radio button popup.

More complex radio button popup.

OMNIFONE

Intro: Context Sensitive Menus 1/4

Like the right hand click on a PC, each type of object in the interface will have common right hand menu options which can be used to apply common actions to the object

Type of Object	Options on the more menu	Add to Playing	Play Next	Play ASAP	Action of the Middle Joystick Button
Highlighting a folder of objects (e.g. the folders in the Browse Playlists or Browse Artists menus)	Will have right hand menu options: Open, Close	N/a	N/a	N/a	Open
A Function in a list (e.g. New Playlist option or Search on the main menu)	Open, Close	N/a	N/a	N/a	Open

(* Enabled for owned playlists only; + Only displayed for a shared playlist; ++ Only displayed for a private playlist;
^ Enabled only when there are tracks in the Current Playlist; ~ Enabled only when the playlist is not empty)

Intro: Context Sensitive Menus 2/4

Like the right hand click on a PC, each type of object in the interface will have common right hand menu options which can be used to apply common actions to the object

Type of Object	Options on the more menu	Add to Playing	Play Next	Play ASAP	Action of the Middle Joystick Button
A Playlist in a list of playlists	Open Add to Playing~ Play Next~ Play ASAP~ Rate~ Delete* Rename* [Share* ++ ~ | Make Private* + ~] Send to Friend+% Who created? + Close	PPT: Prompt if want to buy *n* tracks (see screens 90-93) Display popup. See 4/4 for wording Add all the tracks in the playlist to the end of the current playlist whether downloaded or not.	PPT: Prompt if want to buy *n* tracks (see screens 90-93) Display popup. See 4/4 for wording Once the first track is ready add it after the current track & prefetch. Add the remaining tracks after this track.	PPT: Prompt if want to buy *n* tracks (screen 90-93) Display popup. See 4/4 for wording When the first track is available, add it into the Current Playlist after the current track Add all remaining tracks after this. Prefetch and start playing this track.	Open
A Track ...in an album ...in a playlist (Not ...in a track list ...in My Tracks ...search results)	Add to Playing Play Next Play ASAP Add to Playlist Rate Delete * Artist Profile Send to Friend Who's Listening? Close	PPT: If don't own, prompt if want to buy (screen 80-83) Display popup. See 4/4 for wording. Wording depends on whether first, middle or last track. Add this track and all subsequent tracks in the playlist/album to end of Current Playlist (whether downloaded yet or not)	PPT: If don't own, prompt if want to buy (screen 80-83) Display popup. See 4/4 for wording Wait until the first track is available. Insert first track into the Current Playlist after the current track Insert all remaining tracks after it, even if not downloaded yet. Keep playing current track while still prefetching this 1st track. Play it once the current track has finished.	PPT: If don't own, prompt if want to buy (screen 80-83) Display popup. See 4/4 for wording When the first track is available, add it to the current playlist after the current track. Prefetch it and then immediately start playing it. Add all subsequent tracks from the album/playlist after it in the current playlist even if not downloaded yet.	(Play icon) Add to Playing – add the track to the end of the current playlist immediately (even if not on phone)

* * Enabled for owned playlists only; + Only displayed for a shared playlist; ++ Only displayed for a private playlist; ^ Enabled only when there are tracks in the Current Playlist; ~ Enabled only when the playlist is not empty. This option is not shown on screens listing this user's own playlists – you cannot rate your own lits. It is shown and enabled for editorial and shared playlists but not for automatically generated per service or per member playlists;
@ if any of the tracks on handset %available for editorial/system playlists

OMNIFONE™

Intro: Context Sensitive Menus 3/4

Like the right hand click on a PC, each type of object in the interface will have common right hand menu options which can be used to apply common actions to the object

Type of Object	Options on the more menu	Add to Playing	Play Next	Play ASAP	Action of the Middle Joystick Button
A Track ...in a track list ...in a artist screen ...in My Tracks ...in search results (Not ...in a playlist ... in an album)	Add to Playing Play Next **[Play NOW**[1]**|**Play ASAP] Add to Playlist Rate Artist Profile Send to Friend Who's Listening? Close	PPT: If don't own, prompt if want to buy (screen 80-83) Display popup. See 4/4 for wording Add track to end of Current Playlist (whether downloaded yet or not)	PPT: If don't own, prompt if want to buy (screen 80-83) Display popup. See 4/4 for wording If there is already a track being downloaded under Play Next then stop downloading that track and place it at the end of Current Playlist (i.e. treat as Add to Playing on it) Even if track not fully downloaded. Wait until the track is available. Insert track into the Current Playlist after the current track (Continue playing the current track while prefetching this new track) When the current track has finished, start playing this track.	PPT: If don't own, prompt if want to buy (screen 80-83) Display popup. See 4/4 for wording When the track is available, add it into the Current Playlist after the current track (Continue playing the current track while prefetching this new track) Start playing the new track immediately	(Play icon) Add to Playing – add the track to the end of the current playlist immediately (even if not on phone)

Implementation Suggestion – we'd like to retain some flexiblity of the configuration of the way that Play is handled in these situations. Suggest that every object in a certain context (e.g. a track in an album vs a track in a playlist) be assigned a couple of properties which control the behaviour of Play on it:

- Does this item replace the current playlist or not
- When played is this item and all following items added to the current playlist, or is just this item added to the current playlist.

If a track(s) is added to an empty Current Playlist then that also implies that it should start being played as well.

- * Enabled for owned playlists only; + Only displayed for a shared playlist; ++ Only displayed for a private playlist; ^ Enabled only when there are tracks in the Current Playlist; ~ Enabled only when the playlist is not empty);
  @ if any of the tracks on handset [1] if track is on the handset

OMNIFONE™

Intro: Context Sensitive Menus 4/4

Like the right hand click on a PC, each type of object in the interface will have common right hand menu options which can be used to apply common actions to the object

Type of Object	Options on the more menu	Add to Playing	Play Next	Play ASAP	Action of the Middle Joystick Button
An album in a list of albums	Open Add to Playing Play Next Play ASAP Rate Send to Friend¬ Who's Listening? Close	PPT: Prompt if want to buy *n* tracks (see screens 90-93) Display 1 sec info popup: "This album [is being downloaded and] will be added to the end of the current playlist." Add all tracks to the end of the current playlist whether on handset yet or not.	Prompt if want to buy *n* tracks (see screens 90-93) Display 1 sec info popup: "This album [is being downloaded and] will be played next." When the first track is available, add it into the current playlist after the current track. Prefetch it. Add all remaining tracks from the album into the current playlist after this track and start downloading.	PPT: Prompt if want to buy *n* tracks (see screens 90-93) Display 1 sec info popup: "The album [is being downloaded and] will be played ASAP." When the first track is available, add it into the Current Playlist after the current track. Prefetch it Start playing the new track immediately Add the remaining tracks into the current playlist after this track and start downloading.	Open
A artist in a list of artists	Open Add to Playing Play Next Play ASAP Rate Send to Friend¬ Who's Listening? Close	PPT: Prompt if want to buy *n* tracks (see screens 90-93) Display info popup: "This artist's tracks [are being downloaded and] will be added to the end of the current playlist." Add all tracks to the end of the current playlist whether on handset yet or not.	PPT: Prompt if want to buy *n* tracks (see screens 90-93) Display info popup: "This artist's tracks [are being downloaded and] will be played next." When the first track is available, add it into the current playlist after the current track. Prefetch it. Add all remaining tracks from the album into the current playlist after this track and start downloading.	PPT: Prompt if want to buy *n* tracks (see screens 90-93) Display info popup: "The artist's tracks [are being downloaded and] will be played ASAP." When the first track is available, add it into the Current Playlist after the current track. Prefetch it Start playing the new track immediately Add the remaining tracks into the current playlist after this track and start downloading.	Open

* * Enabled for owned playlists only; + Only displayed for a shared playlist; ++ Only displayed for a private playlist; ^ Enabled only when there are tracks in the Current Playlist; ~ Enabled only when the playlist is not empty);
@ if any of the tracks on handset
[is being downloaded and] = include in message when one or more tracks not already on handset

OMNIFONE™

Intro: Add to Current Playlist Popup Wording

Type of Object being added to Current Playlist	Add to Playing wording for 1.33 second popup	Play Next wording for 1.33 second popup	Play ASAP wording for 1.33 second popup
A Playlist in a list of playlists	"This playlist [is being downloaded and] will be [played|added to end of the current playlist]."	This playlist [is being downloaded and] will be played next."	"This playlist [is being downloaded and] will be played ASAP."
A Track ...in an album ...in a playlist (Not ...in a track list ...in My Tracks ...search results)	If first track: "This [playlist|album] [is being downloaded and] will be [played|added to the end of the current playlist]." If middle track: "This and the rest of the tracks [are being downloaded and] will be [played|added to the end of the current playlist]. If end track: This track [is being downloaded and] will be [played|added to the end of the current playlist].	If first track in album/playlist: "This [playlist|album] [is being downloaded and] will be played next." If middle track in album/playlist: "This and the rest of the tracks in this [playlist|album] [are being downloaded and] will be played next." If end track: "This track [is being downloaded and] will be played next."	If first track in album/playlist: "This [playlist|album] [is being downloaded and] will be played ASAP." If middle track in album/playlist: "This and the rest of the tracks in this [playlist|album] [are being downloaded and] will be played ASAP." If end track: "This track [is being downloaded and] will be played ASAP."
A Track ...in a track list ...in a artist screen ...in My Tracks ...in search results ...in a chart list (Not ...in a playlist ... in an album)	"This track [is being downloaded and] will be [played|added to the end of the current playlist]."	"This track [is being downloaded and] will be played next."	"This track [is being downloaded and] will be played ASAP."
An album in a list of albums	"This album [is being downloaded and] will be [played|added to the end of the current playlist]."	"This album [is being downloaded and] will be played next."	Display 1 sec info popup: "The album [is being downloaded and] will be played ASAP."
An artist in a list of artists	"This artist's tracks [are being downloaded and] will be [played|added to the end of the current playlist]"	"This artist's tracks [are being downloaded and] will be played next."	Display 1 sec info popup: "The artist's tracks [are being downloaded and] will be played ASAP."

[is being downloaded and] = include in message when one or more tracks not already on handset

[played|added to the end of the current playlist]

= Display "played" when the current playlist is empty
= Display "added to the end of the current playlist" when the current playlist is not empty.

OMNIFONE™

Intro: Fonts

Fonts

Default font	10.5 pt Frutiger 55 (Roman)
Popups: Option text alongside radio buttons	10.5 pt Frutiger 45 Light
Popups: Body text	Main text 10.5pt Frutiger Roman... but the bullet points are 9pt frutiger light
Button Text (OK, Yes, No)	9.5 pt Fruitiger 45 Light
Page Blue header (Artists, Browse Artists etc...)	10.5pt Frutiger 75 Black (you could use Frutiger 65 Bold)
Small text under New Playlist in a grey selection bar	8pt Frutiger 45 Light (if not legible then increase it to no bigger than 9.5 pt Frutiger 45 Light

Note on the PC some of the frutiger fonts are different from on the Mac. For one graphic we got from Mark we had to change the font from Mark's Mac one to ~~Frutiger LT 87~~ Extra Black

Intro: Clients Features √

Discussed here are various important functional & architectural elements of the client application.

Memory
- Hide the physical storage place of details (phone & card) from the user.
- App makes some assumptions about what memory it can use, but it can be tuned – see Options.
- Should we perform clever device local memory management – I.e. get a large chunk and write our own files out across it rather than each file being an individual file?
- If the phone can handle a memory card but the user doesn't have one, at some point when the memory is getting full up, mention the fact to the user that they can use one and would benefit.

Downloading & Playing
- Deal with flow issues.
- Stream or not – can we determine the throughput in the app?
- Should we wait until full track down or 3g stream?
- What do we do if there's more to download but we've run out? Play next available? What then? Go back to start?
- Does 2.5G need a slightly different config?
- What's the average d/l to playtime ratio/advantage?
- Partial downloads
- Progressive downloads
- Make sure not possible for client to hang on a network call
- When choose a new track to download play what happens?
- Pop-up saying "Play Immediately Available/Make Next Track When Available/Add To End Of Current Playlist"
- End of playlist but stuff still downloading – ask user what they want to do?

Caching
- Caching locally the right content at the right time.
- Sensitive to GPRS costs for phone/network/package
- Seamless user experience blurs the online/offline boundary so that the user perceives that the application is permanently on

Tracking/Audit Trail
- Should we connect at the start and send off a stream of what's done on the client
- Or should we just buffer the audit data and send when there is a connection?

Client Updates
- Allow modular updates to clients post-launch.
- Can we update s/w too (not via full app re-download).

Exception Handling
- Incoming call, pause track. Then easy press play.
- Can we pick up on standby screen key presses like joystick?
- Can we determine the difference between incoming call and incoming SMS for example?

Intro: Track Info(#)

The Track Info popup is displayed when the user selects Info against any track.

Display of label etc information for a track is a requirement of labels/aggregators. This popup will also be used for displaying a track rating.

Intro: Application Start-up √

Described here are the items important to the start-up of the MusicStation application.

Language on Start-up

At start-up the application uses the J2ME variables to check the current language setting of the phone's o/s. If it detects that the language of the phone has been changed from what it was to a new language (which we support) then the application asks them if they want to change their language (in both of the launages). E.g.

Changez la language de MusicStation en Allemand?
Change MusicStation to English?
Yes/Oui
Non/No (note switched order)

Music Playing on Start-up

- Current Now Playing starts
- If press Play and no Current Playing then starts from start of library.
- Shuffle
- When the app turns on it's music, music, music – the way a car stereo works. The music play should start from where it left off.

Volume on Start-up

- On app start-up read the phone's current volume setting. Write this "orig phone volume" setting to memory.
- Each time MusicStation volume changed write the current volume setting to memory.
- On Close of the application reset phone volume to orig phone volume.
- On app start-up, after read & store orig phone volume, reset MusicStation volume to last volume stored for MusicStation in memory.
- Detect phone volume change and show volume change bar temporarily.
- Write volume changes when in app (whether by phone or app) to memory.
- On call interruption reset phone to orig phone volume from memory.
- On back from call interruption set back to last MusicStation volume from memory.

License on Start-up

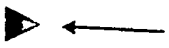
- In some territories would require licence terms agreement on first use.

Visuals on Start-up

- Low key splash screen like Pod's apple symbol on black. Options are:
  - MusicStation branding only with logo and name
  - MusicStation (brought to you by Vodacom) with MusicStation logo and name
  - Vodacom Music Centre with a quick second screen of "Brought to you by MusicStation", and Vodacom defined application name and icon
  - Vodacom Music Centre with Vodacom defined application name and icon
- The first thing to be shown with any real visual impact is the main menu.
- The main menu needs to start ASAP, this is critical. We should not be pre-interpreting lots of other screens or decks before getting the main menu up, we should be focussing on getting the main menu up ASAP.
- If no splash screen now on start-up then perhaps we should have a splash screen?
- Is this pre-loaded content in the Current Playlist at start-up?
- Preloaded content could include welcome track, help tracks, small app tutorial, house tracks?

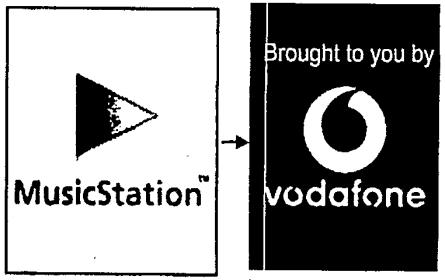
SIM Card

- Check SIM & IMEI?
- Only do phones were we can talk to SIM or get IMEI?
- Remember we are working with operators here so we can solve things in more than one way if we want positive ID.
- How to handle a SIM card change?
- Can we get any other identifiers out of phone other then IMEI?

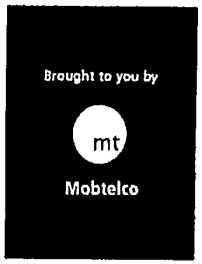


Application Icon ← Something like this on the phone menu.

05. Single Start-up Screen Example



Example of two stage Start-up Screen



06. Second splash screen cou be operator specific. Show for Of 2 seconds.

OMNIFONE™

Intro: Offline demo mode

Details of what happens when the application is switched into offline demonstration mode. This option switches the application so it does not attempt to make a connection. An equivalent of this option might be used when roaming to prevent excessive data charges.

Feature

- **Offline Demo Mode** is switched on and off by selecting the Offline Demo Mode option on the Options screen

Questions

- Is the behaviour for this mode identical to that which the user would experience if they were in a 0G environment (tunnel etc)?

Function	**Behaviour in offline demonstration mode**
Start MusicStation for the first time (causing registration process)	Display popup: MusicStation can not connect in offline mode to register your new account. Please switch to online mode and try again. [OK]
Toggle Unlimited Access in Options screen	Display popup: "Sorry! MusicStation cannot switch to unlimited download mode whilst in offline demonstration mode. Enable the online demonstration mode and try again. [OK]" "Sorry! MusicStation cannot switch to Pay Per Track mode whilst in offline demonstration mode. Enable the online demonstration mode and try again. [OK]"
Press JS or More>OK on Search screen Press JS or More>OK on Advanced Search screen	Display popup: Sorry, search is not available in offline demonstration mode. [OK] (Note we still want to allow the demo-er to go into the search screen, enter their search text and options etc.)
Trying to go to a screen for which we have yet to download the data	Display popup: Sorry! *optionname* is not available in offline demonstration mode. Enable the online demonstration mode and try again. [OK] Where *optionname* is the name of the menu option or screen. For example, "Top Artists"
Pay Per Track: Playing an owned track where track not currently on handset	Use a default track which says: "Sorry! This track is not available because MusicStation is in offline demonstration mode. To download new music please switch to online demonstration mode."
Unlimited: Playing a track not currently on handset	Use a default track which says: "Sorry! This track is not available becaue MusicStation is in offline demonstration mode. To download new music please switch to online demonstration mode."

Intro: Use of capitalisation in labels and menu items

When displaying text in the MusicStation application we will use the following rules to give a consistent look and feel to the text:

The convention is that:

1) Titles in **blue** use capitalisation on every word (as we already do).

2) The following only use capitalisation on the FIRST word:
- Menu items (e.g. "Featured tracks)"
- Items on the More menu (e.g. "Now playing")
-Main screen pop-up radio option list items (e.g. "Unlimited access" and "Hear free preview")

3) Genre names are not treated as menu items for the rules above. Instead genre names will continue to use capitalisation on all words. i.e. **Top Rock** as opposed to **Top rock**, and **Top Rap/R&B**.as opposed to **Top rap r&b**

4) The only other time when capitalisation should be used on all words in a menu item is when what is being referred to is a branded term or similar. E.g. We currently have the menu item **Add Top Track**. If **Top Track** is the quirky name for a feature of the app then that is fine (e.g. as in Add Top Track™). Otherwise it should be **Add top track**.

This gives us a screen like this:

Get New Tracks
Featured tracks
Just released
Recently added
Top tracks

You Might Like
Ashes To Ashes -David Bowie
Callbacks - We Are Scientists

Use of capitalisation on the first word only is a more modern approach and would also allow us to distinguish menu items from Artist / Album and Track names which currently also use capitalisation on every word.

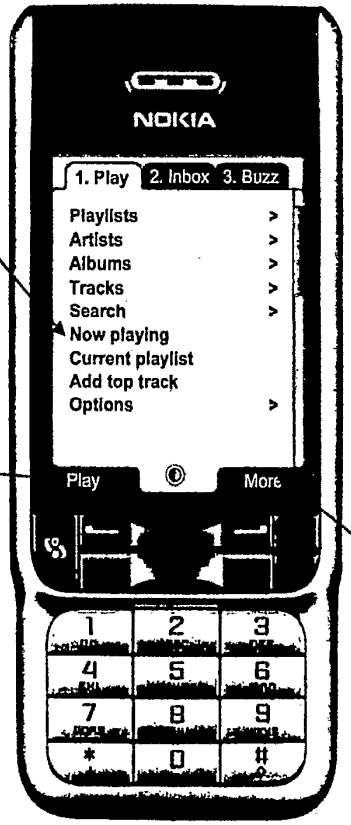
01: Main Menu √ (#)

The **Main Menu** is the first screen that is shown when the system starts up. Inside this menu the user can explore unlimited music and their own musical favourites. This menu starts instantly.

Help Popup

Note changed from Now Playing. Goes to 20. Current Playlist. Greyed out if no tracks in current playlist yet.

Start playing the **Current Playlist** or if none then play first track in My Music (e.g. My Artists, My Tracks, My Albums).
Or use as key as Open? Or have no function on it?



01. Main Menu

Features

- Minimise puts the app into the background whilst continuing to play if currently playing.

Snapback To Play Track

If a track is playing and the user is on this screen (and most other non-modal screens or search results, but not on wordier screens like stories in Inbox), if the user does no key presses for X seconds (configurable, say 5) then the screen changes to the **Play Track** screen with the current track showing.

Add Top Track

Randomly selects a track which is in the user's Top Tracks (i.e. those on the handset) but not in the Current Playlist and adds it to the end of the current playlist. Displays popup whose wording depends on whether current playlist is empty or not:

DARE by Gorillaz will be added to the end of the current playlist.

DARE by Gorillaz will be

Softkey

- Open or Open Folder depending on context
- Now Playing – go to Play Track (if playing)
- Play/Pause (Current Playlist)
- Close (see over)

Navigation	
◀◀	No function
◀	No function
▲▲	Up item
▲	Up item
●	Open folder/Start function
●●	Open folder/Start function
▼	Down item
▼▼	Down item
▶	Open folder (if selected)
▶▶	Open folder (if selected)

Other Ideas

- Should Shuffle & Play be Random Play
- Play My Top 10 Tracks – plays your fave tracks based on what you've listened to

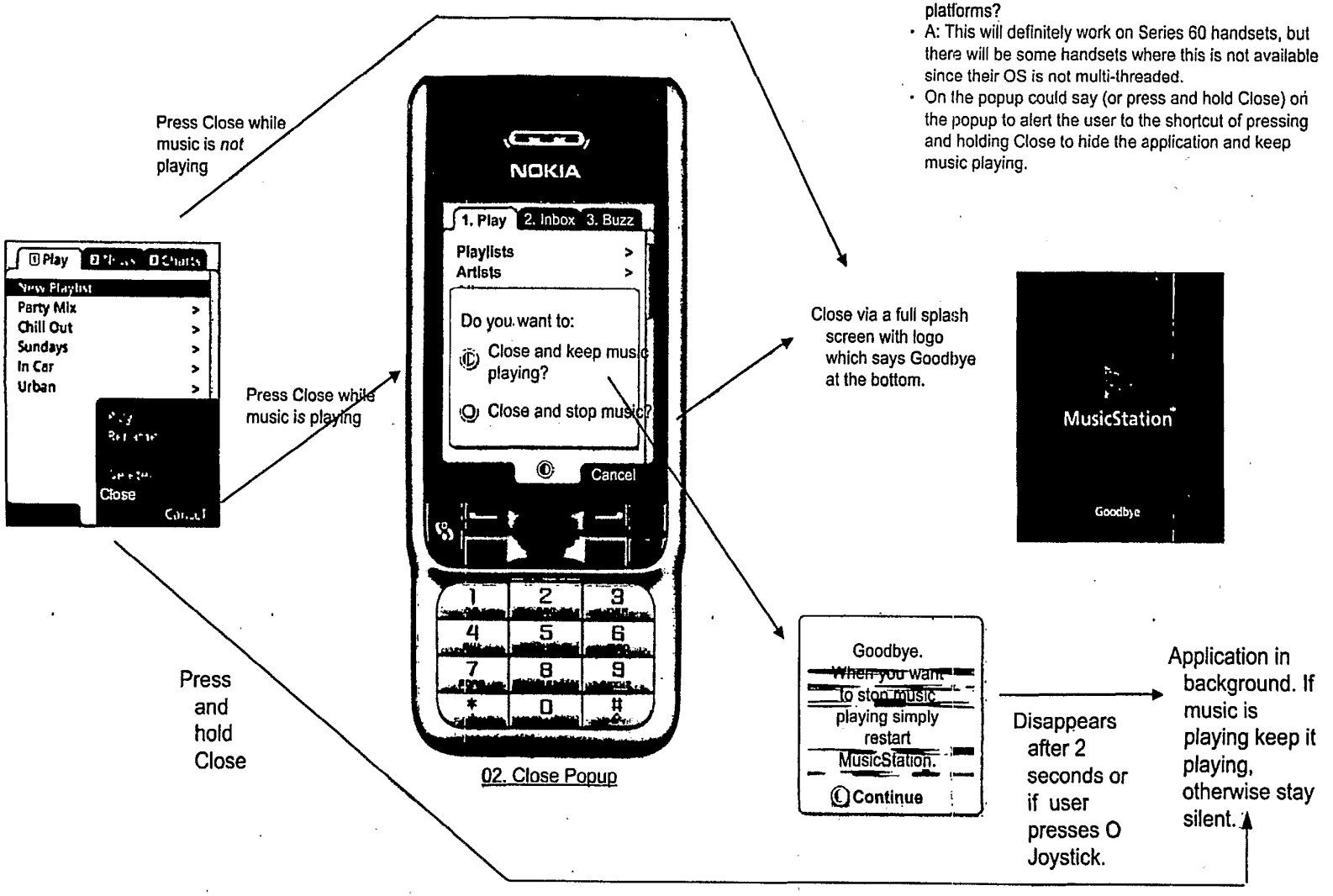
OMNIFONE

02: Close Popup√

Whatever screen the user is in they can close the application from an **Close** option on the **More** softkey.
This popup will only be displayed for handsets which would support the application being minimised and continuing in the background.

Queries

- Can playing on minimise be achieved OK on all platforms?
- A: This will definitely work on Series 60 handsets, but there will be some handsets where this is not available since their OS is not multi-threaded.
- On the popup could say (or press and hold Close) on the popup to alert the user to the shortcut of pressing and holding Close to hide the application and keep music playing.



OMNIFONE

10. Playlists Main √ (#)

Playlists is accessed from the **Main Menu** and allows the user to scan through their own playlists, those on the network and those shared by others.

This tab can be accessed by pressing the '1' key. Press and hold '1' to get to this screen, the Play menu, where you can find & play your favourite music and configure your options.

Continue

Help Popup

Current Playlist is greyed out if nothing in there.

A star rating can only be displayed for public or editorial playlists. Private playlists will not display stars here since they cannot have been rated by other users.

Back up hierarchy or back from where we came?

10. Playlists Main

If the user has marked any playlist as Public then the list of owned playlists will be broken into two sections with headings as shown.

Softkey

- Open
- Play (grey out if empty)
- Add to Playing (grey out if empty)
- Copy – copy a playlist, entering name for the new one
- Delete
- Share (only displayed if Private, grey out if empty)
- Make Private(Only displayed if Public, grey out if empty)
- Rename (Only for owned playlists)
- Now Playing – go to Play Track (grey out if nothing playing)
- Main menu
- Close

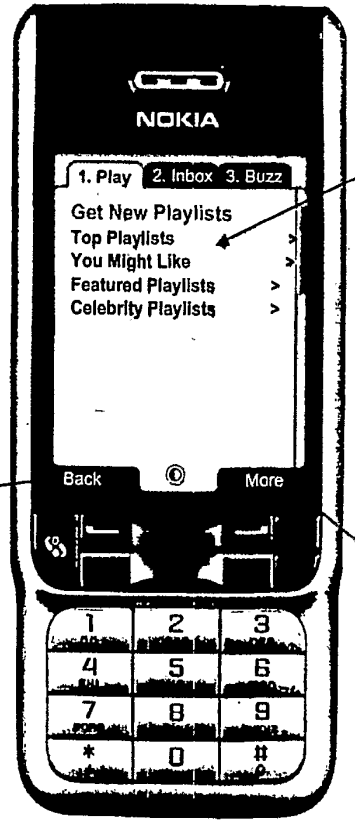
Navigation©	
◀◀	Back/Top Menu?
◀	Back
▲	Up item
▲	Up item
●	Open folder
●●	Open folder
▼	Down item
▼	Down item
▶	Open folder
▶▶	Open folder

OMNIFONE

11/13: Get New Playlists / Playlist Listing √

Get Playlists is the folder where the user can access playlists recommended for them out on the network.

Help Popup

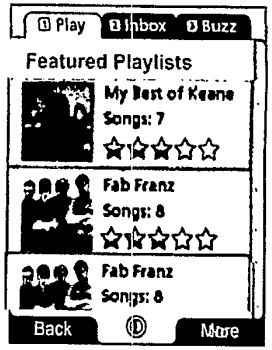


11. Get New Playlists

See over for details of these playlist groups.

Back up hierarchy or back from where we came?

Softkey
- Open
- Now Playing – go to Play Track (if playing)
- Main menu
- Close



25. Get New Playlist – Playlist Group (titles and contents will change according to selected group)

Softkey
- Open
- Play – Will prompt to purchase N tracks if the user on pay-per-track service and does not have all the tracks. See.Buy Playlist Process slide.
- Now Playing – go to Play Track (if playing)
- Main menu
- Close

Navigation	
⏮	Back/Top Menu?
◀	Back
⏫	Up item
▲	Up item
●	Open folder
⏺	Open folder/Play??
▼	Down item
⏬	Down item
▶	Open playlist
⏭	Open playlist/Play??

"Recommended" Discussion

"Recommended" doesn't sound personalised to the user.
"Recommendations4U" is a bit long and naff. "For You", "Picks for You"
Should we have a note on the screen saying how's it's done?
Perhaps "Suggestions" is better?
Do we need a "Recommended" section anyway or is personalisation inherent via the recommendations made in the other playlists in this menu?
How about "You will like" or "You might like"

OMNIFONE™

Playlist Groups shown on Get New Playlists(#)

This is the set of playlist groups that are shown on the 13. Get New Playlists screen. The exact details of the algorithms used for generating these lists are yet to be defined, but the general intention of each group is described.

Group Name	Description
Top Playlists	The top shared playlists across the whole system instance (I.e. an installation of MusicStation for a particular operator). It should be based on all playlists listened to and explicitly rated by all users. It is not personalised to the current user.
You Might Like	Playlists recommended for this user selected from the shared (and editorial?) playlists in the system. (Shared and Editorial playlists can be rated by members).
Featured Playlists	Playlists which have been editorially created and pushed for promotion.
Celebrity Playlists	Editorially created playlists, created or influenced in some way by celebrities.

We need to consider what is shown to a user when they first start the app and have no history to generate recommendations from.

Considerations

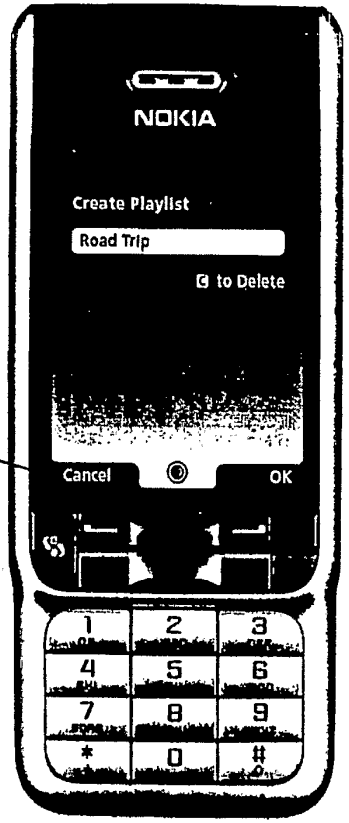
- Include a playlist chart for the week and an all-time playlist chart.
- Consider detailed rules for each of these lists.

11: Playlists (New Playlist Popup) √

New Playlist is a popup which allows the user to specify the name of a new playlist..

This playlist is ready to have tracks added to it. Select Add to Playlist on any track.
Continue

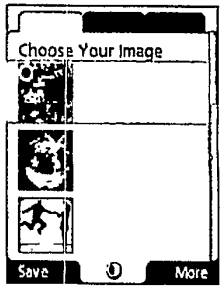
Display this popup after they add the first playlist via this screen.



11. Playlists Main (New Playlist Popup)

Features

- T9 (predictive) text if possible.
- No limit to number of playlists user can create
- Allow playlists of the same name? If not then need exception dialogue.
- Accept any text? Punctuation etc?
- Understand the phones text & keyboard layout and offer exactly the same.



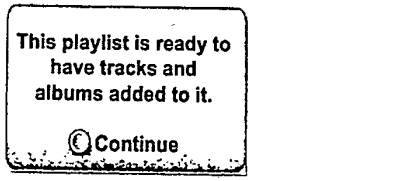
User selects an image for the playlist from the predefined set after they have entered the name. This image selector needs improvement.

Navigation	
◀◀	Start of SLE
◀	Back char
▲	No function
▲	No function
●	Save (if different)
●	Save (if different)
▼	No function
▼	No function
▶	Forward char
▶▶	End of SLE

OMNIFONE™

11: Playlists (Copy Popup) √

Copy Playlist is a popup which allows the user to copy an existing playlist..



Display this popup after they add the first playlist via this screen.

11. Playlists Main (Copy Playlist Popup)

Navigation	
◀◀	Start of SLE
◀	Back char
▲▲	No function
▲	No function
●	Save (if different)
●●	Save (if different)
▼	No function
▼▼	No function
▶	Forward char
▶▶	End of SLE

Features

- T9 (predictive) text if possible.
- No limit to number of playlists user can create
- Allow playlists of the same name? If not then need exception dialogue.
- Accept any text? Punctuation etc?
- Understand the phones text & keyboard layout and offer exactly the same.

OMNIFONE™

12: Playlist Items (No description) √

Playlist Items is the screen which shows all the tracks and albums in a playlist.

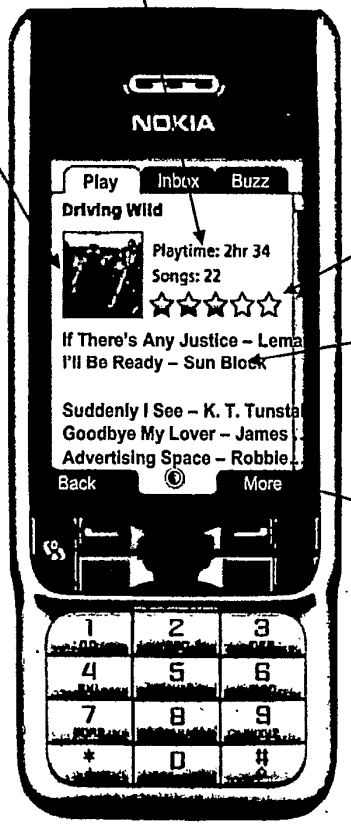
Help Popup

Empty playlist

Navigation	
◀◀	Back/Top Menu
◀	Back
▲	Up item
▲	Up item
●	Open (if album)
●●	Open (if album)
▼	Down item
▼	Down item
▶	Open (if album selected)
▶▶	Open (if album selected)

Play time, tracks and rating shown next to photo. This is total for all tracks in the playlist, not just those owned/listened to.

Playlist photo



12. Playlist Items

A star rating can only be displayed for public or editorial playlists. Private playlists will not display stars here since they cannot have been rated by other users.

Show "Title – Artist" and horizontally scroll when highlighted. Wait for XXXms (e.g. 1000 or 1 sec) before scroll begins, not immediate.

Features

- Include tracks or albums.
- Albums with tracks in order of album
- Clicking on an album goes into tracks in album, stop particular tracks from albums (see **Album Page**)
- Generic playlist image if none chose
- Changes the order on sort

Softkey

- Open (if album)
- Play
- Add to Playing – Add to end of Current Playlist
- Add to Playlist – Add to selected playlist
- Move* – move the selected item in list. See Edit Playlist page for desig
- Description* – enter/edit description. See Edit Playlist page for design
- Photo* – Modify the image associated with the playlist
- Sort* - see Sort Playlist popup
- Shuffle
- Rate - see Rate Track
- Now Playing – go to Play Track (if playing)
- Main menu
- Close

* Only available on playlists which are owned/editable

This is a very long list – we might want to leave playlist level options (like Description and Photo to the Playlist Menu)

12: Playlist Items (With description) √

This is the version of **Playlist Items** when there is a description.

Playlist description

12. Playlist Items

Editable playlists

Only playlists which the user created in the first place can be edited using the
Remove
Move
Description
Photo
Sort options

Softkey

- Open (if album)
- Play
- Add to Playlist – Add to selected playlist
- Add to Current – Add to Current Playlist
- Remove – remov ethe selected item
- Move – move the selected item in list. See Edit Playlist page for desig
- Description – enter/edit description. See Edit Playlist page for design
- Photo – Modify the image associated with the playlist
- Sort - see Sort Playlist popup
- Shuffle
- Rate - see Rate Track
- Now Playing – go to Play Track (if playing)
- Main menu
- Close

Navigation©	
◀◀	Back/Top Menu
◀	Back
▲	Up item
▲	Up item
●	Open (if album)
●●	Open (if album)
▼	Down item
▼	Down item
▶	Open (if album selected)
▶▶	Open (if album selected)

OMNIFONE

12: Playlist Items (moving items within)

The **Playlist Edit** screen is used to edit playlists. Items can be discarded (removed) or items can be selected and then moved up or down using the joystick.

Playlist Description Popup

The popup is shown when Description is selected from the soft key menu

"Save" takes same action as O Joystick and saves edits.

12. Playlist Items (in edit order mode)

Features

- Select an item then move it up or down then drop.

When the user has selected the Move menu option then the highlighted item can be moved up and down in the list. This is achieved by using the joystick up and down. The highlighted item moves up and down in the list accordingly.

To Close this mode the user presses O Joystick

This mechanism is based on the Shortcut editor within Sony Ericsson phones which work in the same way to move items up and down in the short cut menu (typically acessed by pushing the joystick up on the home screen of the

Sort Playlist Popup

Popup shown twhen the user selects Sort soft key option

Navigation	
◀◀	Back/Top Menu
◀	Back
▲▲	Up item/move item up (if selected)
▲	Up item/move item up (if selected)
●	Drop item
●●	Drop item
▼	Down item/move item down (if selected)
▼▼	Down item/move item down (if selected)
▶	Open (if album selected)
▶▶	Open (if album selected)

OMNIFONE™

14: Add Photo

Add Photo allows the user to choose a photo off his phone memory and load it as the image to represent his playlist.

Need to decide what is easier: controlling camera or looking through file system. Accessing file system might give more options to the types of images the user could use.

14. Add Playlist Photo

Attach Image To Playlist

- Browse photo images on their phone.
- Filter out files we can't handle
- Allow select, auto-size/crop
- Post (orig) image to the server with the playlist
- Some people prefer the two line artist and track layout – shall we look at see on device?

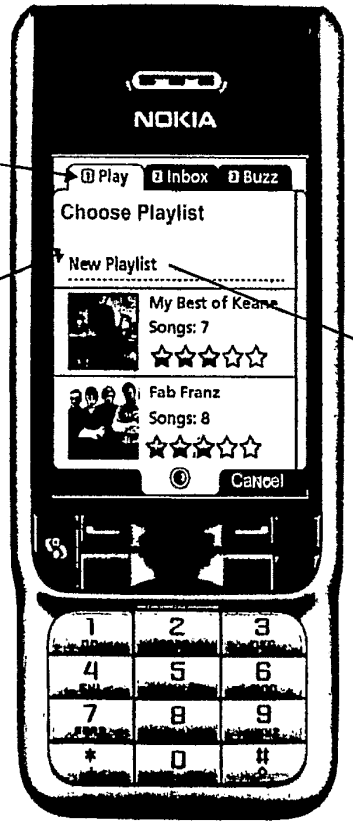
15: Add to Playlist √

Add To Playlist is the screen shown if the user hits the "Add To Playlist" softkey whilst browsing their music or the available music on the network. The user is asked to select an existing playlist or to add a new one.

Help Popup

Maybe we should grey these out

See New Playlist Popup



15. Add to Playlist

Features

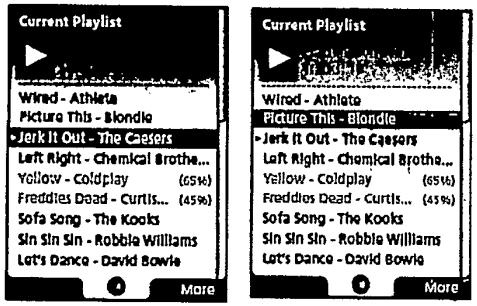
- If this is the first track to be added to a playlist then can we give them the option to start playing it when added?
- If they create a new playlist then have that as selected after creation.
- IF the track is not owned then we will need a popup along those at 80-83 which ask whether the user wants to buy but then go on to allow adding to playlist rather than asking about the play process.

Navigation	
◀◀	No function
◀	No function
▲▲	Up item
▲	Up item
●	Select
●●	Select
▼	Down item
▼▼	Down item
▶	No function
▶▶	No function

OMNIFONE

20: Current Playlist (#)

The **Current Playlist** is accessed from the **Playlists Menu** or **Current Playlist** on most **More** menus. It is the list of tracks which are currently being played and lined up to be played.



The user can select a current track in order to change what is Now Playing. This selection is highlighted with a blue selection bar. By default it will be the Now Playing track but the user can select another track.

Static icon

Triangular indication to show Now Playing track.

Clicking on a greyed out track displays ful lscreen popup

That track is not yet downloaded and is not available to

Should be the triangular play icon

20. Current Playlist

We grey out tracks which have been theoretically added into the playlist but which are not yet available on the phone to play. If the Now Playing pointer gets to them they would be skipped.

Download progress for tracks which are not on the handset are shown in steps of 5%.

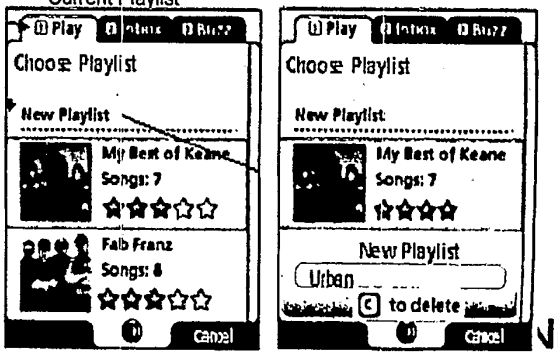
Navigation©	
◀◀	Tbd
◀	Tbd
▲▲	(fast) Up item
▲	Up item
●	Play (tracks on phone only)
●●	Play (tracks on phone only)
▼	Down item
▼▼	(fast) Down item
▶	Tbd
▶▶	Tbd

Features

- The only way to empty/restart the Current Playlist is to select the Clear option on the More menu. Otherwise tracks are just being inserted into this playlist
- We considered that the Current Playlist should be saved on Close and restored on Start, but because most operations now add to the current playlist, this would allow the current playlist to keep growing.
- Screen has a snapback to Now Playing of 20 seconds
- We should consider a Go To Artist option to let user jump straight to artist of a track. Napster provide this option on their current playlist. Similarly Go To Album.

Softkey

- Play – Make the currently highlighted track the Now Playing track. *Only enabled for tracks on the phone.*
- Rate- Rate track
- Clear Playlist – stops playback and clears the current playlist
- Save as Playlist – *goes to screens 15: Add to Playlist and on completion of this process restums to Current Playlist screen*
- Remove – Remove current highlighted track from Current Playlist



22: Choose Genre

The **Choose Genre** screen is shown to new users before entry to a few different recommended tracks, artists, albums or playlists screens. As they have consumed no music so far so we have no genre signature for them. We therefore need to know something before choosing a recommendation set for this guy. This is done by inserting a **Choose Genre** screen so as to get a start point on this user's genre signature. The screen appears to be part of the normal app flow but after they have started to consume music this step won't be here because we have a more personalised genre signature for them.

Discussion
Do we tell them at this point what we are up to with asking them this? Does this ask more questions than it answers?

MusicStation is a personalised service which moulds to your tastes and provides recommendations. Choose a genre of interest now to start the process of learning your interests.
Continue

22. Choose Genre

Features
- Quick up and down then joy in opens the **Recommended Playlists** screen.
- Should we allow multiple genres to be selected here?

Navigation	
◀◀	Back/Top menu
◀	Back
▲▲	Up item
▲	Up item
●	Choose genre
●●	Choose genre
▼	Down item
▼▼	Down item
▶	No action
▶▶	No action

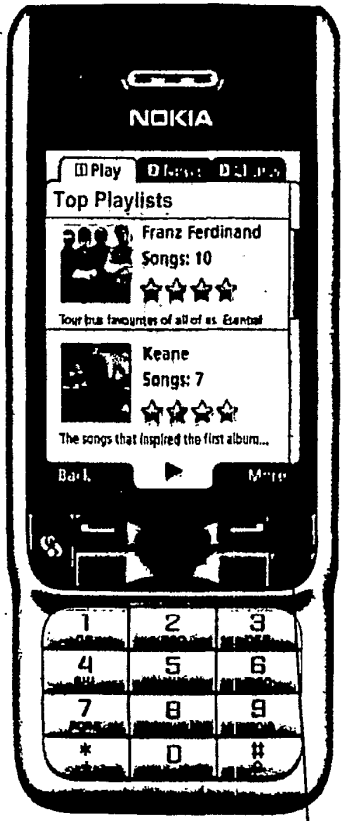
OMNIFONE™

25: Get New Playlist – Playlist Group (Top Playlists etc)

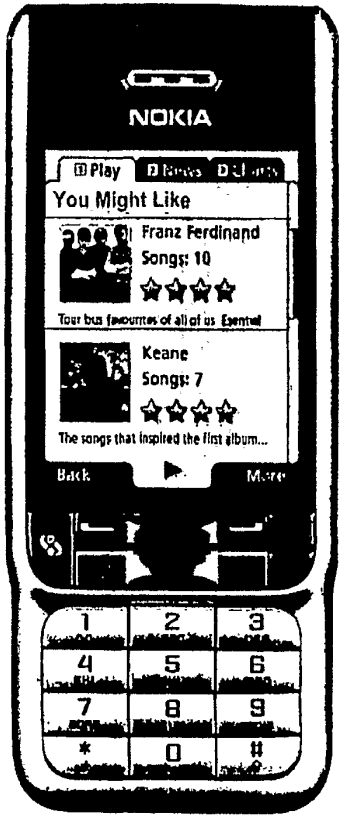
See previous slide for the definition of the contents of this screen.

Questions

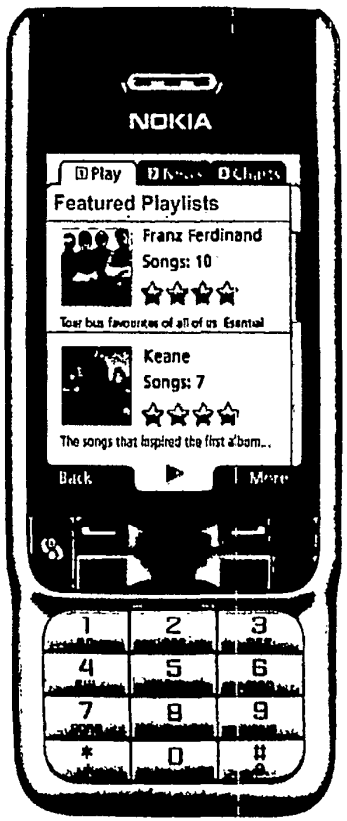
- Top Playlists is probably the same list as the Playlists chart on the Buzz tab



25. Get New Playlist – Playlist Group



25. Get New Playlist – Playlist Group



25. Get New Playlist – Playlist Group

Navigation©	
◀◀	Back/Top menu
◀	Back
▲▲	Up item
▲	Up item
●	Open playlist
●●	Open playlist
▼	Down item
▼▼	Down item
▶	Open playlist
▶▶	Open playlist

Softkey

- Play
- Play next
- Open – see Playlist Items – perhaps no edit though?
- Rate – see Rate Track
- Now Playing – go to Play Track (if playing)
- Main menu
- Close

OMNIFONE™

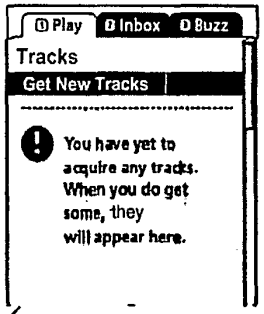
30/31/32: Tracks Main and Get New Tracks (#)

This is the main menu of the tracks

Suggestion from MK to use "Discover Tracks"

30. Tracks Main

When pressed performs Add to Playing



No tracks yet
When I do not yet own any tracks (pay per track)
or listened to any tracks (sub) then displays:

You have yet to acquire any tracks.
When you do get some, they will appear here.

You have yet to listen to any tracks.
When you do, they will appear here.

Discussion Point

- Before Beta need a discussion about how/if we handle the differentiation between owned and downloaded tracks and how they should be displayed here. See email exchanges 1st week of April 06.
- What happens to the list of items I own once I've upgraded to a subscription. Do we need to differentiate from items I've listened to under subscription?

See over for details of the groups of tracks shown on the Get New Tracks screen

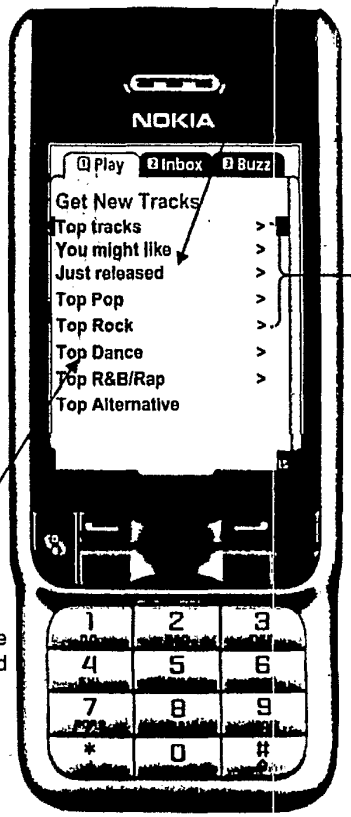
Get New Tracks

All my tracks are listed here for quick and easy access. Meaning of what my tracks is depends on whether pay-per-track or subscription account. For subscription it is tracks cached on the phone. (Those tracks which are 100% downloaded and available to play)

Softkey

- Play Now
- Add to Playing
- Play ASAP
- Add to Playlist
- Rate
- Now Playing – go to Play Track (if playing)
- Play/Pause (Play Queue)
- Main Menu
- Close

All Tracks goes to 60 Simple Search with the Tracks radio button pre-selected



31. Get New Tracks

32. Get New Tracks – Track Grou
(contents depend on group selected, definitions of which are on next slide)

Track Groups shown on Get New Tracks (#)

This is the set of track groups that are shown on the 30. Tracks Main screen. The exact details of the algorithms used for generating these lists are yet to be defined, but the general intention of each group is described.

Group Name	Description	Filtered to exclude tracks the user has listened to (subscription) / own (pay-per-track)
Top tracks	The top tracks across the whole system instance (I.e. an installation of MusicStation for a particular service). It should be based on all tracks listened to and explicitly rated by all users. It is not personalised to the current user.	No
You might like	Tracks that we recommend to the user based on their (recent) listening habits, and taking into account any explicit ratings that they have given. Filtered to remove tracks that the user owns or has listened to (subscription).	Yes
Just released	The same as Top Tracks but only those with a release date within the last two weeks.	No
Top pop Top rock Top dance Top R&B/rap Top alternative	As for **Top tracks** but filtered by genre. We display the top 20 track from each genre. We assign genres to tracks by taking the genre of the release they belong to.	No
Recently Added	A list of (back catalogue) tracks which have been recently added to the system. Even those are new to the system they could potentially be old back catalogue releases.	No
Featured Tracks Some duplication of tracks across lists is allowed	A list of tracks which have been editorially pushed for promotion. Personalised for the artists and genres the user listens to.	Yes
All Tracks	*This is not actually a group, but instead displays the 60. Simple Search screen with the Tracks radio button pre-selected*	N/A

Not v1

Considerations
- Include a track chart for the week and an all-time track chart.
- Recently Added could be a hierarchy of artist/track etc
- Consider detailed rules for each of these lists.
- We don't have track level genre data so will not display per genre track lists here.

We need to consider what is shown to a user when they first start the app and have no history to generate recommendations from.

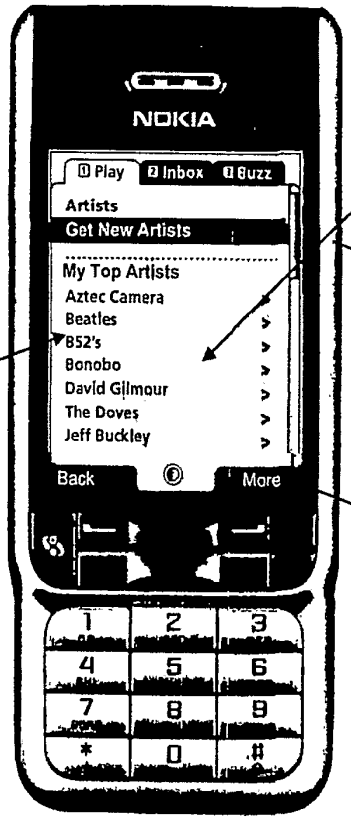
40/41: Artists Main & Get new Artists√ (#)

Artists Main is the main menu for looking for music by artist.

Browse through all the artists in the world, find music from the top artists, play music from your favourite artists or browse some artist recommendations.

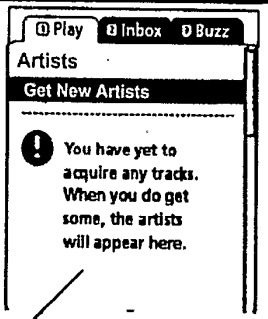
Continue

Help Popup



40. Artists Main

All my artists are listed here for quick and easy access. Meaning of what my artists is depends on whether pay-per-track or subscription account. For subscription: **This is the list of artists for all the tracks which are currently downloaded onto the phone. Only tracks which are 100% downloaded are counted. So this is the list of artists for all tracks in the My Top Tracks list.**



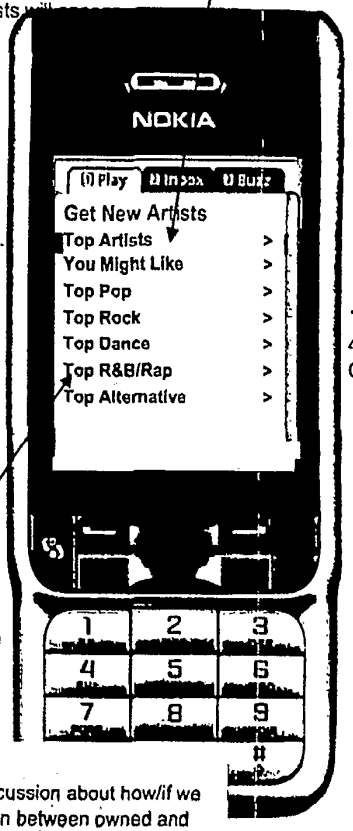
No tracks yet
When I do not yet own any tracks (pay per track)
or listened to any tracks (sub) then displays:

You have yet to acquire any tracks.
When you do get some, the artists will appear here.

You have yet to listen to any tracks.
When you do, the artists will appear here.

See over for details of each of these gr

Get New Artists



42. Get New Artists – Artists Grou

Softkey
- Open
- Now Playing – go to Play Track (if playing)
- Play/Pause (Current Playlist)
- Main menu
- Close

All Artists goes to 60 Simple Search with the Artists radio button pre-selected

Navigation	
⏮	Back/Top menu
◀	Back
⏫	Up item
▲	Up item
●	Open folder
●●	Open folder
▼	Down item
⏬	Down item
▶	Open folder
⏭	Open folder

Discussion Point
- Before Beta need a discussion about how/if we handle the differentiation between owned and downloaded artists' tracks and how they should be displayed here. See email exchanges 1st week of April 06.
- What happens to the list of items I own once I've upgraded to a subscription. Do we need to differentiate from items I've listened to under subscription?

Artists

OMNIFONE

Artist Groups shown on Get New Artists(#)

This is the set of artist groups that are shown on the 41. Get New Artists screen. The exact details of the algorithms used for generating these lists are yet to be defined, but the general intention of each group is described.

Group Name	Description	Filtered to exclude artists the user has listened to (subscription) / own (pay-per-track)
Top artists	The top artists across the whole system instance (I.e. an installation of MusicStation for a particular operator). It should be based on all tracks listened to and explicitly rated by all users. It is not personalised to the current user.	No
You might like	Artists that we recommend to the user based on their (recent) listening habits, and taking into account any explicit ratings that they have given. Filtered to exclude tracks listened to (subscription) or owned (pay per track)	Yes
Top pop Top rock Top dance Top R&B/rap Top alternative	As for Top Artists but filtered by genre. We display the top 20 artists from each genre. Top artists are worked out in the same way as for the Top Artists list above.	No
Just Released	The same as Top Artists but only those with a track or album release date within the last two weeks. It's the artists of the items in the Just Released Tracks and Just Released Albums groups (see Get New Tracks and Get New Albums screens)	No
Recently Added	A list of artists which have been recently added to the system for the first time. This is artists for whom we have added content, where we had *no* content from that artist before.	No
Featured Artists	A list of artists which have been editorially pushed for promotion.	Yes
All Artists	*This is not actually a group, but instead displays the 60. Simple Search screen with the Artists radio button pre-selected*	N/A

Not v1

Considerations
- Include a artist chart for the week and an all-time artist chart.
- Recently Added could be a hierarchy of artist/album etc
- Consider detailed rules for each of these lists.
- Need to define logic we use in classifying an artist as listened to/owned for purposes of filtering out from these lists.

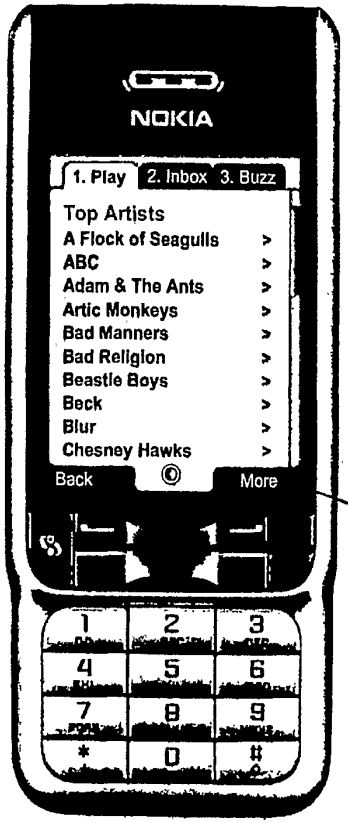
We need to consider what is shown to a user when they first start the app and have no history to generate recommendations from.

OMNIFONE

42: Get New Artists – Artists Group (e.g. Top Artists √)

This screen displays the contents of an Artists Group as selected from 41. Get New Artists. The contents of this screen depends on the group selected and is defined by the rules on the previous slide.

On the full version of MusicStation I we'll need a way to stop featuring an artist in any Get New groups when the user has already selected all available tracks by that artist (SteveG)



42. Get New Artists – Artists Group

Softkey
- Open
- Now Playing – go to Play Track (if playing)
- Play/Pause (Current Playlist)
- Main menu
- Close

Navigation©	
◀◀	Back/Top menu
◀	Back
▲▲	Up item
▲	Up item
●	Open artist
●●	Open artist
▼	Down item
▼▼	Down item
▶	Open artist
▶▶	Open artist

OMNIFONE

45: Artist Homepage √

The **Artists Homepage** is the screen which collates all albums and tracks of this artist.

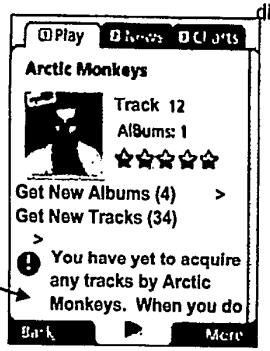
Help Popup

You have yet to acquire any tracks by Arctic Monkeys. When you do get some, they will appear here.

Or

You have yet to download any tracks by Arctic Monkeys. When you do, they will appear here.

No tracks owned



Either artists level image or cover from top selling album (or any album if that not available). If no artist or album image then we show the default image. We do search for track images – that slows screen display a lot.

First track is highlighted when screen is first opened since that is most likely action user wants to take.

45. Artists Homepage

Playtime and count of the tracks which the user owns (ppt) or has listened too (sub)

Browse further tracks and albums by this artist. **Counts** displayed to right in parentheses are total number of albums/tracks available within the option.
Get New Albums goes to **46. Artist Albums**
Get New Tracks goes to **47 Artist Tracks**

All tracks by this artist which I own/downloaded (pay-per-track) or are currently downloaded to the phone and on the phone (subscription)

Features

- Maybe put the record label in there. But perhaps this could vary by track.
- Shows total album and tracks numbers
- Shows a rating which would be nice to be able to highlight and change??
- Artist biog – same as artist profile?
- What happens to the list of items I own once I've upgraded to a subscription. Do we need to differentiate from items I've listened to under subscription?

Softkey

- Open (if Get... option is highlighted)
- Play (if Track is highlighted)
- Add to Playlist (if album or track selected)
- Rate Artist
- Similar Artists (or should it be artists you might like?)
- Artist News
- Artist Profile
- Now Playing – go to Play Track (if playing)
- Main Menu
- Close

Navigation	
◀◀	Back/Top menu
◀	Back
▲▲	(Fast) Up item
▲	Up item
●	Open album (if selected)/Play?
●●	Open album (if selected)/Play?
▼	Down item
▼▼	(Fast) Down item
▶	Open album (if selected)
▶▶	Open album (if selected)/Play?

Discussion Point

- Before Beta need a discussion about how/if we handle the differentiation between owned and downloaded artists' tracks and how they should be displayed here. See email exchanges 1st week of April 06.

Discussion Point

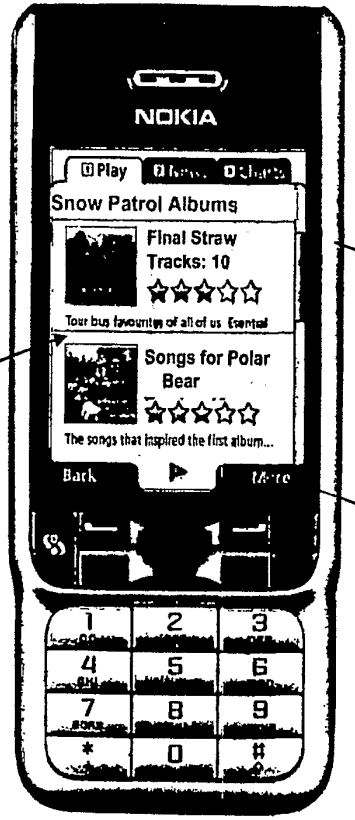
- Consider including "similar aritsts" links in here – I.e. recommendations seeded from what you are currently looking at.

OMNIFONE

46: Artist Albums√

Artist Albums lists all the albums for a particular artist.

Help Popup



46. Artist Albums

All of the artists albums are listed here in alphabetical order initially

Open an album goes to 52 Album Page

Softkey
- Open
- Now Playing – go to Play Track (if playing)
- Play/Pause (Current Playlist)
- Main menu
- Close

Navigation	
◀◀	Back/Top menu
◀	Back
▲▲	Up item
▲	Up item
●	Open folder
●●	Open folder
▼	Down item
▼▼	Down item
▶	Open folder
▶▶	Open folder

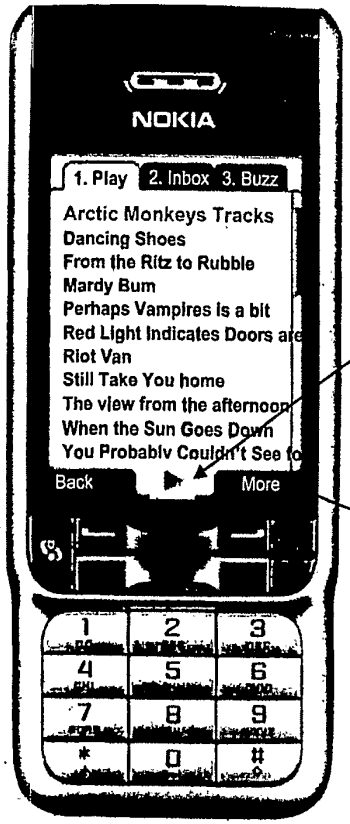
OMNIFONE™

47: Artist Tracks√

Artist Tracks lists all the tracks belonging to a particular artist.

Help Popup

All of the artists tracks are listed here in alpha order originally



47. Artists Tracks

Performs Add to Playing

Softkey
- Play Now
- Play ASAP
- Add to Playing
- Now Playing – go to Play Track (if playing)
- Play/Pause (Current Playlist)
- Main menu
- Close

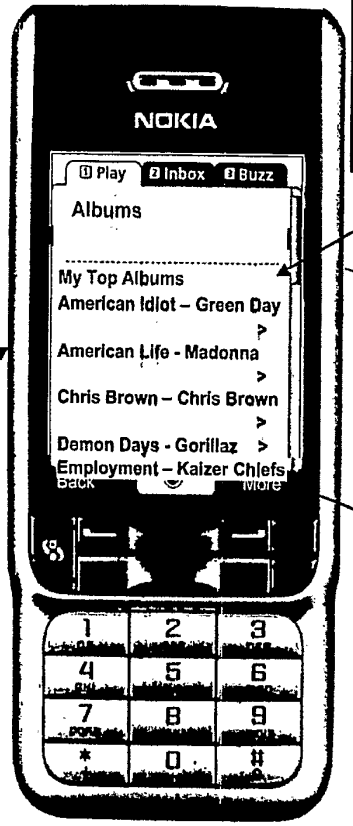
Navigation©	
◀◀	Back/Top menu
◀	Back
▲▲	Up item
▲	Up item
●	Open folder
●●	Open folder
▼	Down item
▼▼	Down item
▶	Open folder
▶▶	Open folder

OMNIFONE™

50/51: Album Main & Get New Albums√

Album Main is the main menu for looking for music by album.

Help Popup



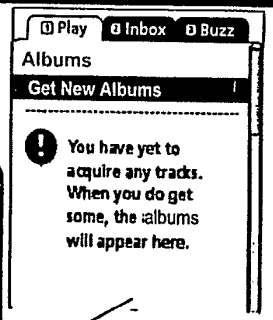
50. Album Main

All my albums are listed here for quick and easy access. Meaning of what my artists is depends on whether pay-per-track or subscription account.

For subscription it is all the artists for the tracks which are 100% downloaded onto the phone.

Note that we display the artist name for clarity.

Navigation©	
◀◀	Back/Top menu
◀	Back
▲▲	Up item
▲	Up item
●	Open folder
●●	Open folder
▼	Down item
▼▼	Down item
▶	Open folder
▶▶	Open folder



No tracks yet
When I do not yet own any tracks (pay per track)
or listened to any tracks (sub) then displays:

You have yet to acquire any tracks.
When you do get some, the albums will appear here.

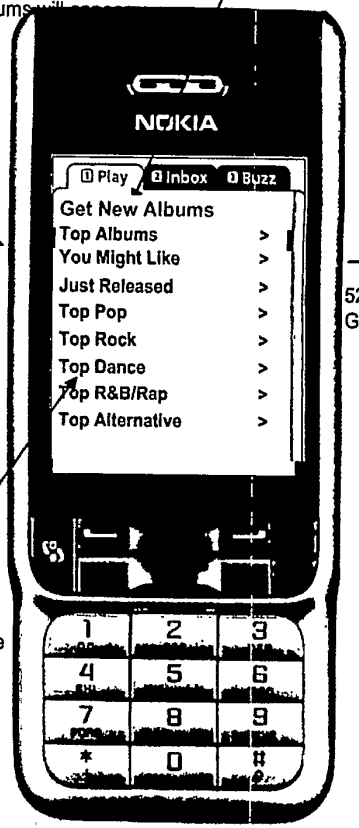
You have yet to listen to any tracks.
When you do, the albums will appear here.

Get New Albums

Softkey
- Open
- Play Now
- Play ASAP
- Add to Playing
- Now Playing – go to Play Track (if playing)
- Play/Pause (Current Playlist)
- Main menu
- Close

All Albums goes to 60 Simple Search with the Albums radio button pre-selected

See over for details of these g



51. Get New Albums

52. Get New Albums – Albums Gro

Discussion Point
- Before Beta need a discussion about how/if we handle the differentiation between owned and downloaded albums and how they should be displayed here. See email exchanges 1st week of April 06.
- What happens to the list of items I own once I've upgraded to a subscription. Do we need to differentiate from items I've listened to under subscription?

OMNIFONE™

Album Groups shown on Get New Albums(#)

This is the set of album groups that are shown on the 51. Get New Albums screen. The exact details of the algorithms used for generating these lists are yet to be defined, but the general intention of each group is described.

Group Name	Description	Filtered to exclude albums the user has listened to (subscription) / own (pay-per-track)
Top albums	The top albums across the whole system instance (I.e. an installation of MusicStation for a particular operator). It should be based on all tracks listened to and explicitly rated by all users. It is not personalised to the current user.	No
You might like	Albums that we recommend to the user based on their (recent) listening habits, and taking into account any explicit ratings that they have given. Filtered to exclude albums the user has listened to (subscription) or owns (pay per track)	Yes
Just released	The same as Top Albums but only those with a release date within the last two weeks.	No
Top pop Top rock Top dance Top R&B/rap Top alternative	As for Top Albums but filtered by genre. Lists the top 20 from each genre.	No
Recently Added	A list of (back catalogue) albums which have been recently added to the system. Even those are new to the system they could potentially be old back catalogue releases. This list should be based on the user's (recent) listening habits in some way.	No
Featured Albums	A list of albums which have been editorially pushed for promotion.	Yes
All Albums	*This is not actually a group, but instead displays the 60. Simple Search screen with the Albums radio button pre-selected*	N/A

Not v1

Considerations

- Include a album chart for the week and an all-time album chart.
- Recently Added could be a hierarchy of artist/album etc
- Consider detailed rules for each of these lists.
- Need to define logic we use in classifying an album as listened to/owned for purposes of filtering out from these lists.

We need to consider what is shown to a user when they first start the app and have no history to generate recommendations from.

OMNIFONE

52: Get New Albums– Albums Group (e.g. Top Albums

This screen shows the contents of the selected Album Group like Top Albums. The definition of the contents of this screen is given on the previous slide.

We need to have an intelligent system of recognising that a user has had certain albums listed enough times for them to take what they want from them but for now on the demo it seems reasonable to have albums in this section where the user has already purchased one or two tracks (SteveG)

42. Get New Albums – Albums Group

Softkey
- Open
- Now Playing – go to Play Track (if playing)
- Play/Pause (Current Playlist)
- Main menu
- Close

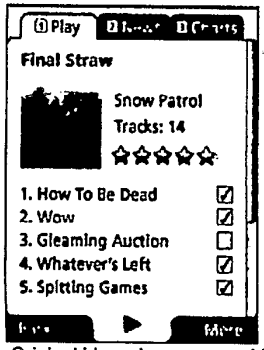
Navigation	
◀◀	Back/Top menu
◀	Back
▲▲	Up item
▲	Up item
●	Open artist
●●	Open artist
▼	Down item
▼▼	Down item
▶	Open artist
▶▶	Open artist

OMNIFONE

55: Album Page √

The **Album Page** is the screen which collates all the tracks in an album. Users can deselect tracks from an album so that they don't play and aren't included in playlists.

Help Popup



Original idea where you could turn of f certain tracks in an album

Album image if we have it, otherwise artist image if we have it, otherwise default image.

Album name.

52. Album Page

This is artist name and the number of tracks the system has for this album – not controlled by how many the user has.

Features
- Include other information like release date or reviews?

Album Tracks included:
- All albums start off with all tracks included.
- User can uncheck tracks as they like.
- Wherever used that album only plays the checked tracks.

Softkey
- Play
- Add to Playlist
- Rate Track
- Similar Artists (or should it be artists you might like?)
- Artist News
- Artist Profile
- Now Playing – go to Play Track (if playing)
- Main Menu
- Close

Navigation©	
◀◀	Back/Top menu
◀	Back
▲▲	Up item
▲	Up item
●	Check/Uncheck
●●	Check/Uncheck
▼	Down item
▼▼	Down item
▶	Check/Uncheck
▶▶	Check/Uncheck

Discussion Point
- Consider including "similar albums" links in here – I.e. recommendations seeded from what you are currently looking at.

OMNIFONE

57: Charts

A variety of charts. Likely will be country specific. Each chart has a graphic which represents it visually.

57. Charts

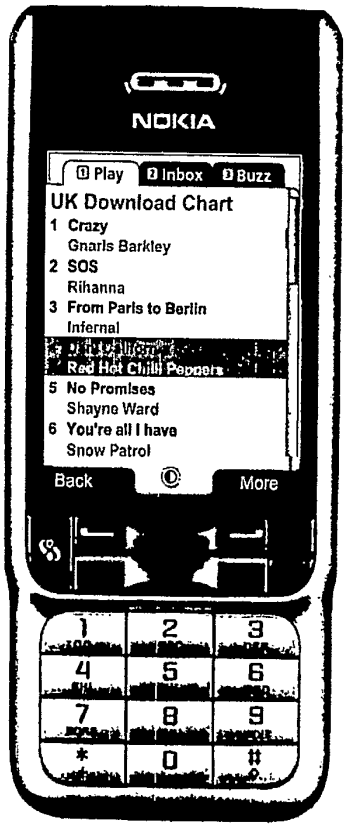
58. Chart Items

Softkey
- Open
- Now Playing (if playing)
- Main menu
- Close

OMNIFONE™

58: Chart Items

A variety of charts. Likely will be country specific. Each chart has a graphic which represents it visually.



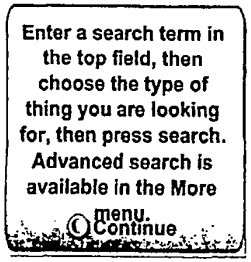
58. Chart Items

Softkey

- Play
- Add to Playing (if playing)
- Add to Playlist
- Rate
- Artist Profile
- Now Playing (if playing)
- Main menu
- Close

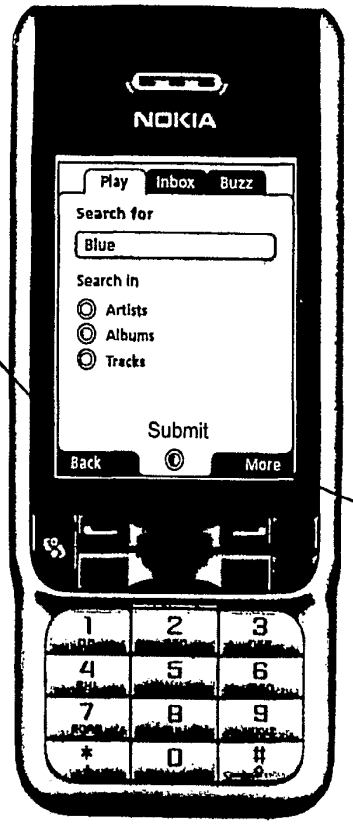
60: Simple Search √

The **Simple Search** screen provides a text based search with control over what it is your are looking for (which defaults to All).



Help Popup

We could advertise **Advanced Search** here at the bottom of the page with a button say.



60. Simple Search

Features

- Can come back to this screen from the **Search Results** screen using **Search Again** softkey.
- Screen has a memory so retains the state of search. There is a rest softkey.
- Predictive T9 text or not?
- What case/numeric/punctuation in SLE?
- Different punctuation char lists on different keys in different phones?
- Is it confusing that "Search" on the sofy key label does something different than "Search" on the main menu? I.e. it performs the search rather than displaying the Search screen. Should it be "Search Now" or something like that?
- We may need to implement some way of implementing a more detailed classical music search

Softkey

- Search (fires search)
- Reset Search (resets to null & defaults)
- Advanced Search (goes to **Advanced Search**)
- Now Playing – go to Play Track (if playing)
- Play/Pause (Current Playlist)
- Main Menu
- Close

Navigation ©	
◀◀	Back/Top menu
◀	Back/Left in SLE
▲▲	Up item
▲	Up item
●	When in text box: Search; When in radio buttons: select option
●●	Select/Search?
▼	Down item
▼▼	Down item
▶	Right in SLE
▶▶	Right in SLE/Search?

OMNIFONE™

61: Advanced Search √

Search For What?
- All
- Artist
- Track
- Album
- Playlists – should allow this

Minimum Rating
- None
- Great
- Good
- Average
- Poor
- Bad

Minimum Rating

☆ ☆☆ ☆☆☆ ☆☆☆☆ ☆☆☆☆☆

Should these be gold? Think not actually as one gold star looks like an award.

To be updated once we have a clear view of the metadata we have available

Does this not show that a 5 star based system is much easier on the eye and brain.

Genre
- All
- Genre1
- Genre2
- Genre3
- ...
- ...

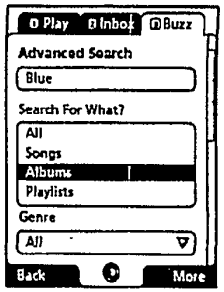
Chart Position
- Any
- Number Ones
- Top Ten
- Top 40
- In charts
- Not in charts
- Somebody??

Search For What? All
Minimum Rating None
Genre All
Chart Position Any
Popularity Any
Language Any
Country Any

Popularity
- Any
- Popular?
- Classic?
- Not Popular?
- Can someone expand on what options & how this is achieved with what meta-data.

As per library meta-data.

Features
- Can come back to this screen from the **Search Results** screen using **Search Again** softkey.
- Once switched you stay on this type of search when you refine.
- SHOULD WE ADD IN HERE DECADE?



Drop-down list box expanded.

61. Advanced Search

Fires off search to the server, see **Search Results**.

Softkey
- Search (fires search)
- Simple Search (goes to **Simple Search**)
- Reset Search (resets to null & defaults)
- Now Playing – go to Play Track (if playing)
- Play/Pause (Play Queue)
- Main Menu
- Close

Navigation	
◀◀	Start of SLE (if keyword input)
◀	Back cursor/Select option (I.e. rating)
▲	Up to select keyword active
▲	Select control (e.g. keyword, rating, etc)
●	Search or DDLB/option activate
●●	Search/No function
▼	Select control (e.g. keyword, rating, etc)
▼	Down to select Popularity
▶	Back cursor/Select option (I.e. rating)
▶▶	End of SLE (if keyword input)

OMNIFONE

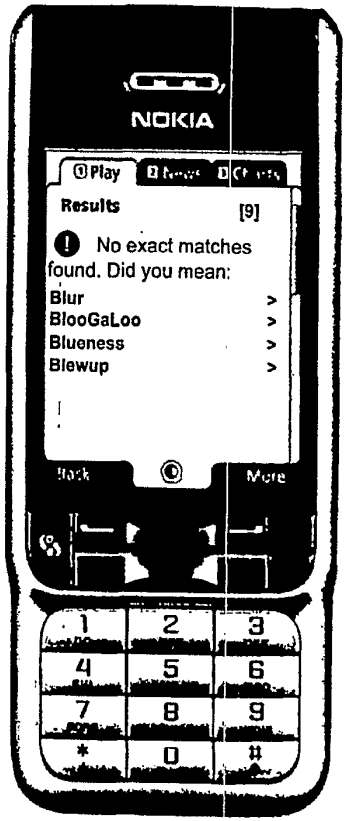
62: No (Exact) Search Results √

When the search returns no results or there are no exact matches but some fuzzy ones, this screen is shown

Zero results found

62. No Search Results even with fuzzy or soundex matching

Use this screen when the only matches which we could find were through the use of SOUNDEX or Fuzzy matching logic.



62a. No exact matches but Fuzzy or Soundex found some results

OMNIFONE™

63: Mixed Results √

The **Track Results** screen is displayed when the user has chosen to have Tracks returned in their search.

This screen shows the matches found for your search. From here you can play music, look at playlists, review the work of artists refine your search.

Continue

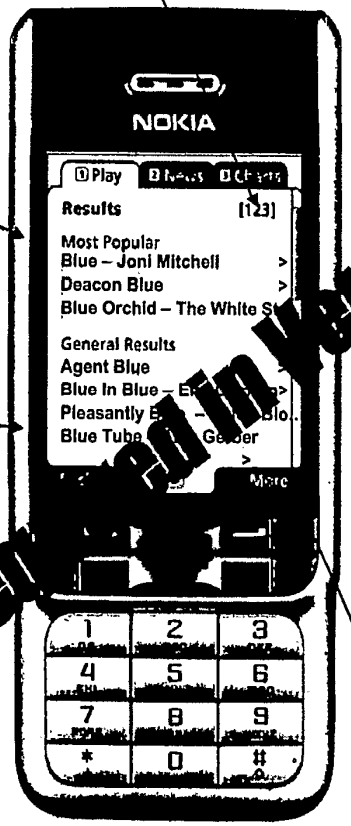
Help Popup

SP 17/7: We will try not include the All option in v1.

Do a **Most Popular** section at the top? Would need us to keeping a rolling popularity with a track so that we don't have to calculate on the fly.

General Results is the main search results. Repeat **Most Popular** in here too – guess so.

Search Results
- If under the limit then just the number
- Else [150 of 375]



63. Mixed Results

Server Search Considerations
- Have to have stop words like "the" – oracle text blade stuff?
- Concatenation fields for cross object search.
- Café = cafe?

Large Results Handling Discussion

We need to decide on a limit to the amount of results we will show in one page. Bear in mind that if we choose 150 then there will be a cleverer thing going on in the background. The comms thread will post back the first part of the 150 results (say 50) and allow the GUI to display them. When the next 50 are down then they are seamlessly added to. Comms & GUI need to shared an understanding of what data's ready to use when.

...could behave differently with different amount of ...s. For example the held down joy might only go fast if ...ere are more than 40 results in the list it's navigating.

Discussion
- Should we mix artists, tracks, albums and playlists in the one list? Or separate?
- There has been some discussion of using Next 50/Previous 50 to break the screen down into "manageable" chunks, but since we can now dynamically add to a screen as the user usees it, there's no real performance advnatage in not displaying all results in the same screen.
- If we do switch to a Next 50/Prev 50 approach then one question to ask is: When the results are split by tracks on phone / tracks not on phone. If the user owns over 50 of tracks they have searched for, then the first screen should just have user results. The second screen would then have some more user owned tracks, followed by a divider and then tracks not on the phone.

Softkey
- Play
- Find (overleaf)
- Artist Homepage
- Sort by Artist/Track toggle
- Search Again (back to simple/advanced search)
- Advanced Search (if in a result list from a simple search)
- Now Playing – go to Play Track (if playing)
- Main Menu
- Close

Navigation	
◀◀	Start of results list
◀	Up one screen of results
▲▲	(Fast) up item highlight
▲	Up item highlight
●	Add to Play Queue
●●	Add to Playlist??
▼	Down item highlight
▼▼	(Fast) down item highlight
▶	Down one screen of results
▶▶	End of results list

When there are more than N (say 50) items in a list, the up/down holds become fast list navigation. When less than N the holds are standard speed. The holds should become fast after a second or two of normal speed.

OMNIFONE™

64: Artist Results √

The **Artist Results** screen is displayed when the user has chosen to have Artists returned in their search and we found some exact matches without having to resort to fuzzy or soundex matching.

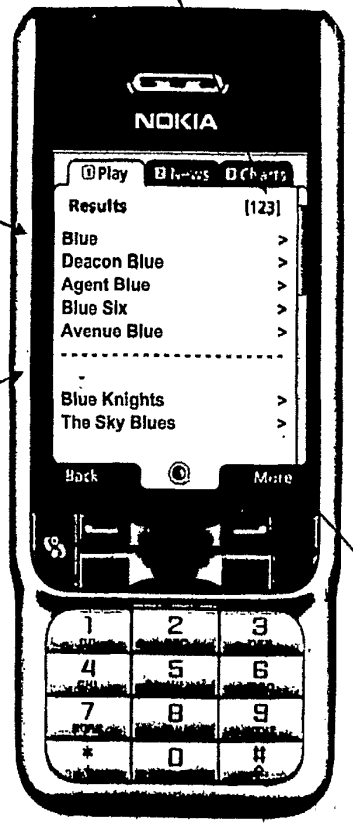
This screen shows the matches found for your search for artists. From here you find an artist and review and play their works. You can also find a phrase in the results list.
Continue

Help Popup

Search Results
- If under the limit then just the number
- Else [150 of 375]

Return the 5 most popular aritists first, ranked by popularity

Remaining result set is displayed below here, ranked ALPHABETICALLY



64. Artist Results

Handling large result sets

When reach maximum number of results that we want to display in a screen (e.g. 100) then display at bottom of the list:

:
Result 98
Result 99
Next 100 >

On the subsequent screen d

Results [100-200 of 375
Previous 100 >
Result 101
Result 102
Result 103
:
Next 100 >

SP 17/7: Implement a solution where the client and server work together to display a window on to a large result set, with results being loaded continuously as the user moves up and down on the screen, rather than using a page by page approach.

Softkey
- Open folder – Artist Homepage
- Find (overleaf)
- Search Again (back to simple/advanced search)
- Advanced Search (if in a result list from a simple search)
- Now Playing – go to Play Track (if playing)
- Main Menu
- Close

Navigation	
⏮	Start of results list
◀	Up one screen of results
⏫	(Fast) up item highlight
▲	Up item highlight
●	Open artist homepage
●●	Open artist homepage/Add to Play Queue??
▼	Down item highlight
⏬	(Fast) down item highlight
▶	Down one screen of results/Open artist
⏭	End of results list

OMNIFONE™

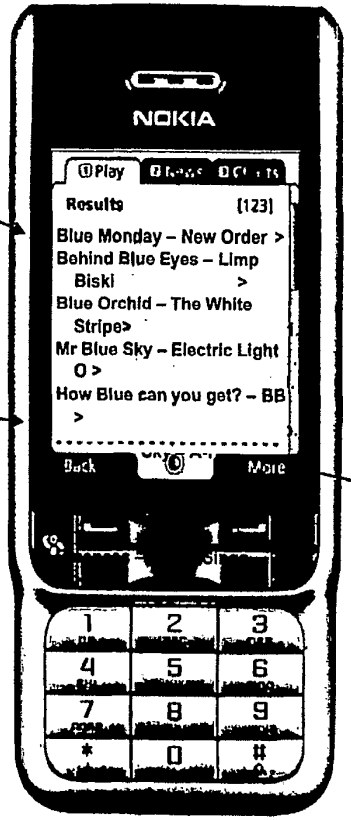
65: Track Results √

The **Track Results** screen is displayed when the user has chosen to have Tracks returned in their search, and we found some exact matches without having to resort to fuzzy or soundex matching.

Help Popup

Return the 5 most popular tracks first, ranked by popularity

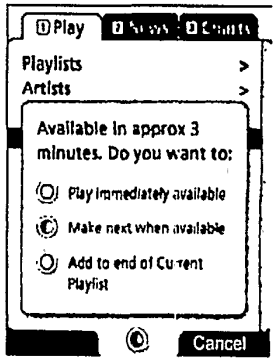
Remaining result set is displayed below here, ranked ALPHABETICALLY



65. Track Results

Features

- Do we need a Track Detail screen beneath this where there's more info about the track.
- Maybe we shouldn't do sort at this stage?



Softkey

- Play
- Add to Playlist
- Find (overleaf)
- Artist Homepage
- Sort by Artist/Track toggle??
- Search Again (back to simple/advanced search)
- Advanced Search (if in a result list from a simple search)
- Now Playing – go to Play Track (if playing)
- Main Menu
- Close

Navigation	
◀◀	Start of results list
◀	Up one screen of results
▲▲	(Fast) up item highlight
▲	Up item highlight
●	Play?
●●	Add to Playlist??
▼	Down item highlight
▼▼	(Fast) down item highlight
▶	Down one screen of results
▶▶	End of results list

OMNIFONE™

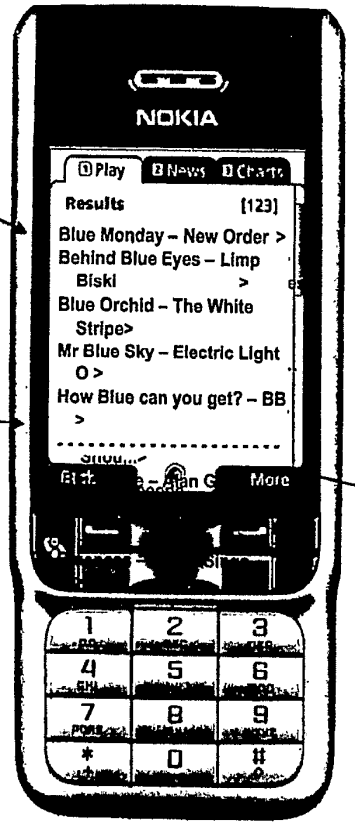
66: Album Results √

*The **Album Results** screen is displayed when the user has chosen to have Albums returned in their search and we found some exact matches without having to resort to fuzzy or soundex matching.*

Help Popup

Return the 5 most popular albums first, ranked by popularity

Remaining result set is displayed below here, ranked ALPHABETICALLY



66. Album Results

Features

- Maybe we shouldn't do sort at this stage?

Softkey

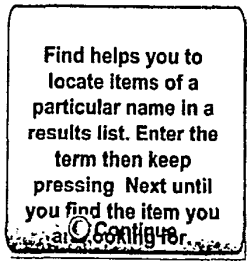
- Play
- Add to Playlist
- Open Album – Album Page
- Find (overleaf)
- Artist Homepage
- Sort by Artist/Track toggle??
- Search Again (back to simple/advanced search)
- Advanced Search (if in a result list from a simple search)
- Now Playing – go to Play Track (if playing)
- Main Menu
- Close

Navigation	
⏮	Start of results list
◀	Up one screen of results
⏫	(Fast) up item highlight
▲	Up item highlight
●	Open Album
●●	Add to Playlist??
▼	Down item highlight
⏬	(Fast) down item highlight
▶	Down one screen of results
⏭	End of results list

OMNIFONE™

67: Find Results √

The **Find** function is a popup off the **Search Results** softkey menu. It allows the user to quickly move through results lists.



Help Popup

Move to next match

Close the **Find** popup

67. Find Results

Features

- When the find popup opens it defaults to the active search term.
- The user presses the Next button to move to the next match in the list
- If long list and next find is way down the results on the server then it skips to the appropriate page of data from the server. Needs server to be working in handshake with the client.
- Could be that all matches on screen are coloured as shown.
- The currently highlighted matching term is a different colour too.

Navigation	
◀◀	Start of SLE
◀	Left cursor SLE
▲▲	(Fast) up item highlight
▲	Up item highlight
●	Find next
●●	Find next
▼	Down item highlight
▼▼	(Fast) down item highlight
▶	Right cursor SLE
▶▶	End of SLE

OMNIFONE™

68: Downloaded: Main

The **Downloaded** screen shows a list of all tracks on the phone available for immediate play irrespective of network connection.

What's On The Phone?
Someone suggested that we show what's immediately playable, what's downloading and what's still in the network. The suggestion was to use colours or icons.
It's possible this is correct. However I am nervous to invent a new GUI widget just yet as I'm not sure whether the information is necessary.

Features
- Downloaded icon
- Active based on connection

70: Now Playing√ (#)

This screen controls and shows the current track which is playing. If a track is playing then this screen appears after XX seconds of inactivity in other screens. The full controls required for the playing process are provided by the joystick alone for a faster slicker experience.

Next Next Next - Fast Track Navigation

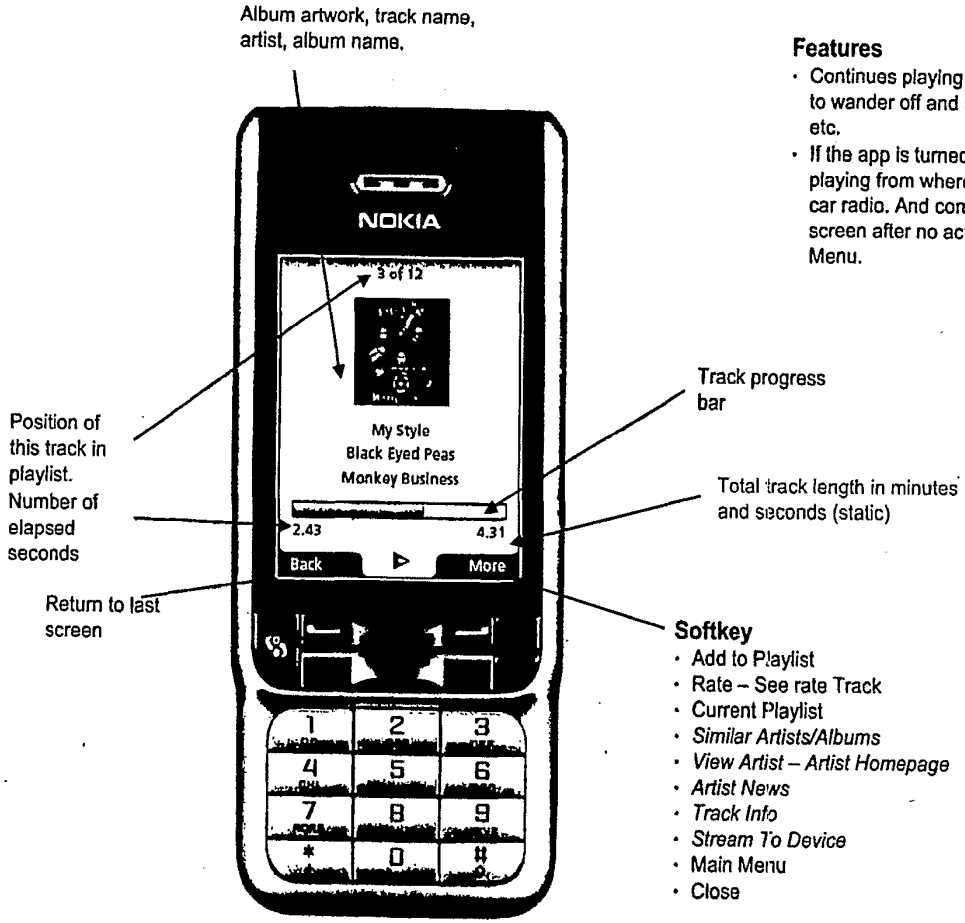
Tracks from the current playlist/album can be navigated very quickly using the joy left and joy right controls. However in order to facilitate a key usability requirement the names of the tracks, artist etc must be displayed near instantly.

User's use this function to scan through music lists and will not like having to wait a second or more before the next track is shown. Album art, track times, animation of the titles etc are not necessary for this quick navigation.

Tab Context of Now Playing Screen

The 70. Now Playing screen is treated a little specially in that it can display on any tab, but it will not be considered as having context on that tab.

If you are on Tab 3 and playing a track then Now Playing screen will appear as if on tab 3. When you press [1] on this Now Playing screen, The last screen that you were in in tab 1 (other than Now Playing) should be the screen you go back to.



70. Play Track

Features

- Continues playing whilst allowing the user to wander off and select their next playlist etc.
- If the app is turned on then it continues playing from where it last left off – just like a car radio. And comes to this screen? Or this screen after no activity for X secs on Main Menu.

Player Contr▶	
◀◀	Rewind track
◀	Previous track (if less than 5 secs into track)
▲	Start of track (if more than 5 secs into track) Volume up
▲	Volume up
●	Play/Pause
●●	Play/Pause
▼	Volume down
▼	Volume down
▶	Next track
▶▶	Fast forward

Discussion Point

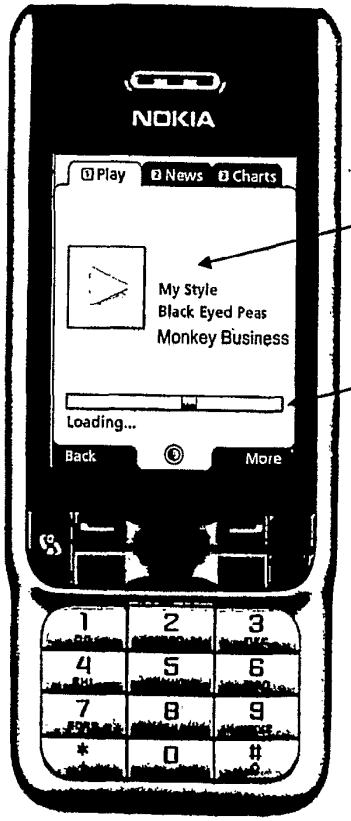
- Before Beta need a discussion whether we provide suggestions seeded from the current track on this screen. Chris E believes this is very important (it's a strong feature of Napster for example). Chris E has suggested one design.

OMNIFONE™

70: Now Playing (Loading) √

Sometimes due to phone limitations there will be a brief delay between a user action which is meant to result in a track starting to play and that track beginning to play. When this happens the user needs feedback that the application and phone have not locked up.

Artist and title displayed immediately, but artwork and track timings not shown until got track playing



Artist and title displayed immediately, but artwork and track timings not shown until got track playing

Moves back and forth while loading.

70. Play Track (Loading)

Volume Cont▶!	
◀◀	Silent
◀	Volume down
▲	Volume up
▲	Volume up
●	Back
●●	Back
▼	Volume down
▼	Volume down
▶	Volume up
▶▶	Full volume

<u>Alternatives which require less Processing Power</u>

We found that displaying this bouncing loading bar had severe effects on the music file loading speed on some handsets so other less intensive mechanisms to consider are:

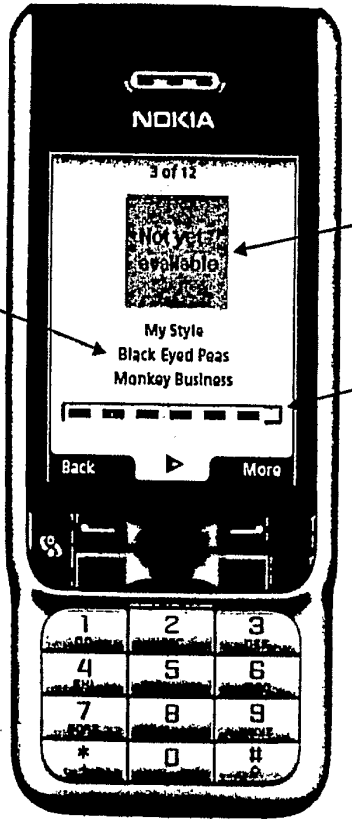
- Use the bar which exists as the loading progress filling up to 100%. This is only an incremental write to the screen and therefore should use less of the phone's resources.
- Otherwise animating the dots on the "Loading..." text

OMNIfONE™

70: Now Playing (Track not Yet Available on Phone) √

This screen is shown when the Current Playlist is playing and the Now Playing screen is shown and the current track reaches a track which is not yet on the phone. When this happens this screen will be displayed for 4 seconds with silence to indicate this. The playback will then continue onto the next track into the playlist.

If the user skips forward or back into a track which is not on the phone and sits there then the same screen is displayed and sits there for 4 seconds or until the user skips forward or back again, whichever is earlier.



Artist and title displayed immediately, but artwork and track timings not shown until got track playing

Some sort of image/wording indicating that this track is not yet on the handset so we cannot play it yet.

Some sort of animation meaning not ready yet to load or play yet. Rob suggested some sort of Mac animation, but this might be too processor intensive.

70. Play Track (Track not yet available on phone)

Volume Cont▶I	
◀◀	Silent
◀	Volume down
▲▲	Volume up
▲	Volume up
●	Back
●●	Back
▼	Volume down
▼▼	Volume down
▶	Volume up
▶▶	Full volume

Alternatives which require less Processing Power

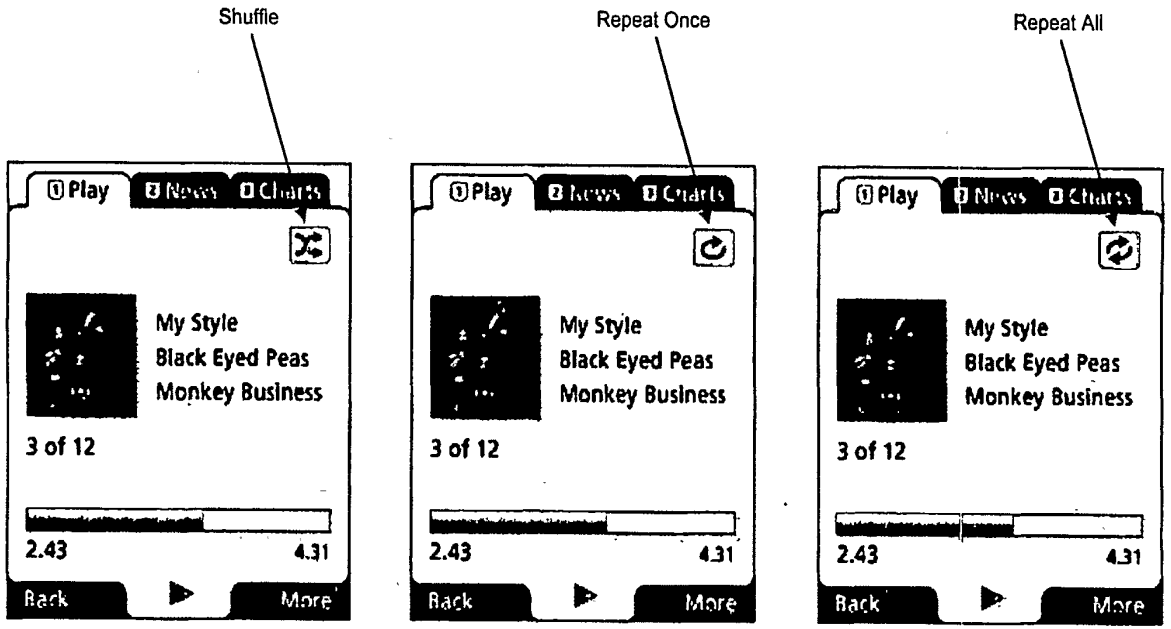
We found that displaying this bouncing loading bar had severe effects on the music file loading speed on some handsets so other less intensive mechanisms to consider are:

- Use the bar which exists as the loading progress filling up to 100%. This is only an incremental write to the screen and therefore should use less of the phone's resources.
- Otherwise animating the dots on the "Loading..." text

OMNIFONE

70: Now Playing (Repeat/Shuffle) √

This slide shows the **Play Track** screen with the icons showing all repeat and shuffle options.

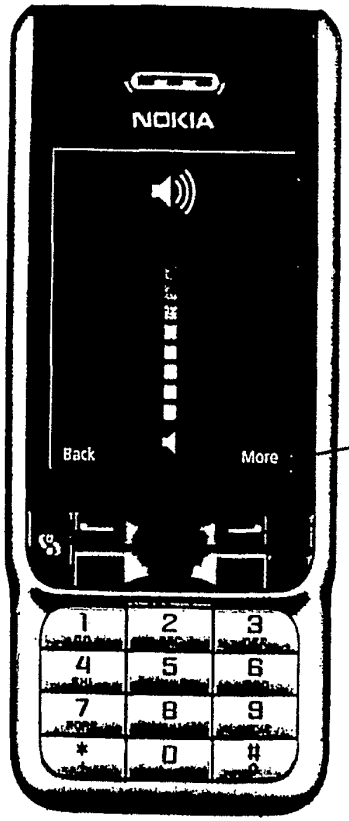


72: Now Playing (Volume) √ (#)

When the joy up or down is used to change the volume the **Play Track** screen changes to show the current volume level.

Features

- After X secs of key/joy inactivity snaps back to normal **Play Track** screen.
- Try to detect the phones physical volume control and show this just as if the joy was doing it.
- Volume must react quickly for usability.
- If sound is muted and volume is changed then mute is turned off.



70. Play Track (Volume)

Same More menu as on Now Playing screen

Volume Cont▶l	
◀◀	Silent
◀	Volume down
▲	Volume up
▲	Volume up
●	Back
●●	Back
▼	Volume down
▼	Volume down
▶	Volume up
▶▶	Full volume

Can't hear the music?

Please check your phone's master volume setting.

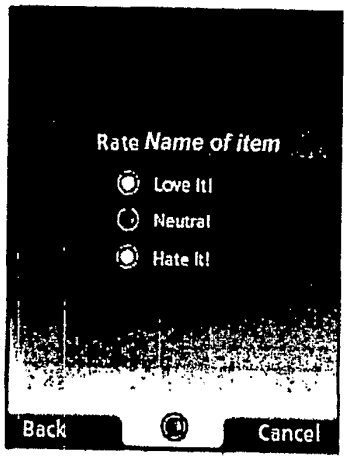
It could be turned down or muted.

When user pushes up from the highest volume setting and music is currently playing then display a full screen popup.

OMNIFONE

74: Rate Track / Album / Artist / Playlist

This popup allows a user to rate a track/album/artist/playlist based on a three way choice. It is displayed whenever the user selects Rate from the soft key menu. Initial value is Neutral.

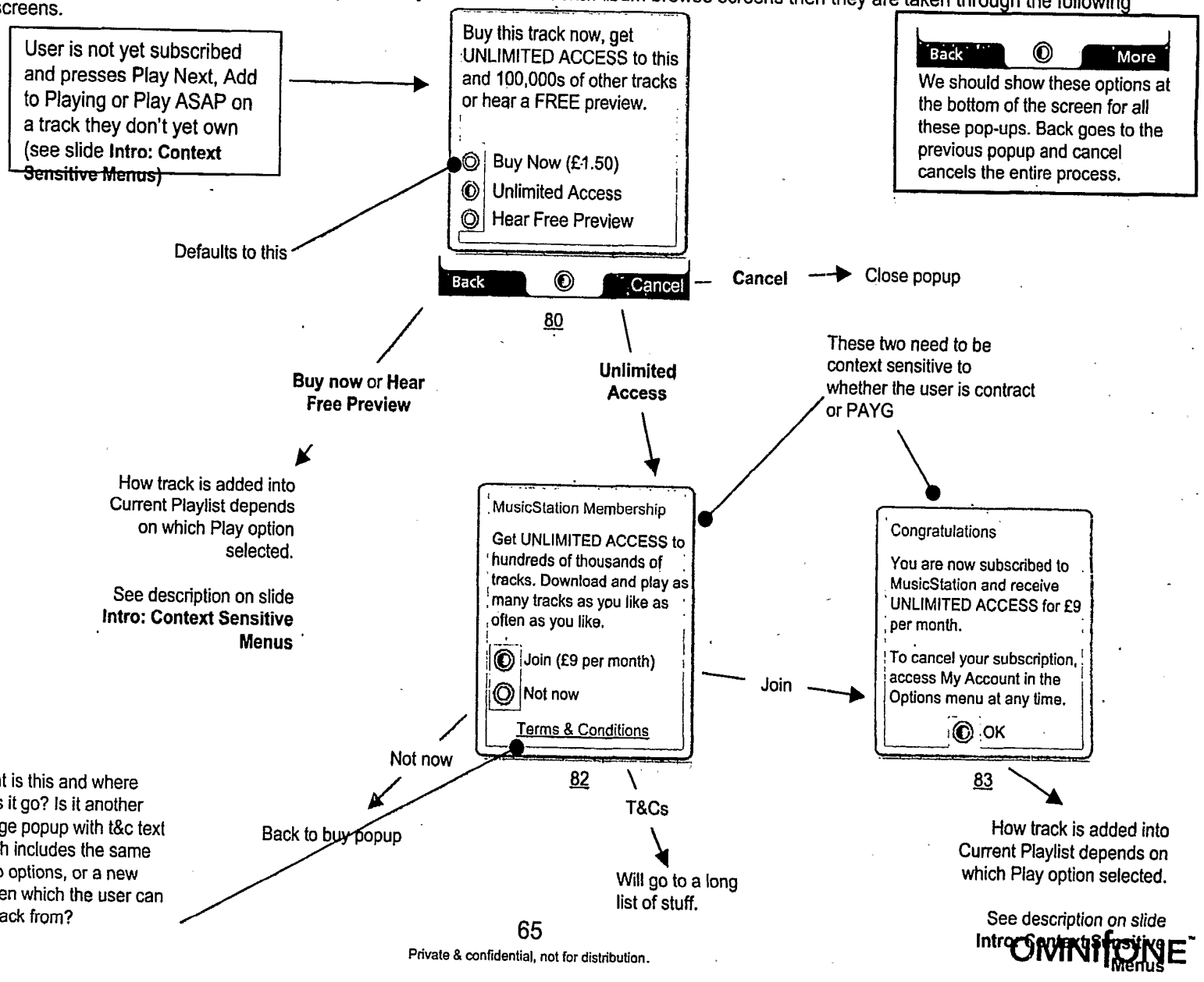


Discussion

- Should we use Like & Dislike + not set?
- Love it/hate it
- Rate artist, album, track.
- Scrap fantastic & dreadful
- Needs cancel with no default
- Colour sync on the button
- Need to be able to see rating as well as rate
- Disclosed only after you have rated?
- Is the rating the group rating or the nearest neighbour only as this will bring down to an average on most stuff eh?
- Go to stars?

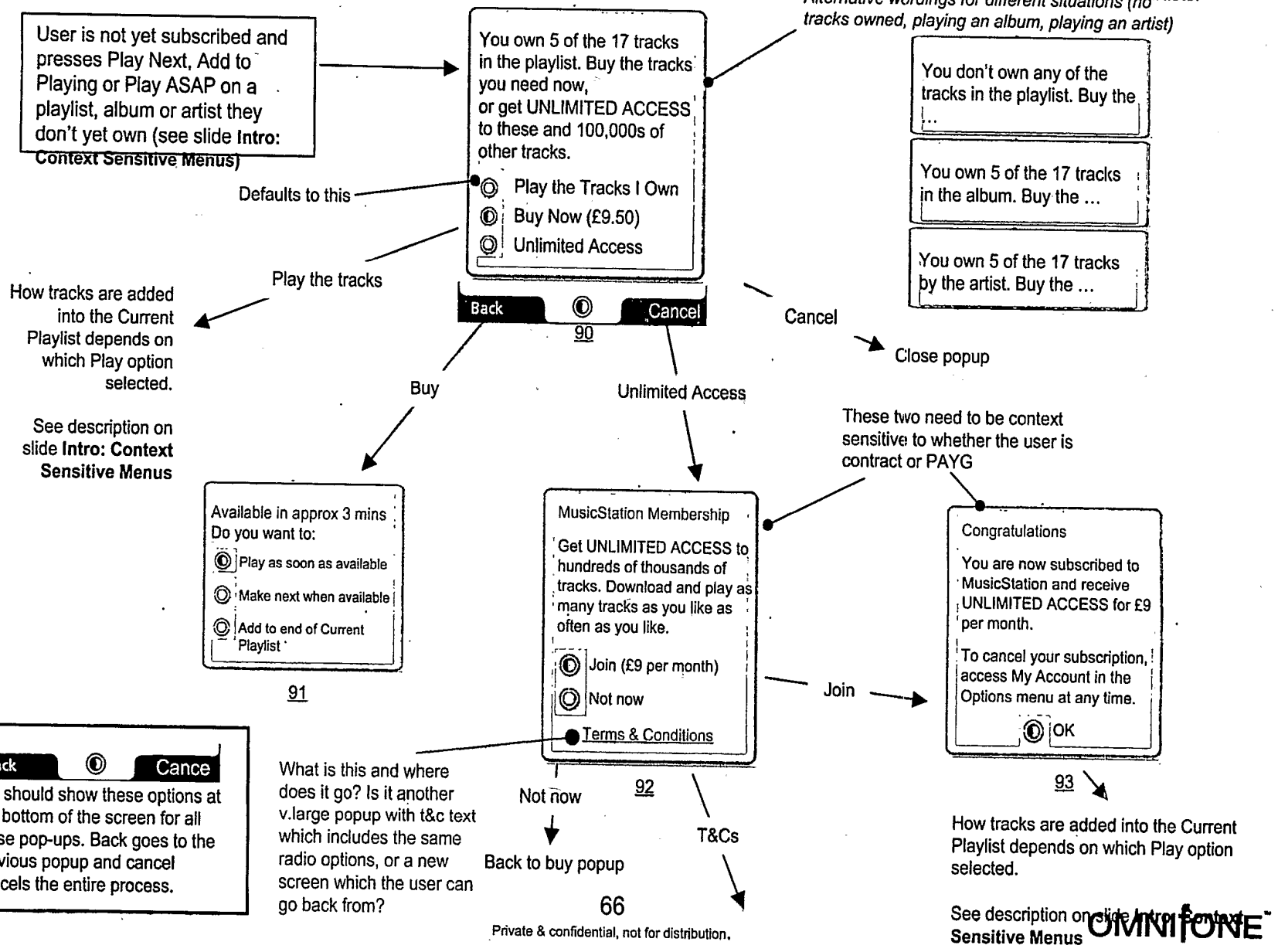
80-83: Buy Track Process (Play or Add to Playlist) (#)

Here is the logic used for a user who is on a blended service where they are allowed to buy individual tracks but can upgrade to a subscription. When the user selects **Play** or **Add to Playlist** on a track inside a playlist or any of the Artist/Tracks/Album browse screens then they are taken through the following sequence of screens.



90-93: Buy Playlist/Album/Artist Process (Play)

Here is the logic used for a user who is on a blended service where they are allowed to buy individual tracks but can upgrade to a subscription. When the user selects **Play** on a **playlist** in a list of playlists on a playlist where the user owns none or only some of the constituent tracks they are taken through the following sequence of screens. The same popup screen applies when the user selects **Play** on an album in a list of albums or artist in a list of artists.



95- Add to Playlist Buy Process (Play)

The logic of popups and screens that are displayed when the user selects Add to Playlist on an item which they do not own yet. Needs to prompt for buy/subscribe etc and If they buy/subscribe then it needs to carry onto playlist building rather than asking the "play next?" questions.

TBC

75: Track Lyrics

The place where we will find the lyrics for a track.

- **Discussion**
  - Karaoke lyrics

76: Track Info

Some MP3 players allow you to see some details about a track – should we do this?

Track Information

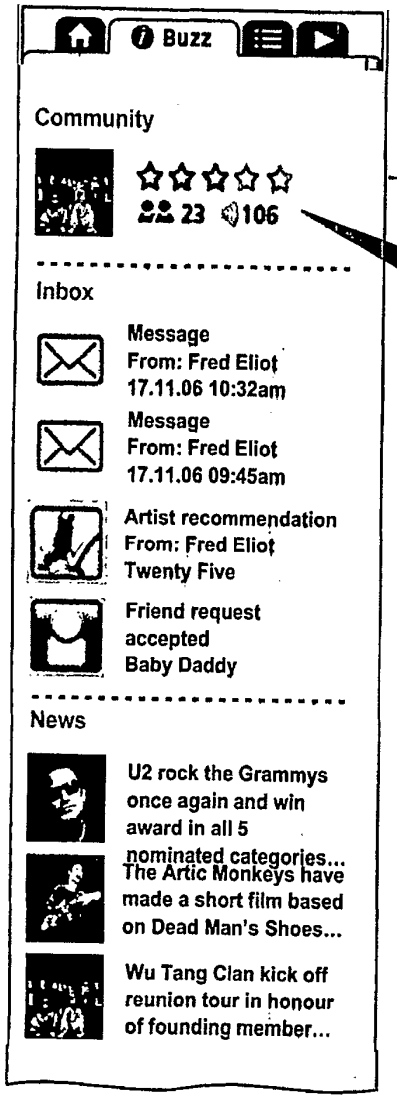
- Available for instance from the Play Track screen via a softkey.
- Includes file name, date received, file size, length (duration) file type (MP3 etc), DRM (distribution)

Feature

- Blah

300: Buzz Main Screen

The new Buzz main screen combines the contents of the Inbox with the contents of the original Buzz tab. It focuses most of its real estate on News Stories as opposed to Buzz functionality since that provides for a stronger graphical look the first time that the user skims through the tabs.



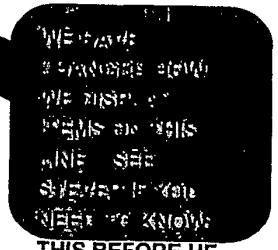
Number of unread messages are displayed here. See below for expiry details.

400: Community Main screen

All non-expired messages will eb displayed here with the most recent at the top.

If this list gets too long we will need to consider whether the older (read) messages are moved off to another screen

We expect a user to have 6 or 7 stories listed here. The selection will be a mix of International and Local general interest stories with a couple of targeted stories.



THIS BEFORE HE UPDATE this doc

310: Inbox Story

Message Expiry

Read messages will be expired **1 day** after they have been read.
Unread messages will be expired **5 days** after the user has been alerted to their existence.
Unread messages will be expired **30 days** if the user has not been in the application and hence not been alerted to their existence.
Expired messages will be removed from the Inbox at a convenient time ... it is not essential that the expiry periods are adhered to exactly. For example they might be removed from the Inbox at the start of the next session after their expiry.

300: Buzz Main

OMNIFONE

310: Inbox Story

A story as displayed when clicking on an item on the Buzz main screen

Artic Monkeys – The Movie!!!

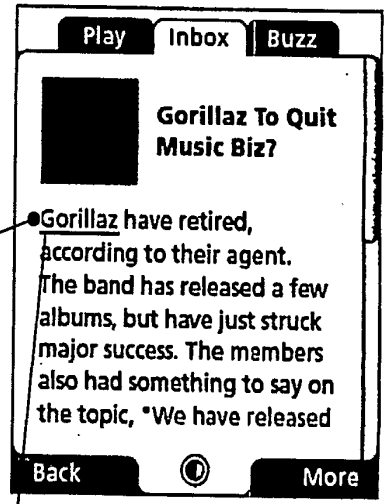
The Arctic Monkeys have made a short film based on their UK number 1 hit single, "When The Sun Goes Down"

The band have teamed up with the production company who worked on 'Dead Man's Shoes' to make two short clips. They tell the story of a prostitute working on the streets of Sheffield - and it's loosely based on the narrative in the band's second Number One single. The video for 'When The Sun Goes Down' contains clips from the film, although the finished movie is longer.

A film source said: "Shooting took place before Christmas. The idea is to tell the story of the song from two different perspectives. The first film is called 'Scummy Man' and is 14 minutes long, and tells the story of a prostitute called Nina, the 'scummy man' mentioned in the title and all the low-life people who live in that world. The second film is called 'Just Another Day' and is shorter, about eight minutes long. It's the film from another perspective."

310: Inbox Story

The currently selected hyperlink is indicated by a blue underline under it. All other hyperlinks are just shown in blue with no underline.



Inbox story with hyperlinks.
Must have support for links to:
Artist
Album
Playlist
A specific set (e.g. Xmas playlists)

Ideally it would technically support links to any screen though the tools to configure a message to link to any screen would be more complex.

For items which are not currently on the phone and could not have their screen at least partially displayed, then display a Loading... full screen popup while it is downloaded (this gives the user some feedback of their click)

OMNIFONE™

400: Community Main

When the user has provided a member name then the Buzz home page shows details for this member.

- The header on this screen is quite large which will mean there could be a large jump when the user scrolls the screen. We will leave this for now however and see how it feels.

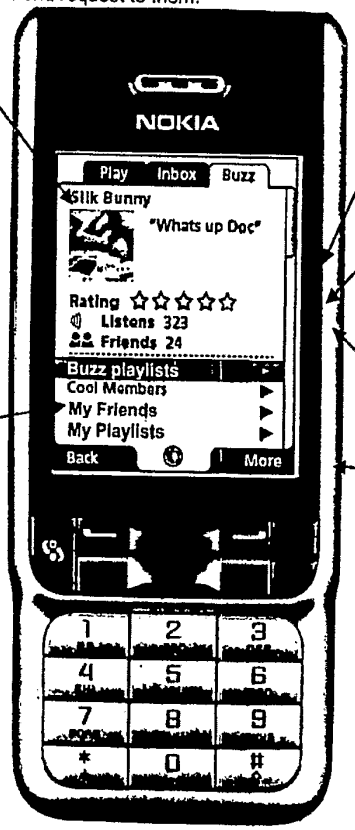
Their member name
Q: Is this clear enough that this is their member name? This is what they need to tell a friend for that friend to send a friend request to them.

Rating is computed from their popularity. The algortihm is included in ChrisE's Recommendation and Rating specification. It should take into account the number of friends the user has.

We want this to display large numbers, so it will show the number of qualifying plays of any track in one of this user's playlists by another user. Plays by this user of their own playlists are **not** included.

Number of friends of the this member.

Menu options give access to playlists (410: Buzz Playlists) and members (420: Buzz Cooll Members) who will be of interest to this member. They also give access to the user's own playlists (10. Playlists Main) and their friends (430: Buzz Friends)



400. Buzz Main (version shown once the user has signed up as a Buzz member)

Softkey
- Open
- **Edit My Profile -> 405. Edit My Profile**
- Close

400. Buzz Main (version shown until the member has defined their member name)

Softkey
- Open
- **Join the Buzz -> 404. Join the Bu**
- Close

Offline Behaviour
We need to consider the dependency of the Buzz screens on being online and minimise such depdendencies. Maybe we use a shorter "no connection" timeout within these screens, and can we queue up any modifications in a queue to be sent when we are nexted connected. Buzz is less depdenent on an immediate connection than the Play tab.

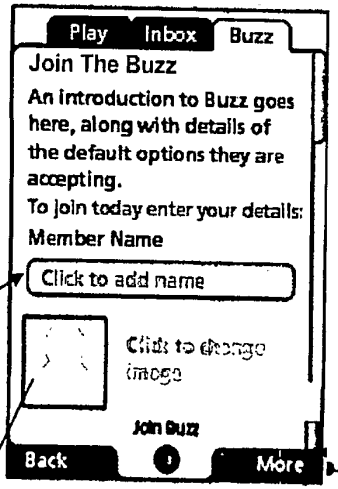
OMNIFONE

404: Join The Buzz (#)

When the user has not registered their Buzz member name and goes to use a feature which requires them to have registered then the following screen sequence is displayed.

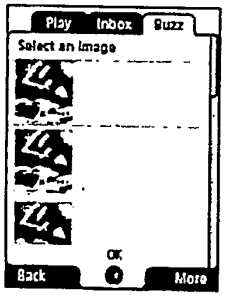
The following screens need the user to have registered with Buzz before they can be accessed:
- 432: Add friend by name
- 434: Add friend by number
- 425: Add selected member as friend
- 440: Send track

Arrive from a screen where a user has selected one of the above options.



404. Join The Buzz

Member name is only mandatory field.



Softkey
- OK – Return to the calling page (i.e. 404 Join The Buzz)
- Close

Softkey
- Join Buzz
- Close

No member name given

Please enter a member name.

Return to the screen, replacing the user entered member name with the suggested member name for the user to either accept or modify.

Display temporary popup for min of 1 sec.

Creating your Buzz account.

If member name is not unique in this service. (Member names are not case sensitive)

Sorry, the member name "zyzigy" is already in use by someone else. "zyzigy3" is available.

Fails stop word filter

Sorry, you are not allowed to use that member name.

Continue

If member name is unique in this service, and passes swear word filter.

Your Buzz account is approved.

Continue

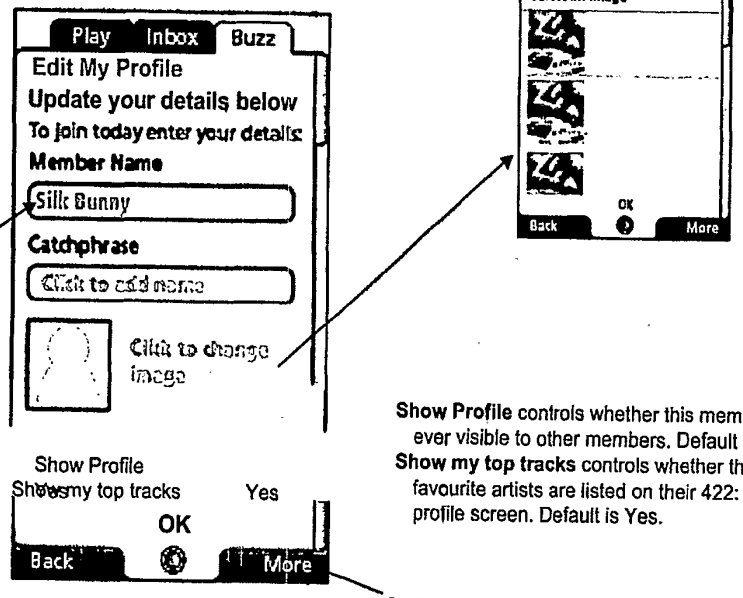
Continue onto the screen/function which was requested before we displayed the Join The Buzz screen. The following can be used straight away:
- 432: Add friend by name
- 434: Add friend by number
- 425: Add selected member as friend

The following would require the user to add at least one friend before they can be used:
- 440: Send track/playlist/album/artist

OMNIFONE™

405: Edit My Profile (Buzz Members Only)

This screen is used for the user to update their Buzz member details. The only situation in which a user can arrive at this screen is when they have previously at least registered their Buzz member name.



Member name cannot be modified once set. It is just displayed greyed out.

Softkey
- OK – Return to the calling page (i.e. 405 Edit My Profile)
- Close

Show Profile controls whether this member's profile is ever visible to other members. Default is Yes.
Show my top tracks controls whether this user's favourite artists are listed on their 422: member profile screen. Default is Yes.

405. Edit My Profile

Softkey
- OK – Return to the calling page (i.e. 400 Community Main)
- Close

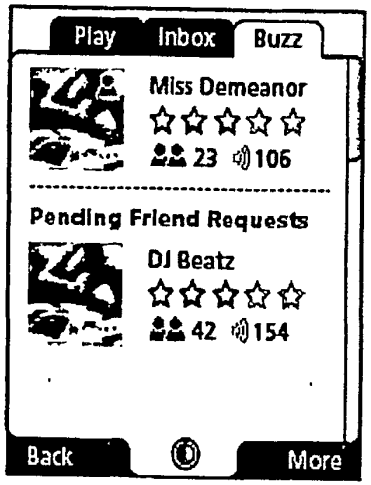
406: Online Friends (V2, low priority)

When friends are online we want to show the user that they are online.

This is because.
1) It makes user more likely to contact one-another if they know they can get a quick response.
2) Encourages users to make quick spur of the moment comments to one-another about, say, Artists they have in common.
3) Really brings the community and 'interactiveness' of the product alive.

Have a look at how MySpace uses it - they always promote online users when returning user lists. I guess this is because many users have not interacted with the system for months, and even years. Not saying that this will be the case with MusicStation, but it is nice to know that someone you message is likely to respond, and positive early experience of this is important in encouraging you to use the community features more....

Technically this is not so hard to implement: the use does not actually have to strictly be online. We should simply say that a user is considered to be online for the time band of their last request to the server + 1 hour.



Early idea: Any friend who has been online in the last hour would have their image displayed with a small red person in the top right hand corner.

OMNIFONE

410: Buzz Playlists

Gives access to shared playlists which will be interesting to this user.

Offline Behaviour
Consider separate loading of the images in the list from the text of the list to limit connectivity and on phone storage requirements for these screens.

410. Buzz Shared Playlists

You Might Like playlists are other users' shared playlists which have been selected for this user by the Recommendation Engine. See the Recommendation engine specification for how this list is generated.

For each we display the Shared playlist name, the star rating and the member who created this playlist.

Softkey
Open – opens the playlist
Add to Playing
Play Next
Play ASAP
Rate
Now Playing
Current Playlist
Main Menu
Close

OMNIFONE

420: Buzz Cool Members(#)

Gives access to members who will be interesting to this member

420. Buzz Cool Members

You Might Like members are other members who are similar to this member. See the Recommendation Engine specification for detail of how this list is created.

Only members who have Show Profile option set on their Edit My Profile screen and are not already my friend will be listed here.

Softkey

Open
Add as Friend (when member is highlighted)
Now Playing
Current Playlist
Main Menu
Close

422: Another Member Profile (#)

View details of another member of the MusicStation service. Displayed when open a member from a list of members, e.g. from the 420. Cool Members screen or when the user uses the Who created? option against a shared playlist. This screen can never be shown for a member who has not yet signed up to Buzz and set up at least their member name. It can nenver be shown for any member who has set Show Profile to false on their Edit My Profile screen.

- The header on this screen is quite large which will mean there could be a large jump when the user scrolls the screen. We will leave this for now however and see how it feels.

Details for this member.

422. Another Member Profile

A list of all of this member's playlists which they have shared.
Opening one of these displays the 12:Playlist Items screen

For each playlist the screen shows the rating and the number of times that a track from that playlist has been listened to with a qualifying play. If there are no shared playlists then the message "**This member has not yet shared any playlists.**" is displayed in this section under the Shared Playlists heading.
This section (including the title) is only shown if the member being displayed on this screen has the "Show my top tracks" flag set on the 405: Edit My Profile screen.

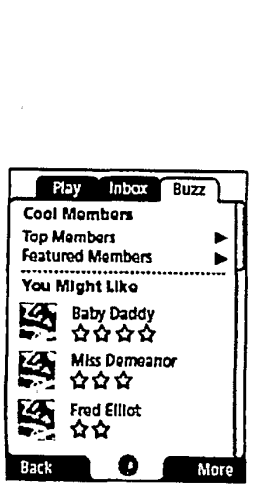
Displays a list of this member's top 5 tracks. This is the all time top 5 most played tracks by this member listed with most popular at the top.
The user can select Play on any of these tracks.

Softkey

Open (for playlists) – Display artist or playlist screen
Add to Playing (for track)
Play Next (for track)
Play ASAP (for track)
Add to Playlist (for track)
Rate (for track) Q:note conflict with rating of this member
Artist Profile (for track)
Send to Friend (for track)
Add as Friend – display 425. Add as Friend to add this member as a friend of yours. (Only enabled for members who are not already my friend.).
Close

425: Buzz Add as Friend (Buzz members only)

This screen is displayed when a user selects **Add as Friend** option anywhere a member is selected. The user can send a message as part of their friend request. This user must be a Buzz member to access this screen.

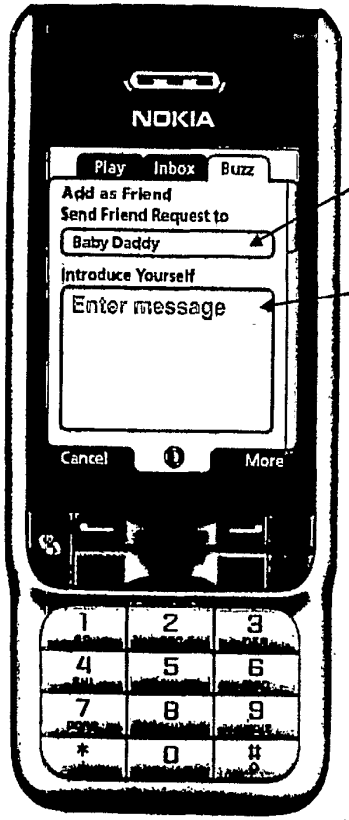


Add as Friend option selected against a member, and the member is not yet this user's friend.

Add as Friend option selected, and member is already a friend of this user

Miss Demeanour is already your friend.

Continue



425. Add as Friend

The name of the member the friend request is going to

The member can enter some text which will be sent to the other member as an introduction as to who they are.

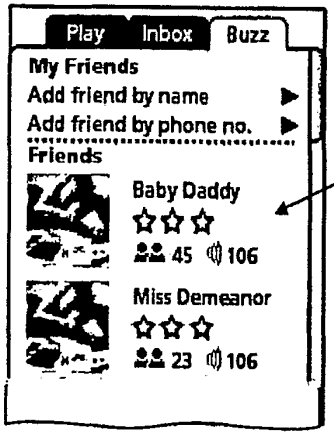
Display popup:

Your friend request will be sent to Miss Demeanour. MusicStation will let you know when there is a reply.

Continue

Softkey
Send
Close

430: My Friends(#)



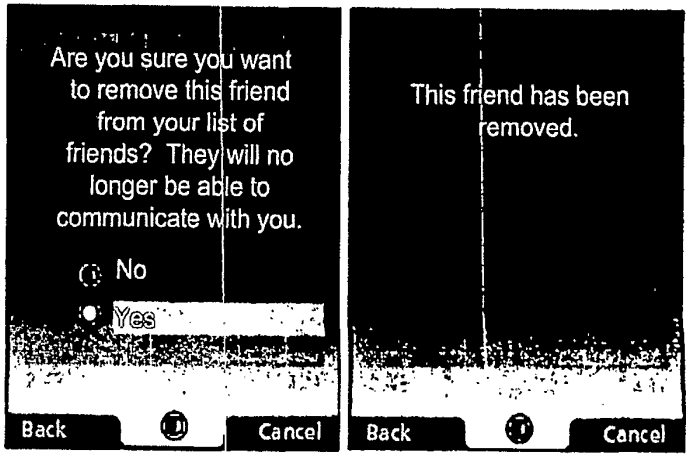
A list of this member's friends. If the user has not yet signed up to Buzz or has signed up but has no friends then this ~~displays the message~~

Your list of friends will be shown here.

If this member is member A and the other member ,member B.

Member B is removed from member A's list of friends and member A is removed from member B's list of friends.

Any recommendations or messages from member B to member A and any messages from member A to member B are removed from the each others' inboxes.



A list of this member's pending friends requests. This title and list is not shown if there are no pending requests.

430. My Friends

Softkey

Open

Remove – Enabled for full friend only. Remove the selected friend

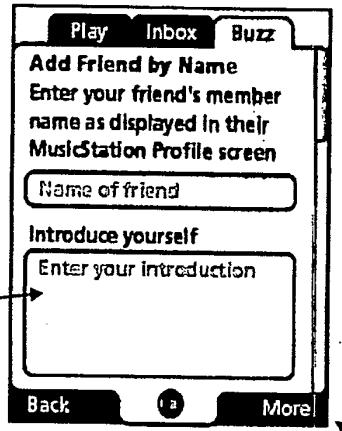
Send message. Enabled for full friend only. Goto 460: Send Message

Close

432: Add Friend by Name (Buzz members only)

This screen is displayed when a user selects a **Add Friend by Name** option. This option is only available if this user has done the Buzz registration and registered their member name. This user must be a Buzz member to access this screen.

- The message should go through our swear word filter and if it fails then it should be blanked before it is sent. We need to be very sensitive to any text which might be visible to other members.
- Members who only want to get friend requests from people they know in the real world could set their Show Profile on Edit My Profile to off and then only people who can ask them directly for their member name would be able to send them friend requests.
- We may not send the request immediately since we may not be online. Also the other member will not necessarily receive the request immediately since they may not be in the application. So all the messages and behaviour need to be based on this potential time lag.



Enter the member name of the friend to send friend request to.
The member name is not case sensitive.
The other member will be able to tell you what their member name is by looking on the 400: Buzz Main or on their 405: Edit My Profile screen.
Q: Are we making it clear enough where their other friend can find their member name?

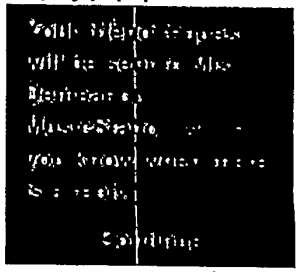
The introduction is limited to **100 characters** (to be confirmed) so that the message can be displayed in the popup on 510: Friend Request Received screen

432. Add Friend by Name

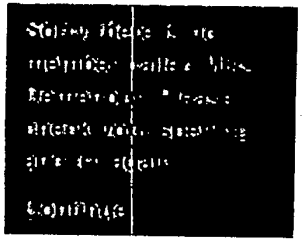
Softkey
Send
Close

Send and a member with that name found (whether that member has their Show Profile option set or not)

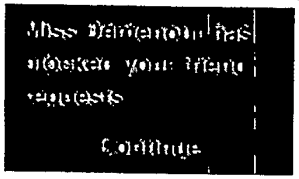
Display popup:



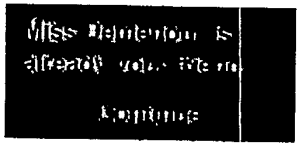
Send and no member of that name found.



Send and that member has previously blocked this member.



Send and that member already this user's friend.

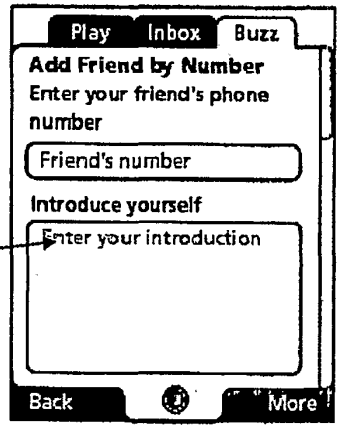


OMNIFONE™

434: Add Friend by Telephone Number (Buzz

This screen is displayed when a user selects a **Add Friend by Number** option. This user must be a Buzz member to access this screen.

- The message should go through our swear word filter and if it fails then it should be blanked before it is sent. We need to be very sensitive to any text which might be visible to other members.
- Members who only want to get friend requests from people they know in the real world could set their Show Profile on Edit My Profile to off and then only people who can ask them directly for their member name would be able to send them friend requests.



Enter the phone number of the friend and some intro text

We will assume that they are entering without the country code and the default country code will be that of the country that the service is associated with.

The introduction is limited to **100 characters** (to be confirmed) so that the message can be displayed in the popup on 510: Friend Request Received screen

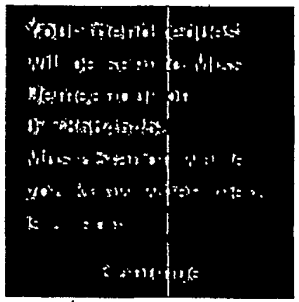
432. Add Friend by Number

Softkey

Send

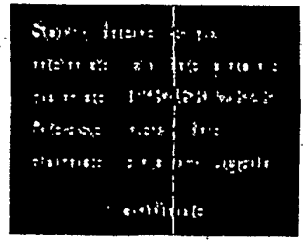
Close

Display popup:

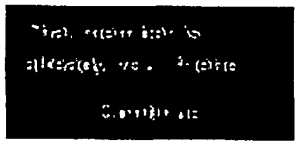


Send and a member with that phone number found in the service (whether that member has their Show Profile option set or not) .

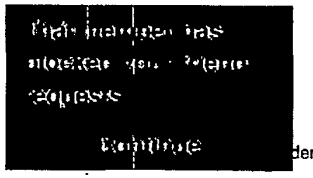
Send and no member with that phone number found in the service.



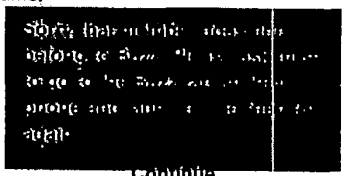
Send and that member already this user's friend.



Sending to a member who has blocked you



Send and a member with that phone number found in the service, but they have not yet signed up to Buzz by specifying their member name.



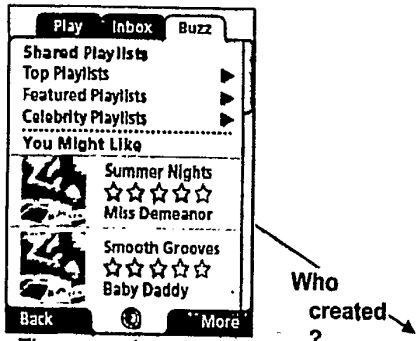
Continue

OMNIFONE™

435: Who created?

A user can select the Who created? option against any shared playlist in the application and is taken to the member profile screen of the member who created that shared playlist.

The Who created? Option is only enabled for shared playlists where the member who created it still has a member profile, and has not unset the Show Profile flag on their Edit My Profile screen. The user must also be a Buzz member.



The user selects a shared playlist somewhere in the UI and selects the More > Who created? Option.

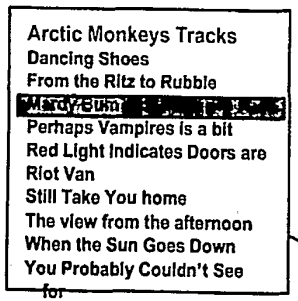
Who created ?

422. Another Member Profile

OMNIFONE

440: Send Track or Playlist or Album or Artist (Buzz

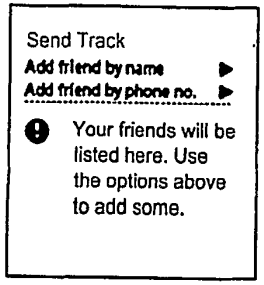
This screen is displayed when a user selects a **Send to Friend** option on a **track, playlist, artist** or **album** anywhere in the menus (see Intro:Context Sensitive Menus 3/4). The user may select **one or more** friends to send the playlist to. The examples of Send Track is shown. This user must be a Buzz member to access this screen.



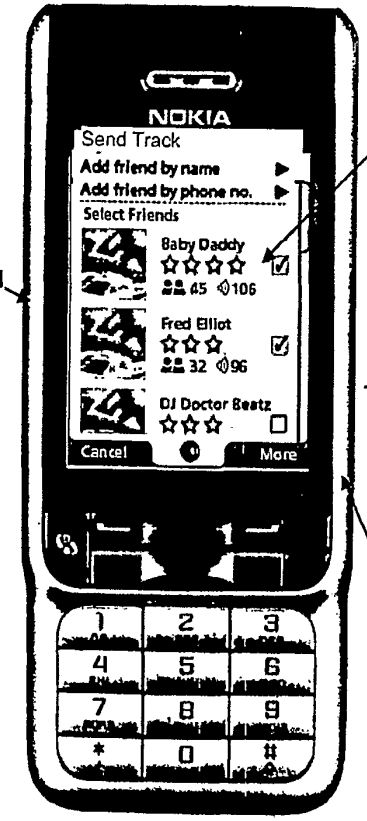
Send to Friend

The user selects a track from a list of tracks or a playlist from a list of playlists and selects More > Send to Friend

The user must have registered with Buzz. If they do not then they would be sent via the **404: Join The Buzz** screen before arriving here.



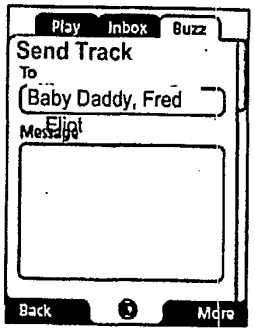
440. Send a Track , playlist, album or artist
(User currently has no friends. This is the screen that the user would see if they had just registered with Buzz after selecting Send to Friend)



440. Send a Track , playlist, album or artist

A list of this member's friends. The user clicks a friend to select and clicks again to deselect. Any number of friends amay be selected. For each friend their rating, number of friends and number of listens are displayed.

Sending a track



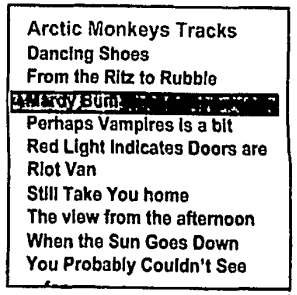
Softkey
Send
Close

Send

Display popup:

The track will be sent to your friend[s] as a recommended track.
Continue

Continue



The user is returned to their initial screen.

Softkey
Send
Close

Popups for other types:

The playlist will be sent to your friend[s] as a recommended playlist.

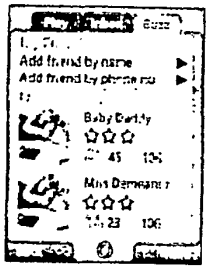
The album will be sent to your friend[s] as a recommended album.

The artist will be sent to your friend[s] as a recommended artist.
Continue

OMNIFONE™

460: Send Message (#) – Lower Priority (Buzz

This screen is displayed when the user selects one of their friends they want to send a message to. Because it is displayed off a command run against friends it will only be seen when the user has registered with Buzz and has **at least one friend.**

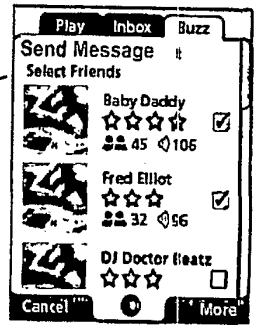


User selects the Send Message option

Send message

460. Send a Message

Press Joystick when focus here to go to Select Friends screen
When Friends have been selected their names are listed here separated by semi-colons
Q: It's not clear to the user that they should do this. You could do as Sony Ericsson do and have an Add Friends option on the More Menu



Softkey
OK
Close

User can type in message here.
Q: How do we handle scrolling off the bottom of this box? The box probably needs to be as large as the amount of text we allow and goes off the bottom of the screen.

How do you scroll on such a screen?

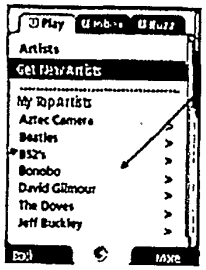
Softkey
Send
Clear
Close

- The message should go through our swear word filter and if it fails then it should not be delivered at all. We need to be very sensitive to any text which might be visible to other members. Operations are very sensitive to this.

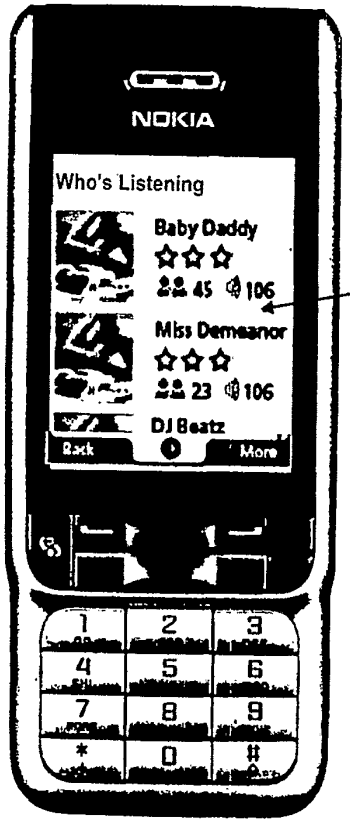
OMNIFONE™

470: Who's Listening

This screen is displayed when a user selects a **Who's Listening** option from the More menu on a **track, album, artist** or **playlist** anywhere in the menus (see Intro:Context Sensitive Menus 1/4 2/4 and 3/4). The screen shows the user some members who have been recently listening to the selected item.



Who's Listening?



470. Who's Listening

Lists the last ten members registered with Buzz who played the item that this user selected Who's Listening option on. What is displayed depends on the type of objects the option was selected against:

Track – the last 10 members to play that track
Album – the 10 members to play a track that is in that album
Artist – the last 10 members to play a track by that artist
Playlist – the last 10 members to play a track from that playlist.

For **album** and playlist **only those users who added a track from specific album or playlist to the current playlist should be listed.**

This function does not necessarily have to be true realtime since the user will not be able to tell whether it is or not. The information could be a little out of date.

Softkey

Open – Go to 422:Another member profile

Add as Friend – Go to 425: Add as friend with member name pre-filled

Close

Who's Listening

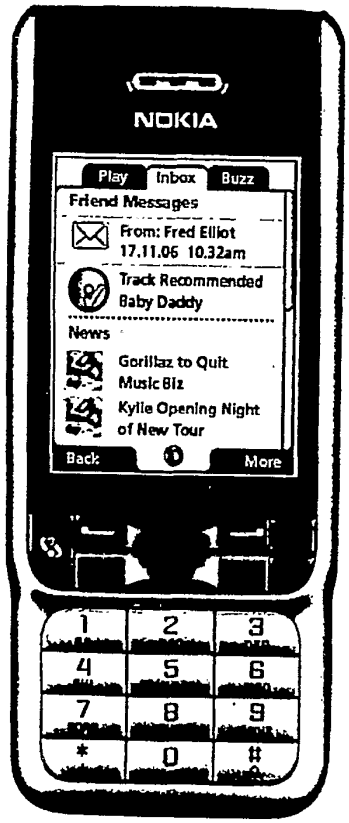
You are the only one who has recently listened to that [track|album|artist|playlist].

470. Who's Listening when no one found who has listened to that item

OMNIFONE™

500: Inbox Track Recommendation Message Arrived

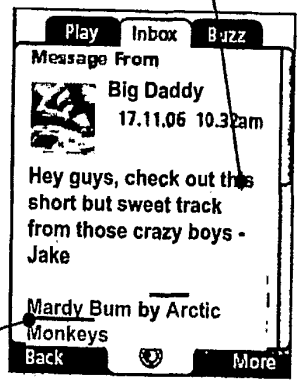
When a member sends this user a track or playlist a message will appear in the Inbox. This user would have to be a Buzz member to receive such a message.



Inbox Messages

Open message

The message Big Daddy entered when they sent the track. The hyperlink function used here is the same as user in other Inbox news stories.



The currently selected hyperlink is indicated by a blue underline under it. All other hyperlinks are just shown in blue with no underline.

Softkey

Reply – go to 460: Send a Message with the TO field pre-filled with the sender of this message

Close

Click on the artist name

Note we switch to the Play tab.

Click on the track name

Same behaviour as if the user had selected Add to Playing on a track in a track list. I.e. the track will be added to the end of the current playlist and a popup displayed.

"This track [is being downloaded and] will be [played]added to the end of the current playlist]."

- As we add recommendations and friend requests to the Inbox it could get a very large screen. WE may want to consider breaking it into sub-menus:

Messages(3) >
Recommendations (0) >
Friend requests (6) >
News (12) >

502: Inbox Album Recommendation Message Arrived

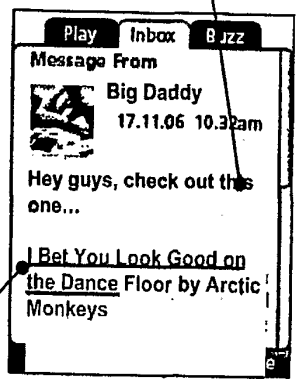
When a member sends this user a album a message will appear in the Inbox. This user would have to be a Buzz member to receive such a message.

Inbox Messages

Open message

The currently selected hyperlink is indicated by a blue underline under it. All other hyperlinks are just shown in blue with no underline.

The message Big Daddy entered when they sent the track. The hyperlink function used here is the same as user in other Inbox news stories.



Softkey

Reply – go to 460: Send a Message with the TO field pre-filled with the sender of this message

Close

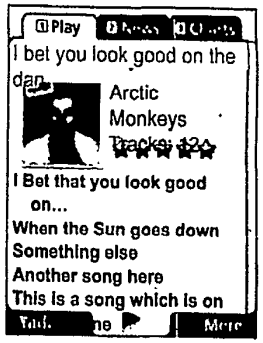
- As we add recommendations and friend requests to the Inbox it could get a very large screen. WE may want to consider breaking it into sub-menus:

Messages(3) >
Recommendations (0) >
Friend requests (6) >
News (12) >

Click on the artist name

Note we switch to the Play tab.

Click on the album name

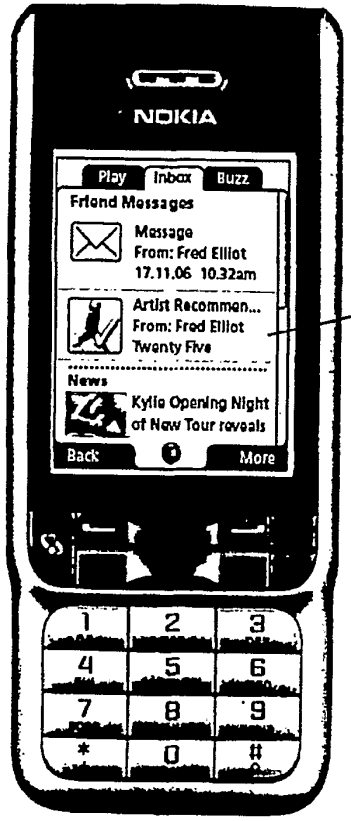


Note we switch to the Play tab.

OMNIFONE™

504: Inbox Artist Recommendation Message Arrived

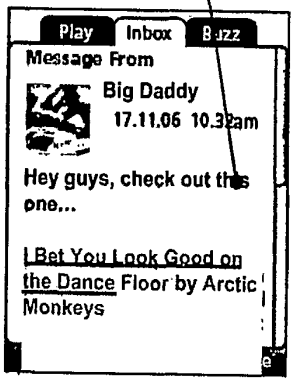
When a member sends this user a artist a message will appear in the Inbox. This user would have to be a Buzz member to receive such a message.



Inbox Messages

Open message

The message Big Daddy entered when they sent the track. The hyperlink function used here is the same as user in other Inbox news stories.



Softkey

Reply – go to 460: Send a Message with the TO field pre-filled with the sender of this message

Close

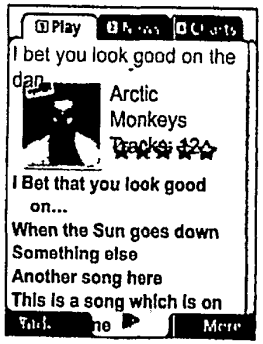
- As we add recommendations and friend requests to the Inbox it could get a very large screen. WE may want to consider breaking it into sub-menus:

Messages(3) >
Recommendations (0) >
Friend requests (6) >
News (12) >

Click on the artist name

Note we switch to the Play tab.

Click on the album name

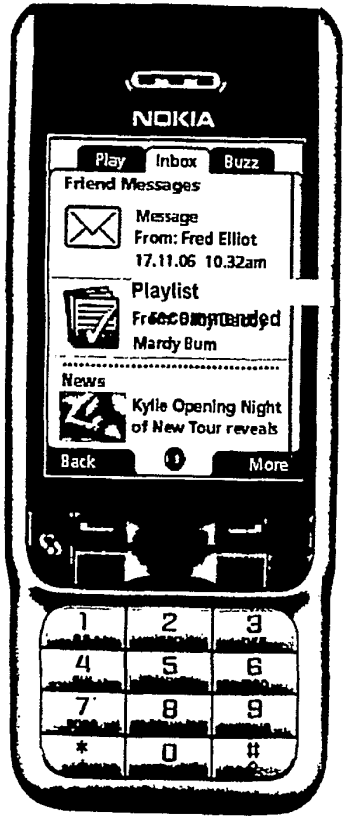


Note we switch to the Play tab.

OMNIFONE

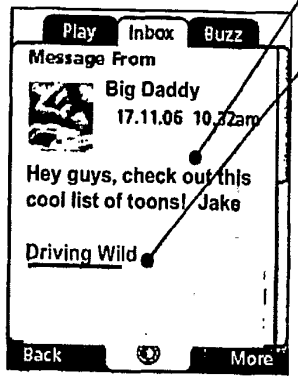
505: Inbox Playlist Recommendation Message Arrived

When a member sends this user a track or playlist a message will appear in the Inbox. This user would have to be a Buzz member to receive such a message.



Inbox Messages

Open message



The message Big Daddy entered when they sent the playlist.

Uses our standard hyperlink mechanism for articles.

Click on the playlist name

Note we switch to the Play tab.

Softkey

Reply – go to 460: Send a Message with the TO field pre-filled with the sender of this message

Close

510: Inbox Friend Request Message Arrived (#)

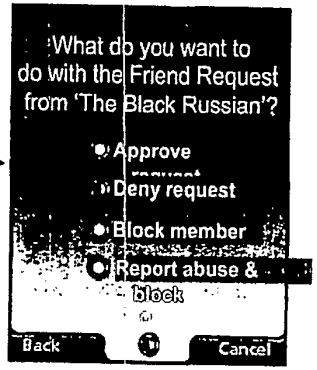
When another member makes a friend request to this member then a message will appear in this member's inbox. When opened they have chance to approve or deny it. We do this within a popup since the user is being asked for interaction. This user would have to be a Buzz member to receive such a message.

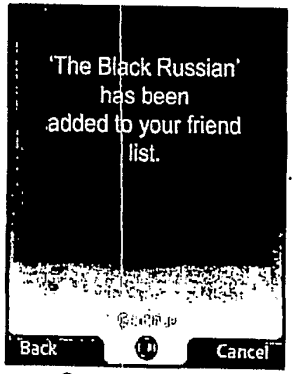
Inbox Messages

Open message

The amount of text that can be entered as part of a friend request will be limited so that it can be displayed in this popup

Continue





'The Black Russian' has not been added to your friend list.

'The Black Russian' has been blocked.

'The Black Russian' has been blocked and the request has been reported.

Softkey

Open

Approve – only displayed if Friend Request highlight (don't grey out since for vast majority of Inbox items this option is not relevant)

Deny – only displayed if Friend Request highlight

Block – only displayed if Friend Request highlight

Report Abuse – only displayed if Friend Request highlight

Now Playing

Current Playlist

Main Menu

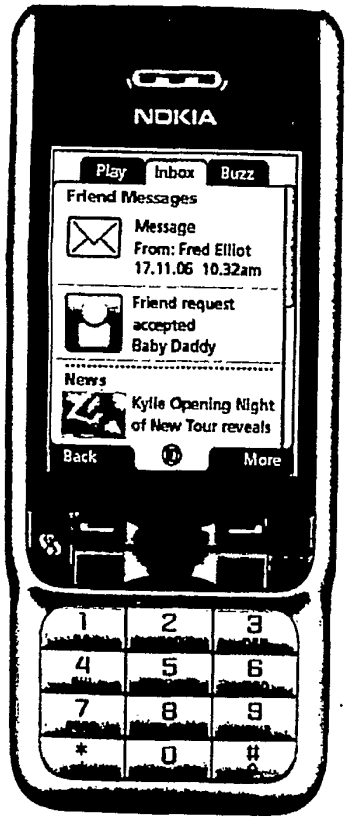
Close

If the friend sending this request had previously had this member blocked, then that friend needs to have this member marked as unblocked again.

OMNIFONE

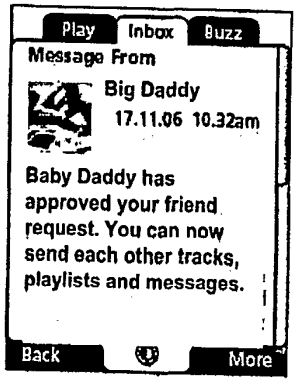
515: Inbox Friend Request Response Arrived (#)

When a member responds to a friend request from this user then that response will appear in this member's inbox. There are three possible responses that this member will see depending on whether the other member **accepted, denied** or **blocked** the friend request. This user would have to be a Buzz member to receive such a message.



Inbox Messages

Open message



Softkey

Reply – go to 460: Send a Message with the TO field pre-filled with the sender of this message

Close

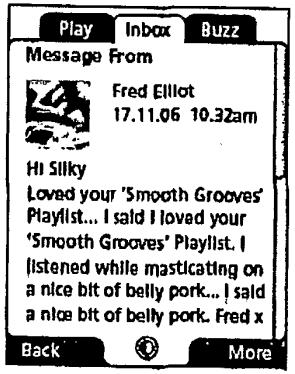
Message Title shown on Inbox screen	Message Content	Result
Friend Request Accepted *OtherMemberName*	*OtherMemberName* has approved your friend request. You can now both send each other tracks, playlists and messages.	OtherMemberName is added to this user's friends and this user is added to OtherMemberName's
Friend Request Denied *OtherMemberName*	*OtherMemberName* has denied your friend request.	No change to either member's friends lists. This member can resubmit their friend request.
Friend Requests Blocked *OtherMemberName*	*OtherMemberName* has blocked all friend requests from you.	This member will have all friend requests to this other member blocked automatically. The other member will never see requests from this member again.

OMNIFONE™

520: Inbox Text Message Arrived (#)

Inbox Messages

Open message



Softkey

Reply – go to 460: Send a Message with the TO field pre-filled with the sender of this message

Close

General Message Alerts and Expiry (#) –

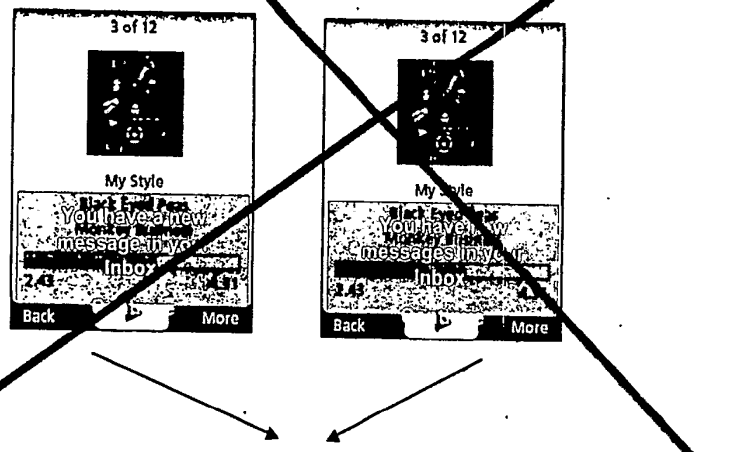
Message Alerts

When a message or messages arrive for this member then we display one of the small popups at the base of the scre once for each group of messages that arrive, and can be displayed as soon as the server has passed those messages to th be checked for and displayed 2 seconds after the client has snapped back to the Now Playing screen so as not to interru urrent Playlist then the popup is displayed 2 seconds after when the client would have snapped back if there had been a Cur

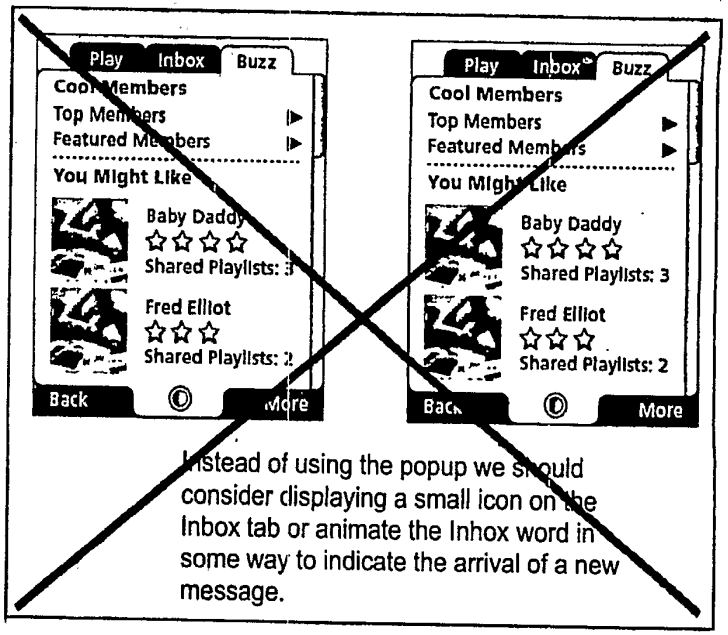
Latest plan is that the number of unread messages will be be displayed against the Messages item on the 300.

After one popup has been read and acknowledged then if any more messages arrive during this session then another popup will be displayed.

Messages sent while this member was offline will therefore likely be displayed soon after they start the application.



Switch to the 300: Inbox screen



Instead of using the popup we should consider displaying a small icon on the Inbox tab or animate the Inbox word in some way to indicate the arrival of a new message.

Message Expiry

Read messages will be expired **1 day** after they have been read.
Unread messages will be expired **5 days** after the user has been alerted to their existence.
Unread messages will be expired **30 days** if the user has not been in the application and hence not been alerted to their existence.
Expired messages will be removed from the Inbox at a convenient time … it is not essential that the expiry periods are adhered to exactly. For example they might be removed from the Inbox at the start of the next session after their expiry.

95: Video

Features

- Go full screen.
- Video clips of yourself.
- How do these work with audio playlists?

OMNIFONE™

100: Options Main

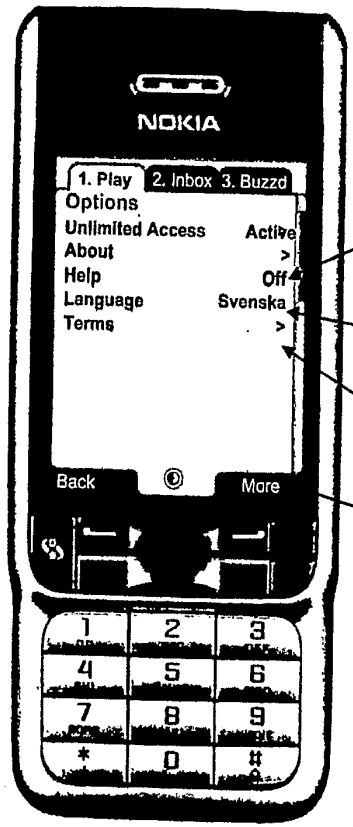
Accessible from the **Main Menu** controls settings and access to other features.

Re-register deletes RMS contents and re-reigsters the client with the server. Refresh Data downloads all data files. Demo modo toggles between online and offline and debug mode controls whether errors are displayed in the client or not and switches the logging level to INFO. View log displays the log data currently on the client.

Hidden Options	
Re-Register	
Refresh Data	
Demo Mode	Online
Debug Mode	Off
View Log	

100a. Hidden Options for development only
Displayed on holding 0 and pressing 7.

Navigation	
◀◀	Back/Top menu
◀	Back
▲	Up item
▲	Up item
●	Select/Open
●●	Select/Open
▼	Down item
▼	Down item
▶	Open
▶▶	Open



100. Options Main

Features
- Opt out of recommendations – under privacy/legal?
- Add history
- My Account needs to be added here (referenced in a couple of popup dialogs for subscription management)

Help is either On or Off. When On displays popup help on entry to each screen. Q: Does not seem to make much sense really.

Rotate between the languages this service is available in. The language names displayed need to be shown in the currently selected language.

Go to 102: Terms

Softkey
- Open (if folder)
- Select (if option control)
- Now Playing – go to Play Track (if playing)
- Main menu
- Close

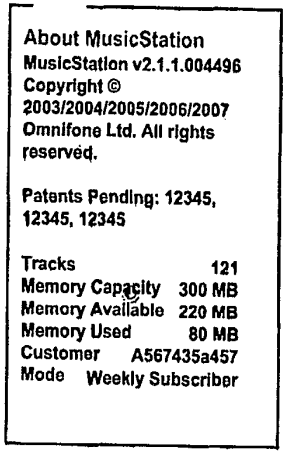
Other Options Main Menu Items
- Autoplay on app launch? On/Off
- Memory settings? Control % of phone memory to use. Control % of memory card to use.
- Tab help On/Off.

OMNIFONE

101: About √

The **About** screen is accessible from the **Options** menu. It shows some key information about the application and it's contents.

Feature
- Patents pending stuff?
- Show a serial number type identifier?



Navigation©	
◀◀	Back/Top menu
◀	Back
▲▲	Up item
▲	Up item
●	No function
●●	No function
▼	Down item
▼▼	Down item
▶	No function
▶▶	No function

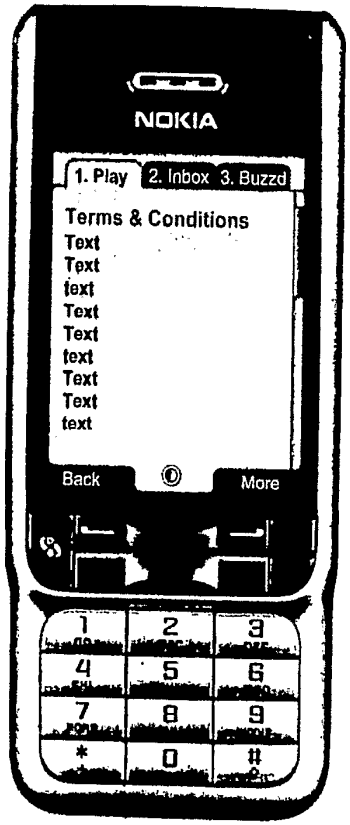
101. About

OMNIFONE

102: Terms & Conditions

We will need a terms & conditions screen.

Navigation	
◀◀	Back/Top menu
◀	Back
▲▲	Up item
▲	Up item
●	No function
●●	No function
▼	Down item
▼▼	Down item
▶	No function
▶▶	No function



102. Terms & Conditions

Feature

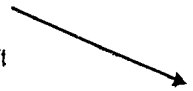
- DX3's agreement stipulate some T&Cs which must be included in the anywhere the content is being sold (I.e. in the application)
- Do we need the user to explicitly agree to any T&Cs or is it acceptable for them to be just included somewhere in the application?

OMNIFONE

105: History Main

Screen where the history is shown.

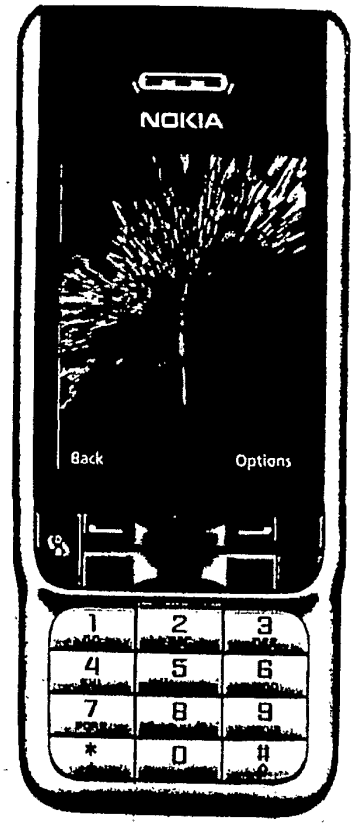
History option appears when there is history. History stored on the server. This is a history browser. Client only stores what it creates, otherwise gets it off the server.



Features

- Today
- Thursday
- Wednesday
- Tuesday
- Monday
- Last Week
- 2 Weeks Ago
- 3 Weeks Ago
- Do we need to go further back and how far with what interface?

115: Fractalizer



115. Fractalizer

Discussion

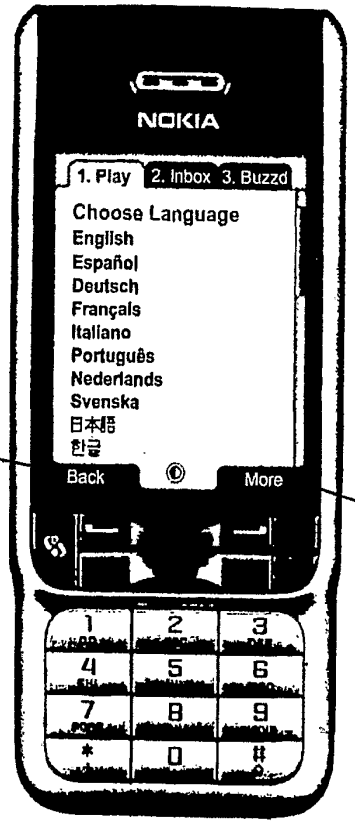
- Will the battery saver spoil all our hard effort in producing this?
- Still very sexy!
- When should it be shown.

120: Choose Language √

The **Choose Language** screen is accessible from the **Options** menu. It controls the language which the application displays text in. It allows the user to change the language.

Language on Start-up

At start-up the application uses the J2ME variables to check the current language setting of the phone's o/s. If it detects that the language of the phone has been changed from what it was to a new language (which we support) then the language of the application is automatically changed.
Or perhaps we should ask them if they want to do this?



120. Choose Language

Return to the Options Main screen.

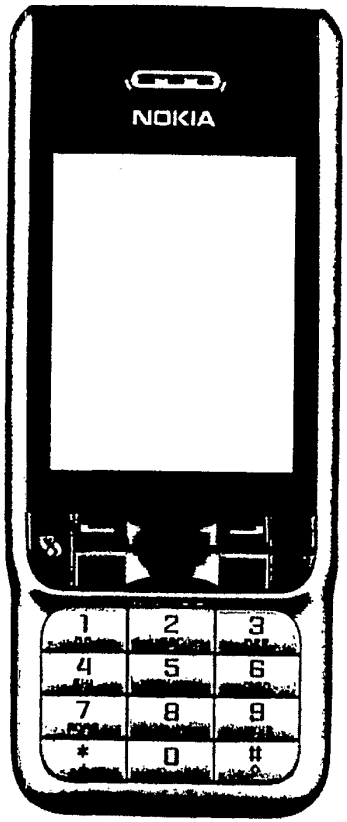
Softkey
- Select (language)
- Now Playing – go to Play Track (if playing)
- Play/Pause (Current Playlist)
- Main menu
- Close

Feature
- Language of the interface is changed dynamically as the language is selected.
- On selecting a language this screen is redisplayed with the title and both softkeys now changed to the newly selected language.
- The language "pack" is either on the client or is downloaded from the network.
- Rob had comments about this and how it should offer options in the current language and the target language (since the user is likely switching from a language they don't know to one they do, or vice versa)

Navigation	
◀◀	Back/Top menu
◀	Back
▲▲	Up item
▲	Up item
●	Select language
●●	Select language
▼	Down item
▼▼	Down item
▶	No function
▶▶	No function

OMNIFONE™

130: Options: My Account



130. My Account

Features

- Presents the member's current status with the serviice using wording like:
- You have not yet upgraded to the full MusicStation service!
- Or, You are subscribed...
- With optoins:
  - Subscribe/Cancel subscription
- More details of what the subscription entails.
- Any identification information like account number/ msisdn/ etc?
- Rob had suggestions for here

OMNIfONE™

140: Options: Downloads

ChrisE's notes on managing the tracks stored on the phone

A scenario where this nigth be particularly applicable is where a user is travelling abroad but wants the benefit of their music - since they may not be able, or want, to use data services in their destination, they would benefit from the ability to cache a list of songs locally.

Here's one way it could work:

1) Add the "Just Download" option to the 'More' menu when relating to a **new album** or **new track** - as touched on in the email:

2) Distinguish tracks that exist in the device's memory from those the user has listened to (subscription model) or purchased (pay-per-track model). This could be done by using bold font for memory resident tracks, with perhaps an option in the **Options menu** allowing for distinguishability between memory-resident and server-resident tracks.

3) Allow **tracks, albums** and **playlists** to be erased from the phone's memory (simple **'More' menu** option). (N.B. For albums and playlists not all tracks may be in the device's memory to begin with, so we just remove those that are).

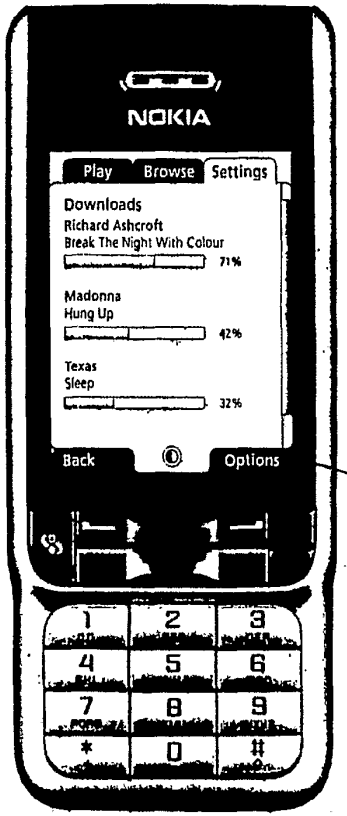
4) Provide an estimate of how many **tracks** (rather than raw memory remaining) there is still room for, and in a more accessible place than Options>About

   e.g. in the top right-hand corner of the Tracks Main screen we could have:

   256/1500

   indicating that there are currently 256 of an estimated 1500 Tracks on the device

Using the above we can ensure that MusicStation users have as good (if not better) an experience than iPod users when they go abroad - I can see people at airports updating their MusicStation catalogue at the last minute, something you could never do with an iPod (unless you have your laptop and CD collection with you!)



Features
- Ability to control the order of the music queue
- Reprioritise – just like that
- Button/bar colour clash

Softkey
- Rename softkey label to **More.**
- More by Artist
- Stop/Resume
- Stop All/Resume All
- Discard
- Play/Pause (Play Queue)
- Close

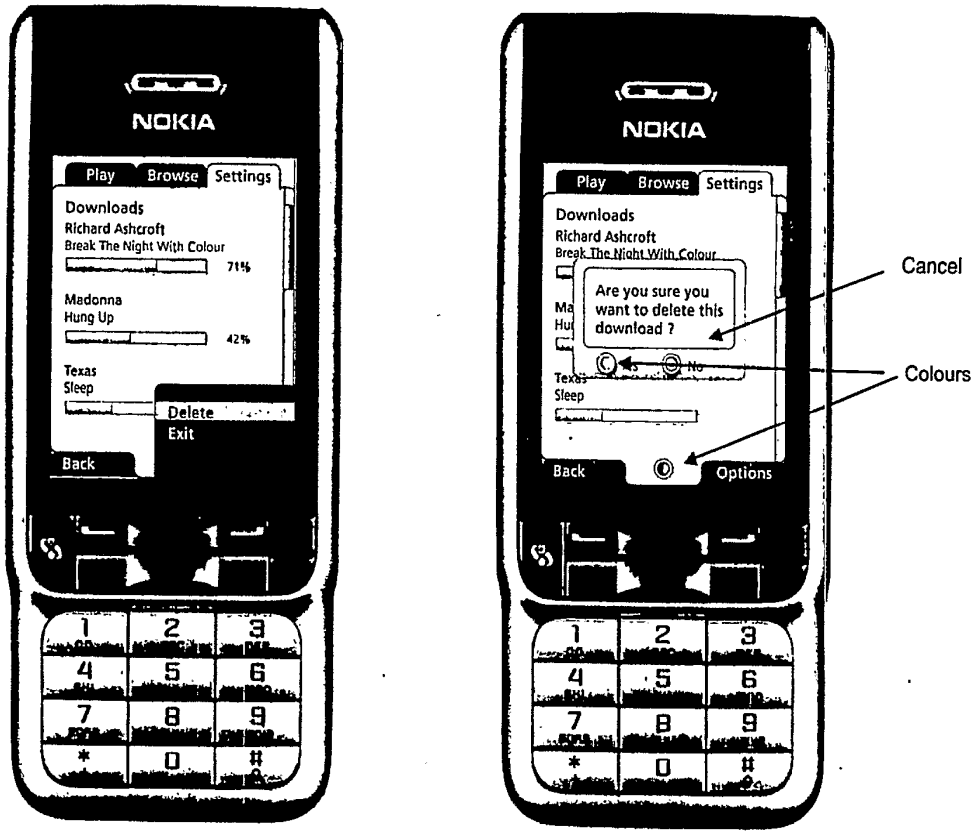
Do we allow the user to manually switch to offline mode or are we always online when we can get a signal? This feature maybe useful whilst the user is roaming and therefore may have pay for data.

When we are offline because there is no signal how often do we check to see if we can reconnect?

How do we communicate to the user that an option is not available because they are offline? Is the option simply greyed out or do we use an icon as well?

OMNIFONE™

140: Options: Downloads (deleting a download)



150: Options: Community

Should this be a main menu item?

Blah
Our automated BlueTooth local app carrier detect
Allow the setting of a bluetooth handle

DJ Radio
- Be a DJ for your mates.
- Rate DJs.
- Automated DJs (based off a profile of a virtual DJ who does neighbour matches to play the top stuff for that genre interest.

Users
- Ability to build a profile
- Set up a user handle
- Take a photo and add to profile
- And title so make them sexy.

Community Playlists
- Listen to other playlists
- Rate shared playlists
- Share a playlist
- Playlist charts
- What are other people listening to?
- Opportunity to listen in.
- Top playlists
- Send message with playlist
- Members collections (Their Music)

Artists
- Artist chat

Groups
- Group rate a track or playlist.
- Group screen shows what who is listening to.
- Private groups (invitation only)
- Instant messaging
- Playlist sharing
- Sync listen to music
- Show if share the same music tastes
- Receive playlist
- Send recommendation
- Chat forum
- Voting
- See others albums, playlists & artists
- Make it easy for you to roll over someone else's and take it all or just some of it
- Make it ultracool in the way it does this
- Set people as mates. MyMates

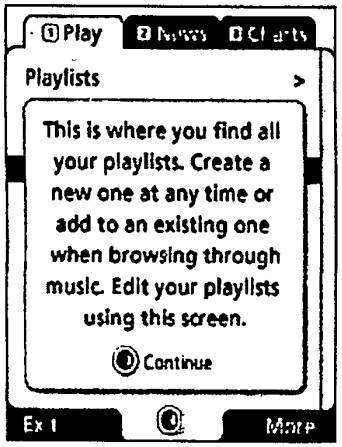
Bluetooth®

OMNIFONE™

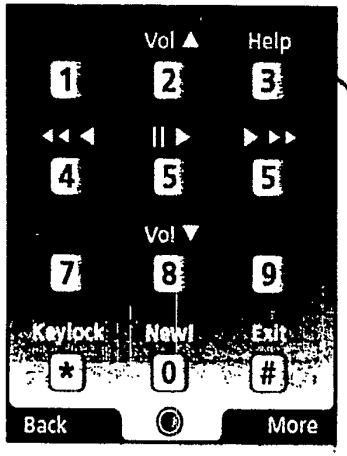
160: Options: Help

First Entry
On first entry to the **My Playlists** screen we could try a little education via a popup (right) which also serves to explain why the only item in here is **New Playlist**.

Help popup on first entry to most screens shown on first entry and also off the help (*) hold.



Keyboard Help
Displayed from an Help option on the More menus. Context sensitive to the screen.



Features
- All help is in the app – or WAP, but not SMS.
- On board help. Audio help with screen control commands embedded. Takes over screen and plays features description audio whilst showing the feature.
- Welcome message audio file. Get a tour. Get going.
- On board help in separate
- Integrated into Options tab.
- Related set of files inside Help in Options. Hidden files unless come through this route or in by context specific.
- Allow retrospective publishing of help patches.
- How Tos when you first start.
- Auto popup if feature not used before and user hesitant.
- One touch help?
- Integrate education into the application. (network based)
- Userbase feedback – how?
- Visual pictures of joystick controls and key controls
- Audio hints on?
- Show picture of keyboard, press button and see what it does.

Consider moving Help to the 1 key. This would then make it available during text input situations such as on the Search screen.

Stream To Device

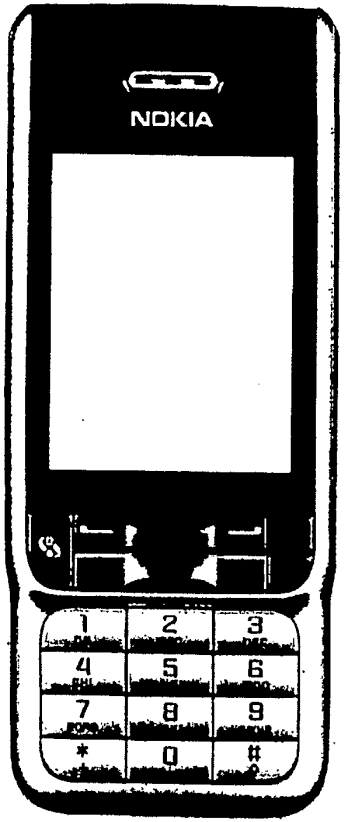
Using the A2DP profile could we stream to a home hi-fi or car stereo?

- Play a stream?
- A2DP BTh streaming

Feature
- Blah

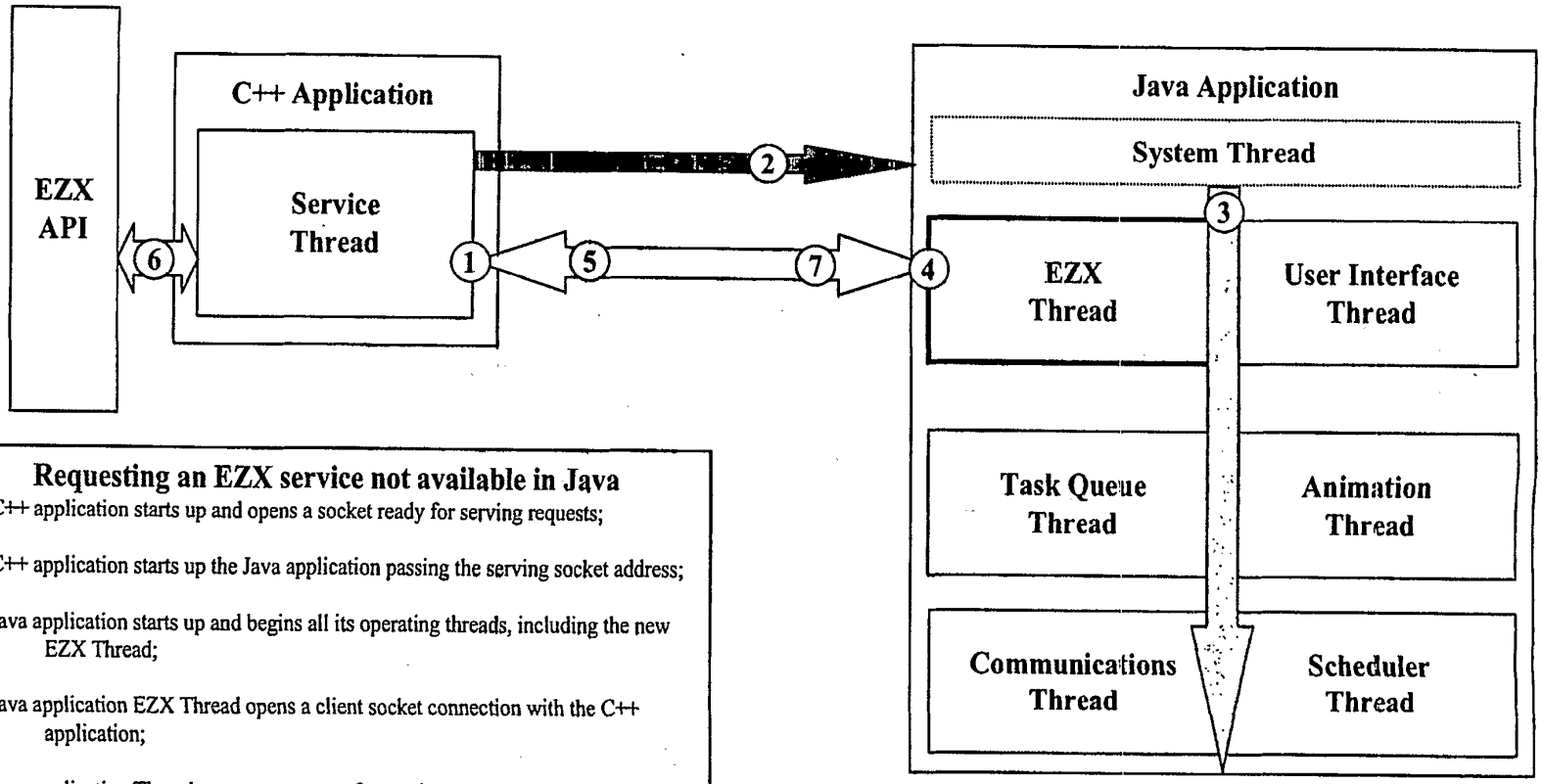
OMNIFONE™

Branding/Whitelabelling



Features

- Background image?



Requesting an EZX service not available in Java

(1) C++ application starts up and opens a socket ready for serving requests;

(2) C++ application starts up the Java application passing the serving socket address;

(3) Java application starts up and begins all its operating threads, including the new EZX Thread;

(4) Java application EZX Thread opens a client socket connection with the C++ application;

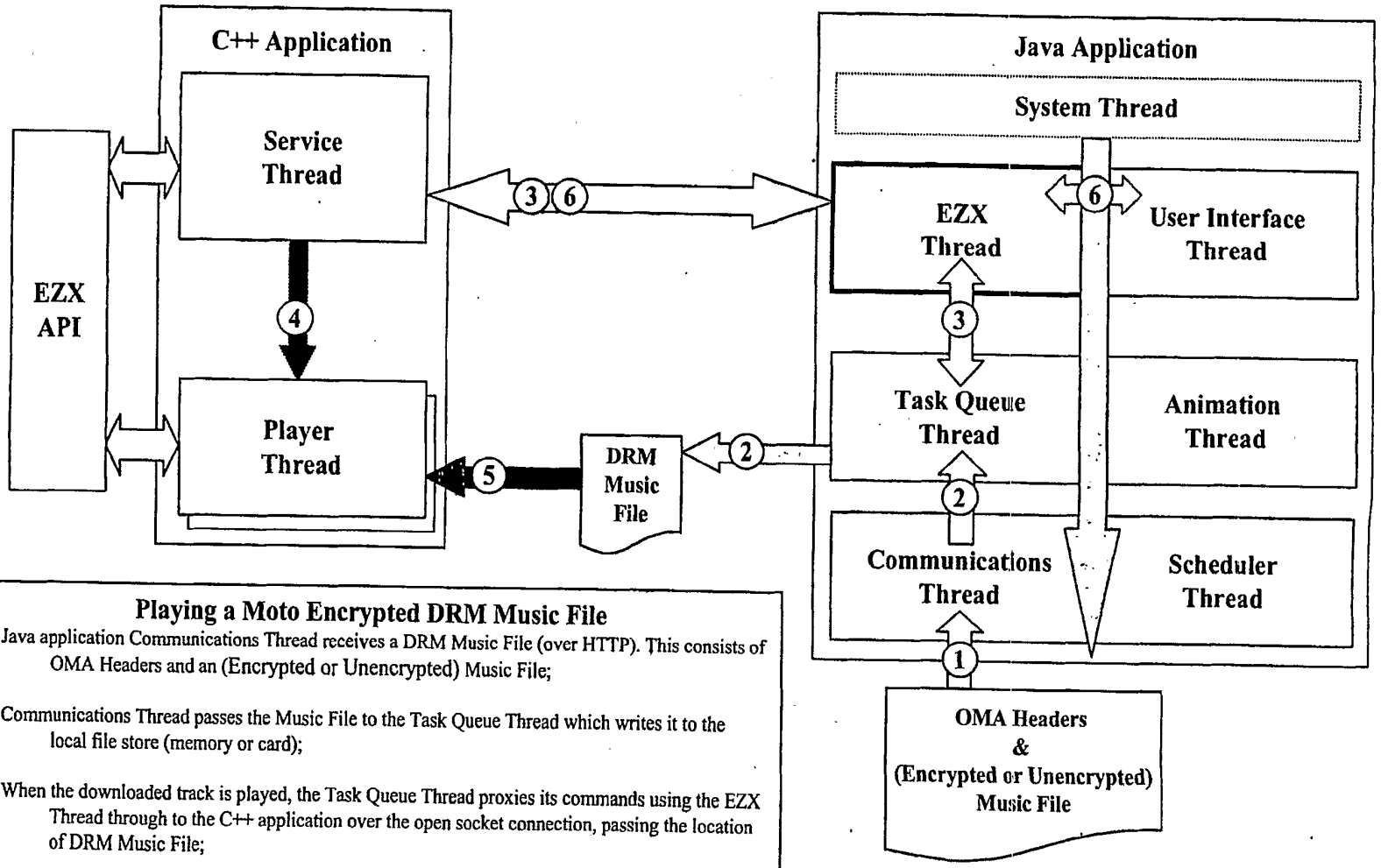
(5) Java application Threads execute requests for services not available to Java using the EZX Thread which passes the request through to the C++ application via the open socket connection;

(6) C++ application Service Thread calls EZX API;

(7) C++ application passes EZX API result data to Java application via the open socket connection.

The C++ application will periodically ping the Java application over their shared socket connection. In the event that the Java application has gone down or is hung then the C++ application will kill any existing MusicStation Java application and re-start it thus assuring continuous operation of MusicStation on EZX.

Motorola EZX: Serving a non-Java Feature



Playing a Moto Encrypted DRM Music File

(1) Java application Communications Thread receives a DRM Music File (over HTTP). This consists of OMA Headers and an (Encrypted or Unencrypted) Music File;

(2) Communications Thread passes the Music File to the Task Queue Thread which writes it to the local file store (memory or card);

(3) When the downloaded track is played, the Task Queue Thread proxies its commands using the EZX Thread through to the C++ application over the open socket connection, passing the location of DRM Music File;

(4) C++ application starts a new Player Thread;

(5) Player Thread starts up and plays DRM Music File through EZX API calls;

(6) When the user selects a Player control (pause etc.) the User Interface Thread proxies its commands using the EZX Thread through to the C++ application over the open socket connection.

Motorola EZX: Separate Delivery by SMS DRM

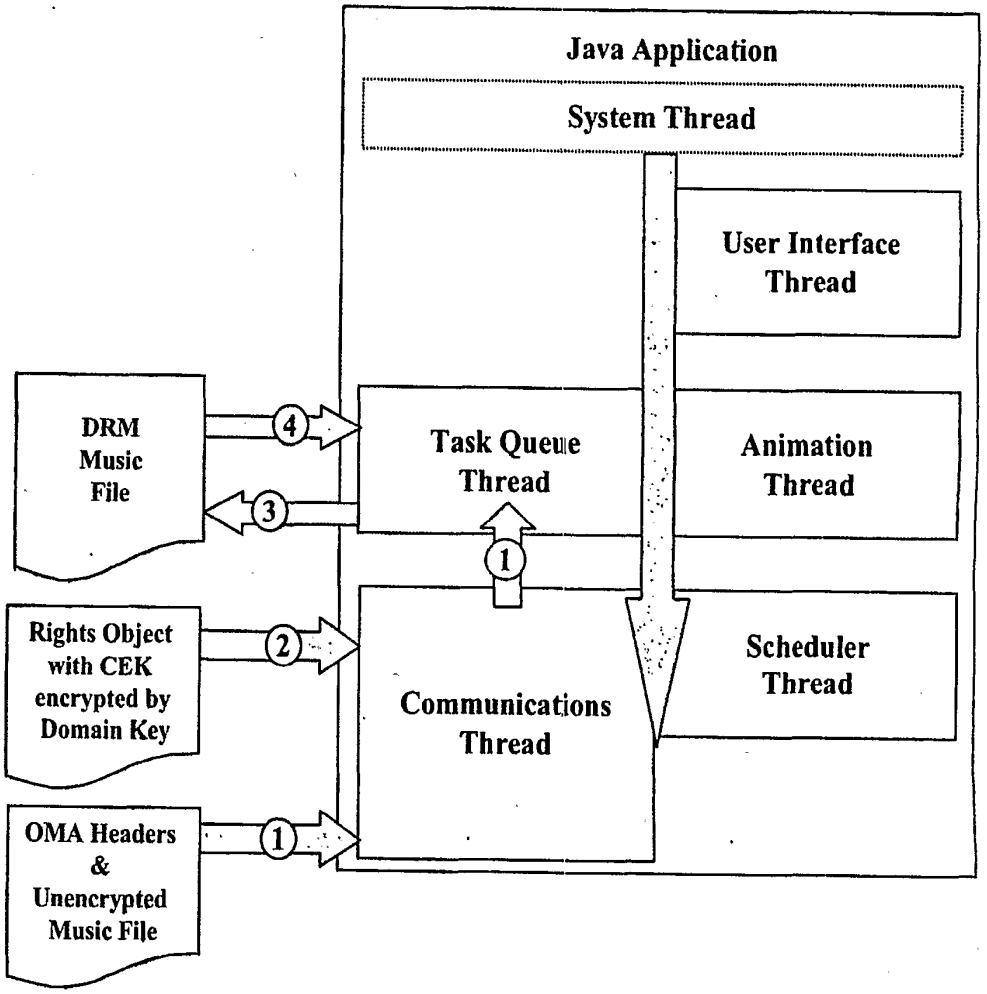
Playing a Moto Encrypted DRM Music File

(1) Java application Communications Thread receives a DRM Music File (over HTTP) from the Motorola Content Servers and passes it to the Task Queue Thread;

(2) Communications Thread receives a Rights Object (RO) from the Omnifone MusicStation Servers. The RO contains a (Omnifone generated) Content Encryption Key (CEK) and is encrypted by the Domain Key (passed to the MusicStation client during automatic registration of the handset on first use);

(3) Task Queue Thread writes the music file it to the local file store (memory or card), encrypting the file as it is written using the CEK found in the corresponding RO;

(4) When the downloaded track is played, the normal DRM of MusicStation comes into operation and the track is decrypted with the CEK found in the corresponding RO which is decrypted with the handset's Domain Key.



Motorola EZX: Single Delivery Unencrypted DRM

From: Sandy Gordon
Sent: 22 January 2007 13:04
To: Steve Pocock
Subject: FW: Calling native services from MIDlets
I have been sent this by mark knight (see quoted below).

Basically it is just creating a peice of server code which auto starts when the phone is turned on and listens on a specific port. Then tha Java client does a localhost socket connection to talk to the server and requests it do do anything that the Java client can't do.

Shall i go and do mark's thing or carry on with my other tasks ??

-----Original Message-----
From: Mark Knight
Sent: 22 January 2007 12:27
To: Sandy Gordon
Cc: Mike Lamb
Subject: FW: Calling native services from MIDlets

Hi Sandy

Please check this out and, if possible, get it working (with a small test java appl) in Symbian. We would want to use a similar method for EZX. The plan being: run a Java client on EZX OS's but have a parallel C++ process which provides 'COM-style' API services to the Java client using the JNI. After you have proven this in Symbian then get it working in very basic form on the Linux box installed with Qtopia. Then install that on a Moto phone with EZX and we will have found our development solution for Moto, i.e. only write those bits we need to in C++ as a service to the normal Java client.

Cheers

Regards

Mark Knight
Director
Omnifone
Island Studios
Office 0208 600 0591
Mobile 0771 774 3921

-----Original Message-----
From: Mike Lamb
Sent: Thursday, January 18, 2007 7:11 PM
To: Mark Knight
Subject: Calling native services from MIDlets

In this ZIP is a PDF which covers MIDlets calling Symbian methods via a Java to Native connection. It would be similar for any native service running on a handset.
It's specific to Symbian, but the same method can be applied to any handset OS.

http://developer.symbian.com/main/downloads/papers/MIDlet_Native_Services_Framework/MIDlet_Native_Services_Framework_v1.1.zip

Mike

Client Data Synchronization

Mark Sullivan – 26 Jun 2006

Table of Contents

1. Scope

This document describes how the user will see a consistent view of data including visible changes made by the user's own transactions and transactions of other users.

2. Introduction

Most screens in MusicStation are populated by data. This data is transferred from the server and stored locally in files on the client. When data on the server changes the files on the client need to be updated to reflect those changes. Also, the user is able to create and modify files on the client, for example adding tracks to a playlist. These changes need to be reliable communicated back to the server.

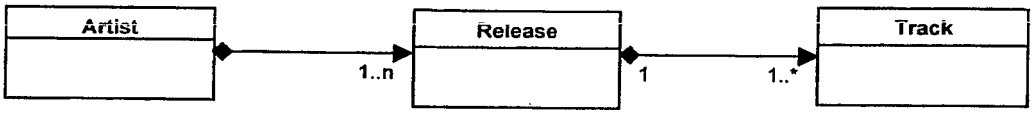
The user can also make changes to data through MusicMate. These changes may conflict with changes made on the device. The client and server need to be able to synchronize their data and the server will handle any conflict resolution.

3. Data Objects

Data objects contain the data used to populate screens in MusicStation. They use methods that allow them to write and read themselves to and from a file or stream. They are used to transfer data between the client and server and to load and store data locally in files on the memory card.

3.1 Data Object Groups

A data object can contain a collection of other data objects, for example an Artist data object contains a collection of Releases. In turn, a Release contains a collection of Tracks.



Data object can also store lists of objects, for example ArtistGroup stores a list of Artists. The 'My Artists' screen uses an ArtistGroup data object to display all artists owned by the user. Because Artist contains Releases and a Release contains Tracks, Artist and Release are also data object groups.

3.2 Data Object Views

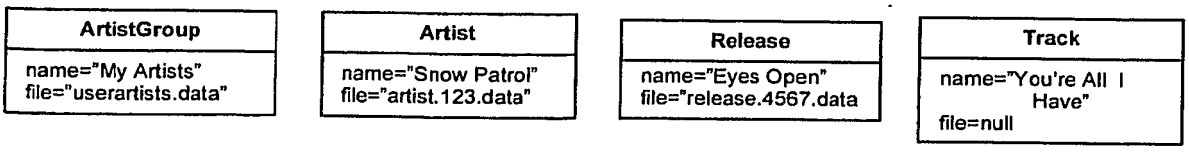
A data object view provides a sorted and filtered view of a data object group. All screens in MusicStation that are populated by data are backed by one or more views. Any changes to a data object group are propagated to the view, which is responsible for updating the screen to reflect these changes.

This allows us to display a screen immediately before a data object is loaded. As the data object is loaded in the background these changes result in updates to the screen, for example the list of Artists on the 'My Artists' screen grows as each Artist is loaded.

4. Data Object Files

Each data object group is stored locally in a file. For example, the 'My Artists' ArtistGroup is stored in its own file. If the user owns 100 artists, each with an average of 2 albums* containing 10 tracks this data object soon becomes very large. When this ArtistGroup object is written it will create a large file and when it is read back from the file it will take a while to populate.

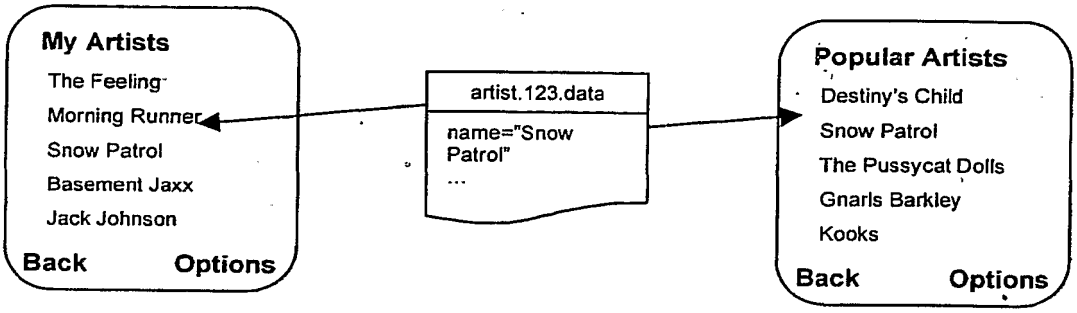
An alternative method is to store each collection of objects in its own file. So in our 'My Artists' example the list of Artists is stored in a file (userartists.data) but the list of albums for each artist is not. The list of albums is stored in a separate artist file, one for each artist (e.g. artist.123.data). Each album is then stored in its own file (release.4567.data) that contains the tracks.



Because each data object is stored in its own file, object groups can use the same data object without having to duplicate the data. For example 'Snow Patrol' are in the 'My Artists' group and the 'Popular

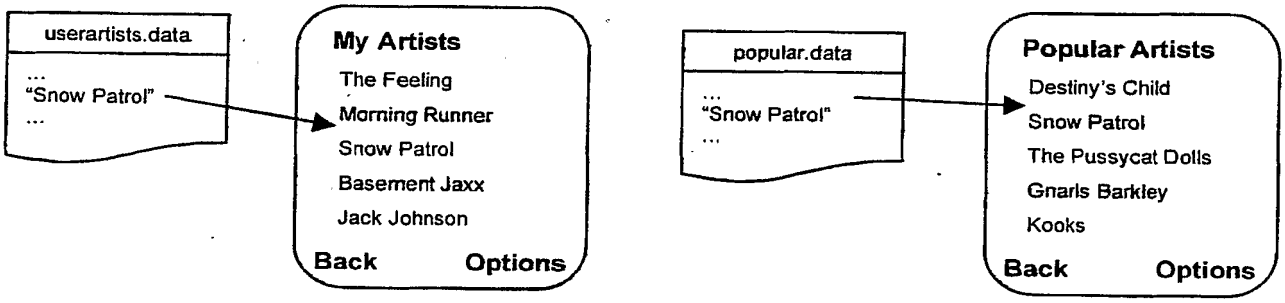
* The user doesn't have to own both albums but both albums exist in the artist data object

Artists' group. If the user buys 'Chasing Cars' from the 'Eyes Open' album we only have to update the 'Eyes Open' album data file. When the user navigates to 'Popular Artists' then 'Snow Patrol' the screen will show that the user has bought 'Chasing Cars'.



However this approach presents its own set of issues. Because the 'My Artists' data file only contains a list of artist ids, we need to open each artist file and read the name of each artist to populate the 'My Artists' screen. There are several major problems with this approach. Firstly, we need every artist file stored locally so any missing files need to be downloaded from the server. Without these files we are unable to display the artist names. Secondly, it's relatively expensive to open a new file connection for each artist in the list so this approach will be slow.

To avoid this we can store the artist name as well as the id in the 'My Artists' data file. This means we can build the 'My Artists' list quickly. However we are then introducing redundancy because the name is now stored in both the artist group data file and the artist data file.



We also may want to sort or filter the list on another property. For example 'Search Results' show user owned artists at the top of the list. To do this we need the owner property as well as the name to display the list. This is more redundant data that we are adding to the group data file.

Because an object can be stored in many groups we need to be aware of this redundancy and make sure that either the client or the server takes responsibility for the updates. In general the server will be responsible for these updates and they will be transferred to the client in response to a client request. Whenever it is possible for these changes to occur when the client is offline, the client will take responsibility for propagating these changes. In these cases, the client will update the local files whether online or offline. For example when a customer modifies a playlist image, any playlist groups that contain that playlist must be updated.

5. Data Object Transfer

Data objects are transferred between the client and the server using the Connected MusicStation Protocol. It is expected that most communication with the client will be over HTTP therefore the client will be responsible for making the initial request.

5.1 Client Request

The client won't always know where objects are duplicated on the memory card. For example the 'Popular Artists' group was pushed to the client however the client has never opened 'Popular Artists' and is unaware that when the user buys 'Chasing Cars' by 'Snow Patrol' that the 'Popular Artists' data file needs to be updated to reflect this. However the server does have this knowledge because it built the 'Popular Artists' data file and sent this file to the client.

For this reason the server is responsible for updating files on the client when records on the server are modified. When the customer purchases 'Chasing Cars' the server will calculate which data files on the client contain 'Chasing Cars' and therefore need to be updated. The server will then either push these updated objects with the purchase response or send commands to the client to update these files when it can. It is preferable that the response contains all data objects that have been modified as a result of the request.

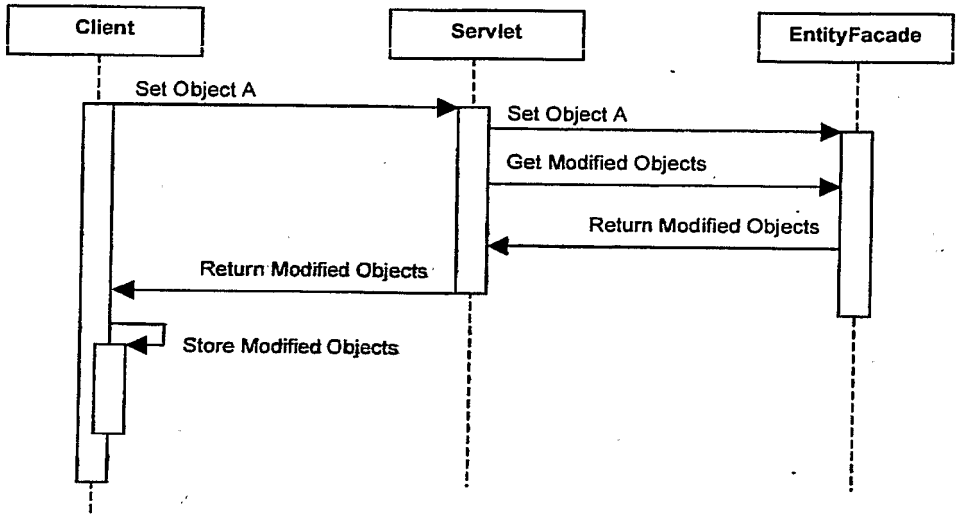
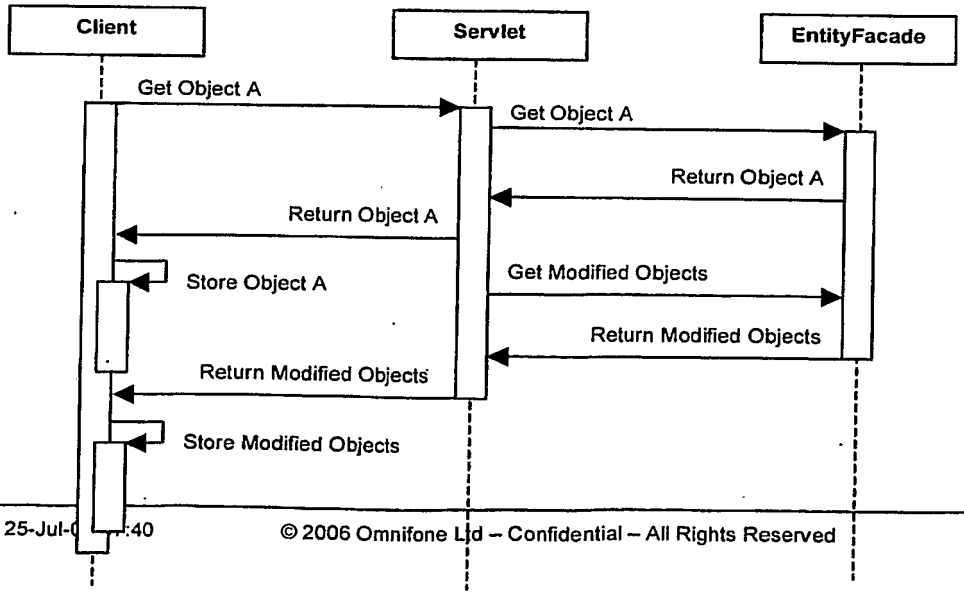


Figure 1. Client sets object and gets all modified objects

5.2 Data Object Push

When records are updated on the server whilst the client is offline and these changes need to be propagated to the client the server will push these to the client on the next request. For example, if the customer purchases 'Chasing Cars' from MusicMate, when the client next connects to the server any objects that need to be updated will be pushed to the client.



5.3 Offline Mode

When the client is offline, the customer is prevented from performing most actions that can modify data. For example, they are not able to buy a track.
However, they should be able to create, edit and share playlists. The client needs to maintain a list of files that have been edited on the client but have not been sent to the server. When the client is next connected it must send these files to the server. All changes made by the client are sent to the server when the client next connects. The server will then return any modified files to the client.

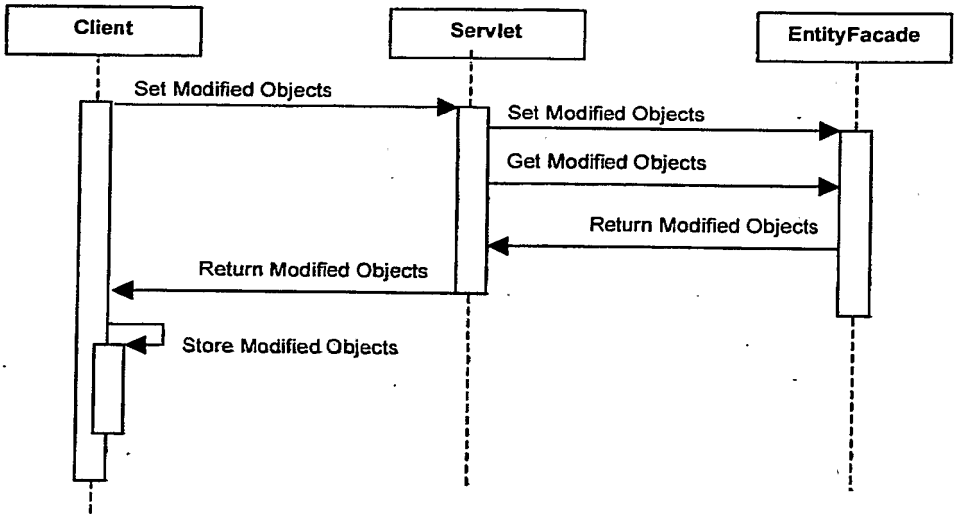


Figure 3. Client sends objects modified in offline mode

6. Change Log

The server maintains a list of objects that have been sent to the client. When one or more of these objects are modified on the server, the modified objects must be delivered to the client as soon as possible. Similarly the client maintains a list of objects that have been created or modified on the client and need to be updated on the server.

The object_change_log table is where changes that don't need to be sent immediately are stored. This is for general system wide changes such as adding a new artist. This table also handles merging changes made from several databases. These changes are communicated when a user session is next created.

Customer specific changes occur in customer_object_change_log. These changes are communicated to the client immediately.

6.1 Server Objects

The changes to objects on the server are stored in the object_change_log table. Whenever a record is inserted, updated or deleted that may affect one or more object data files one or more records are . inserted into this table. This table also allows for changes to made in a separate database, for

example on a staging server, and then when the changes are imported the object_change_log is also imported.

object_change_log
object_guid
object_type
object_modified_date
object_modified_type (addition, replace, deletion)
priority

The list of data objects that exist on the client are stored on the server in the customer_object table. Whenever a session is created for the client we query the customer_object and the object_change_log tables to determine which objects have changed for this customer. It is possible that this query could return several change records for a single object. In this case we only need to consider the last change record. The objects that have changed need to be returned to the client.

customer_object
customer_id
object_guid
object_type
object_created_date
object_modified_date
object_last_used_date
object_deleted_date

Objects that need to be returned to the client are inserted into the customer_object_change_log table. Records may also be inserted into this table when changes occur for objects that affect only one customer. For example when the customer purchases a track and we need to update object data files that reference that track.

customer_object_change_log
customer_id
object_guid
object_type
object_modified_date
object_modified_type (addition, replace, deletion)
priority
acknowledgement_id
acknowledgement_date

Whenever we receive a request from a client we'd like to return all modified objects in the response. In some situations (where bandwidth is limited or the objects are large) we may send a command to the client to request the modified objects later. In situations where many objects need to be returned to the client, the priority field is used to determine which objects are sent first.

To get the list of modified objects for a client we select from the customer_object_change_log table where acknowledgement_date is null.

At first glance it appears that the object_modified_date would be duplicated for each object_guid and could be separated into another table. However for performance reasons object data files on the client contain data from more than one table and an object may need updating on one client and not on another. For example, artist lists contain ownership information for each artist so that they can be sorted with user owned artists at the top. When a customer buys a track by an artist only that customer's artist list has been modified and needs to be updated.

One or more of the following methods could be used to update the object_change_log and customer_object_change_log tables:

- *Database trigger* on a table could populate the object_change_log table when data was added, updated or deleted.
- *Batch process* populates the object_change_log table for example on an import of new content data.
- *Entity Listeners or callback methods* are used on EJB persist, update and remove events.

In most cases callback methods are the most appropriate however for large inserts, for example a data load, it may be more efficient to use another method.

6.2 Client Objects

The client must also keep a log of changes that need to be sent to the server. The client maintains this list in RMS. Each change is stored in a ChangeLogRecord object.

ChangeLogRecord
objectGUID
objectModifiedDate
objectModifiedType (ADDITION, REPLACE, DELETION)
acknowledgementId
acknowledgementDate

The *objectGUID* will be the same GUID used to identify the object on the server unless the client has added this object. In which case the client will assign a temporary GUID that will be used until the server updates the object with its new server generated GUID.

Whenever the client connects to the server it will send all of the objects in the change log. The server should respond with an acknowledgement for each object. When the client receives an acknowledgement it will then delete the corresponding ChangeLogRecord.

7. Conflict Resolution

When conflicts occur because the same object has been modified on the client and server the server is responsible for resolving the conflict. The server communicates the resolution to the client by sending it the updated object.

We'll attempt to minimize the number of situations where conflicts can occur by making the server responsible for most updates. Only in a few cases will the client be able to modify objects and send the changes to the server.

In the prototype the client modifications are limited to:

1. Create playlist
2. Edit playlist
3. Delete playlist
4. Edit customer profile (catchprase, icon)
5. Rate track

When designing the conflict resolution strategy we need to bear in mind the following types of conflict:

- *Update conflicts* occur when the update to a record conflicts with another update.
- *Uniqueness conflicts* occur when the update to record violates a uniqueness constraint with a conflicting record.
- *Delete conflicts* occur when a record is updated that has also been deleted.

Are there any general strategies we can use to resolve conflicts? For example accept the changes from the last modified record only.

8. Use Cases

Whenever an object is updated or deleted the object_change_log or customer_object_change_log tables must be updated to reflect this change. Because object data files on the client contain redundant data it's likely that a change will affect more than one object.

8.1 Server Changes

8.1.1 Artist releases a new album

The artist 'Snow Patrol' releases the album 'Eyes Open'. Every client that contains the artist data file for 'Snow Patrol' needs to be updated.

First we insert change records for 'Snow Patrol' and 'Eyes Open' into the object_change_log table.

object_guid	object_modified_date	object_modified_type	priority
SnowPatrolXYZ	18/07/2006 13:16:33	REPLACE	3
EyesOpenXYZ	18/07/2006 13:16:33	ADDITION	3

When a customer who has the 'Snow Patrol' artist file connects to the server and a session is created the customer_data_object table is joined with the object_change_log table to find any objects that have been modified for this customer.

```
SELECT FROM customer_object, object_change_log
WHERE customer_object.object_guid = object_change_log.object_guid AND
    customer_object.deleted_date IS NOT NULL AND
    customer_object.object_modified_date <
    object_change_log.object_modified_date;
```

This query returns the 'Snow Patrol' object_change_log record. This record is inserted into the customer_object_change_log table.

customer_id	object_guid	object_modified_date	object_modified_type	priority
567	SnowPatrolXYZ	18/07/2006 13:16:33	REPLACE	3

The customer_object.modified_date field is also updated to '18/07/2006 13:16:33'.

The 'Snow Patrol' data file is then sent to the client and the customer_object_change_log.acknowledgement_id field is set. When the client acknowledges the file then the customer_object_change_log .acknowledgement_date field is set.

8.1.2 Artist is removed

The artist 'Cliff Richard' is removed from MusicStation. Every client that has stored the 'Cliff Richard' data file or has a list that contains 'Cliff Richard' needs to be updated.

The object_change_log table is updated and a deleted record is inserted for the following objects:

Artist
Artist.getAlbums()
Artist.getLists()
Artist.getAlbums().getLists()
Artist.getPlaylists()

8.1.3 Import from new content provider

A new content provider supplies 20,000 new Artists, 50,0000 new releases and 100,000 new Tracks.

For each newly added object, a record is inserted into object_change_log.

8.2 Client Changes

8.2.1 Customer buys a new track

The customer buys 'You're All I Have' by 'Snow Patrol'. It's the first track he's bought by 'Snow Patrol'. When the customer requests to buy 'You're All I Have' the 'My Artists', 'Popular Artists', 'Snow Patrol' and 'Eyes Open' data files all need to be updated to reflect the fact that the customer now owns 'You're All I Have'. This is because 'Popular Artist' contains the Artist object that contain whether the user owns that Artist. At the moment the Popular Artists screen isn't divided into artists you own and those you don't but this may be a requirement in the future.

When the new CustomerPurchase object is created the following objects need to be updated in the object_change_log table for this client:

Track.getRelease()
Track.getRelease().getPrimaryArtist()
Track.getRelease().getLists()
Track.getRelease().getPrimaryArtist().getLists()
Track.getPlaylists()

8.2.2 Customer shares a playlist

The customer decides to create and share a new playlist 'Sunday Stroll'. The client sends the new playlist to the server. Any changes are sent on the next request to the server. As you are browsing tracks to add to the playlist it is likely you are communicating with the server, and each time the changes will be sent.
When the Playlist object is created the object_change_log table is updated and a record is inserted into customer_object_change_log for every client that has the customer's data object file.

8.2.3 Customer changes language

The customer selects a different language. We'd like all files that contain language specific data to be updated.

Both the messages property file and editor captions need to be updated to reflect this change. Only playlists display editor captions on the client and so for any playlist on the client that has an editor caption a record is inserted into customer_object_change_log.

8.2.4 Customer adds track to playlist whilst server deletes track

A user adds a track T to a playlist whilst offline. Meanwhile the server deletes track T.

When T is deleted a record is inserted into object_change_log. When the client sends the updated playlist we'll compare the changes with the records in object_change_log and delete the track from the playlist and send it back. The customer won't be informed of this, the track will just disappear.

8.2.5 Customer adds track to playlist whilst server renames track

A user adds a track T to a playlist whilst offline. Meanwhile the server renames track T.

When T is renamed a record is inserted into object_change_log. When the client sends the updated playlist we'll compare the changes with the records in object_change_log and rename the track on the playlist and send it back.

9. Device Memory Management

The device is able to communicate to the server how much memory there is left for storage. The server will use this information to decide if any files should be deleted from the client when delivering updates.

The object_last_used field in the customer_object table stores the date the client last used a particular object. This field is populated from log data sent from the client to the server. The server uses this data to determine which files should be deleted. The server may also use other methods to predict which files should be deleted, for example a story no longer exists in any list.

The client also maintains a list of last used files and is able to delete these itself before it runs out of memory. This list is stored in RMS and references files by relative path and filename. The path and filenames will be short because we intend to remove any meaning from the names. This acts as a safety valve in case there's a problem with the deletion logic on the server.

10. Further Considerations

The object_change_log and customer_object_change_log tables will grow very large over time. Do we need to consider a process to purge or archive data in these tables?

Intelligent Parallel Downloading

Mark Sullivan – 25 Jul 2006

Table of Contents

1. Scope

This document describes how the MusicStation client schedules requests so that data for the current screen is loaded immediately whilst data we think the user might need is loaded whenever there is capacity.

2. Introduction

Typically the UI thread will respond immediately to the user navigating to a new screen by displaying that screen and scheduling a task to load the data behind the screen, either from the local file system or remotely over an HTTP connection.

The load task is added to the task queue. The queue is ordered by task priority, task type and scheduled execution time. Most tasks are scheduled for immediate execution, in which case the execution time is set to the time the task was added to the queue. Some tasks are scheduled with a small delay; for example playing a track is scheduled with a one second delay to allow quickly skipping through the tracks on a playlist.

When a new task is added to the task queue we compare its priority to the currently executing task (if there is one). If its priority is higher then the current task, we attempt to cancel the current task. Only tasks that take an excessive time to complete can be cancelled. This is to avoid tasks hogging the execution thread whist other higher priority tasks are waiting. An excessive amount of time is longer than a few seconds. The cancelled task is then rescheduled. The execution time is set to the time the task was originally added to the queue.

3. Scheduler

The Scheduler is a facility for threads to schedule tasks for immediate or future execution in a background thread. Tasks may be scheduled for one-time execution, or for repeated execution at regular intervals.

The Scheduler object has a single background thread that is used to execute all of the scheduler's tasks sequentially. If a scheduler task takes excessive time to complete, it "hogs" the timer's task execution thread. This can, in turn, delay the execution of subsequent tasks, which may "bunch up". Any task that may take longer than a few seconds to execute must implement interrupt().

The interrupt() method is called when a task with a higher priority is added to the task queue and will be called on the currently executing task by the thread adding the task. The run() method called by the scheduler thread must throw an InterruptedException at the earliest opportunity. The Scheduler will catch this exception and then reschedule the interrupted task for execution based on its priority and the time it was originally added to the queue. The newly added task is then picked up and executed.

This class is thread-safe: multiple threads can share a single Scheduler object without the need for external synchronization.

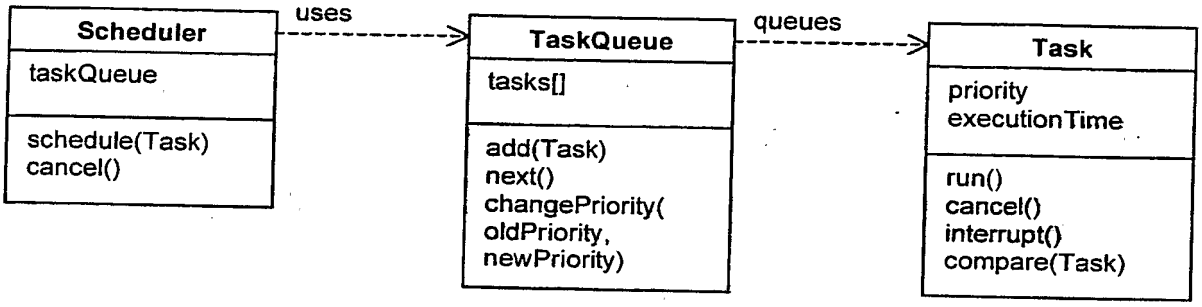


Figure 1. Scheduler class diagram

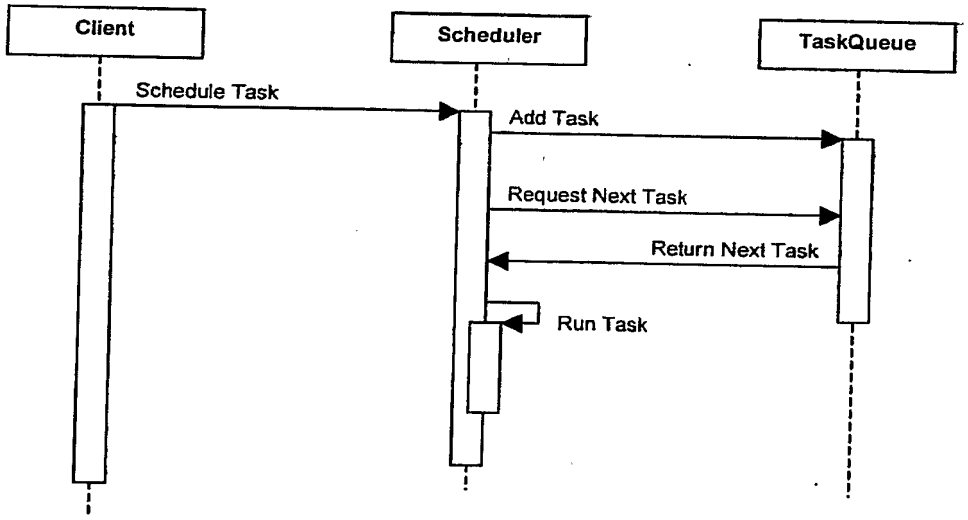


Figure 2. Client schedules a task

3.1 TaskQueue

This class represents a scheduler's task queue: a priority queue of Tasks, ordered on priority, taskType and executionTime.

The task priorities are based on the CLDC Thread priorities. As such there are 3 priorities defined:

- *MAX_PRIORITY* is the maximum priority that a task can have.
- *NORM_PRIORITY* is the default priority that is assigned to a task.
- *MIN_PRIORITY* is the minimum priority that a task can have.

Tasks with the same priority are further subdivided by taskType. For example, this allows us schedule the data for a screen before the images. This could be achieved by using different priorities however it is likely we'll want to lower the priority of a task (e.g. the user navigates to a different screen) without changing the type. By separating the concepts of priority and taskType the design is more flexible and I think easier to understand. Initially the 3 types ordered by importance are:

- *DATA* is used for tasks that request object data files.
- *AUDIO* is used for tasks that request audio files.
- *IMAGE* is used for tasks that request image files.

The executionTime ensures that tasks with the same priority and taskType are executed in the order that they are added to the queue.

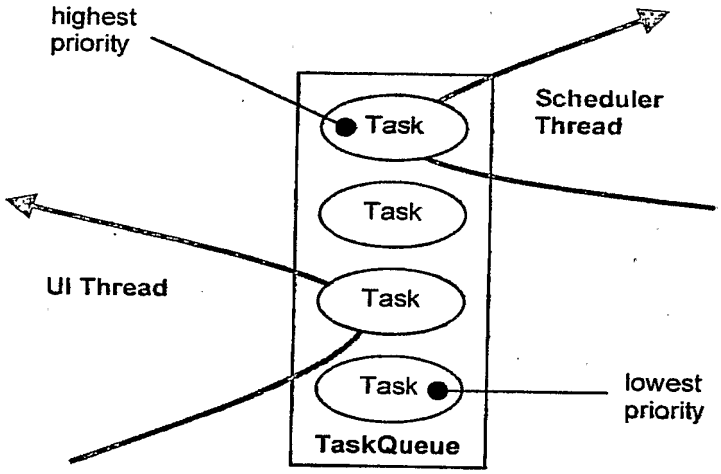


Figure 3. UI thread adds a task to the queue

Internally the queue is stored as a binary heap so the cost to schedule a task is log n, where n is the number of concurrently scheduled tasks. A large number (thousands) of scheduled tasks should present no problem. There is no cost for retrieving the next scheduled task, which is always at the root.

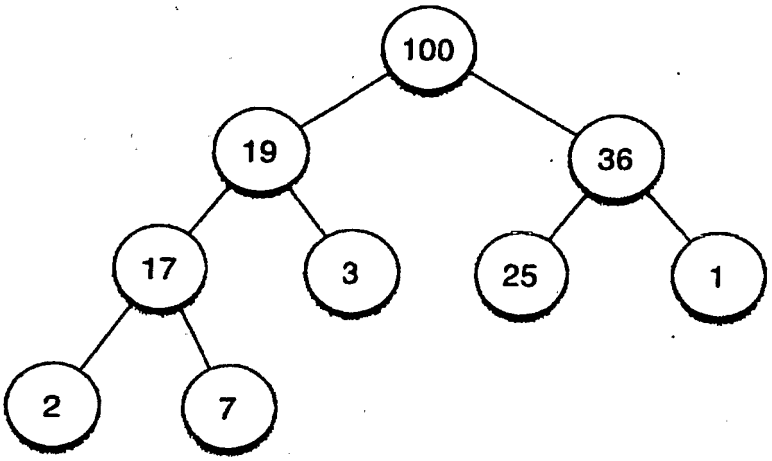


Figure 4. An example of a binary heap

We always add an element to the bottom of the heap and then call the fixUp() method to find its place in the heap. The fixUp() method compares the added element with its parent and swaps them if they are not in the correct order.

An array is used to store the heap and because the heap is always complete (there are never any gaps in the tree) it can be stored compactly. No space is required for pointers; instead, for each index i, element a[i] is the parent of two children a[2i+1] and a[2i+2].

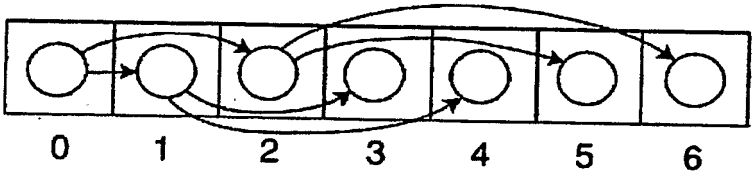


Figure 5. Binary heap stored in an array

3.2 Task

A task can be scheduled for one-time or repeated execution. A task can be in one of 3 states:

- *SCHEDULED*: This task is scheduled for execution. If it is a non-repeating task, it has not yet been executed.
- *EXECUTED*: This non-repeating task has already executed (or is currently executing) and has not been cancelled.
- *CANELLED*: This task has been cancelled (with a call to Task.cancel).

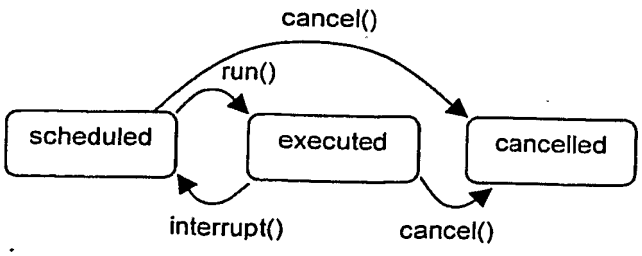


Figure 6. State diagram for a Task

The MusicStation client uses a single Scheduler to schedule all file connections, either from the local file system or remotely over an HTTP connection. The scheduler uses a single thread so all file connections are handled serially. A Task must ensure that it only has an open connection when it is in the executed state. As only one task is ever in the executed state we can guarantee that we only ever have one connection open. Also, any task that supports the interrupt() method must be able to resume without storing any state information about the file it was writing to. This is important because another task may have modified the file since the task was interrupted.

4. Use Cases

4.1 . User opens Playlists

The user opens the application and immediately opens the Playlists menu. The Playlists menu display the "My Playlists" PlaylistSet filtered using the two filters "My Private Playlists" and "My Public Playlists".

When the screen is displayed a LoadTask is added to the TaskQueue to load "My Playlists". The LoadTask.taskType is DATA and the LoadTask.priority is MAX_PRIORITY.

When the LoadTask is added to the TaskQueue, the scheduler thread, which is waiting on the queue, is notified. It takes the task from the queue and executes it by calling the Task.run() method. The task checks to see if the "My Playlists" object data file exists on the file system. In this case it doesn't so an HttpConnection is opened and the file is read over the stream. The file is read into a buffer (65k) and each time the buffer is filled it is written to the memory card and used to populate part or all of the data object (note very few data files will be larger than the buffer).

As the PlaylistSet data object is populated with Playlists, these Playlists contain image references. As each image reference is read, an ImageLoadTask is created and added to the TaskQueue. The ImageLoadTask.taskType is IMAGE and the ImageLoadTask.priority is MAX_PRIORITY.

Once "My Playlists" has finished loading, the scheduler takes the first ImageLoadTask from the queue. Because the image doesn't exist on the local file system it is loaded over HTTP. This continues until all images have been loaded.

4.2 User opens Playlists and immediately selects New Playlists

The user opens the application and then opens the Playlists menu. Before "My Playlists" have loaded the user selects "Get New Playlists".

As above a LoadTask is immediately added to the TaskQueue to load "My Playlists" when the user opens Playlists. The LoadTask.taskType is DATA and the LoadTask.priority is MAX_PRIORITY.

Before the LoadTask has finished the user selects "Get New Playlists". This immediately calls TaskQueue.changePriority() to downgrade all MAX_PRIORITY tasks to NORM_PRIORITY because we are changing screens. Any outstanding tasks for the last screen need to have a lower priority than tasks for the new screen.

A LoadTask is then added to the TaskQueue to load "New Playlists". The LoadTask.taskType is DATA and the LoadTask.priority is MAX_PRIORITY. Adding the new task causes interrupt() to be called on the "My Playlists" LoadTask. As data objects are typically small (less than 4k) interrupts are ignored. However because the "My Playlists" LoadTask has had its priority lowered to NORM_PRIORITY any ImageLoadTasks it creates are also created with NORM_PRIORITY.

Once the "My Playlists" LoadTask has finished loading, the scheduler takes the "New Playlists" LoadTask from the queue and executes it. Once "New Playlists" has loaded the images for the "My Playlists" screen load in the background.

4.3 User starts Playlist

The user selects a Playlist from "My Playlists" and chooses the Play option.

All of the Tracks in the Playlist are added to the play queue. A StartTask is added to the TaskQueue for the first track. The StartTask.taskType is AUDIO and the StartTask.priority is MAX_PRIORITY. We then add a FetchTask to the TaskQueue for each Track. The FetchTask.taskType is AUDIO and the FetchTask.priority is MIN_PRIORITY. Note, a FetchTask is added for each Track including the first track. This is because the StartTask could be cancelled by the user selecting Next before the task has finished. A FetchTask will first check to see if a file exists and has been fully downloaded before making an HttpConnection.

When the StartTask has finished (and the track begins to play) a PrefetchTask is added for the second track. The PrefetchTask.taskType is AUDIO and the PrefetchTask.priority is MAX_PRIORITY. Depending on the connection rate the second track should have prefetched before the first track finished. In which case the first and second FetchTasks are discarded (the files already exist) and the third tracks FetchTask begins to load.

4.4 User starts Playlist and opens Inbox

The user selects a Playlist from "My Playlists" and chooses the Play option. Halfway through downloading the first track the user opens the Inbox tab.

As above, a StartTask is added for the first track and FetchTasks added for each track. When the user opens the Inbox a LoadTask is created for the ""Inbox" StorySet. The LoadTask.taskType is DATA and the LoadTask.priority is MAX_PRIORITY.

The priority of the StartTask is changed from MAX_PRORITY to NORM_PRIORITY and the "Inbox" LoadTask is added to the TaskQueue. The interrupt() method is called on the StartTask which causes the StartTask.run() method to throw an InterruptedException the next time read() returns (when the 65k buffer is filled). The scheduler catches the InterruptedException and the StartTask is rescheduled to run after the "Inbox" LoadTask.

The "Inbox" LoadTask is executed and it creates ImageLoadTasks for each story. These are created with MAX_PRIORITY and will all be executed before the StartTask is resumed. Should we leave the StartTask at MAX_PRIORITY so it is loaded before the images?

Once the images are loaded the StartTask resumes by first checking if the file exists and how much has already been read. The task will then request the remainder of the audio file. Once the file has been downloaded the track will play and a PrefetchTask will be added for the next track.

Client Logging

Mark Sullivan – 24 Oct 2006

Table of Contents

1. Introduction

We need to log user actions, events and exceptions on the client and send them to the server in order to:

- Debug information during testing
- Provide information for customer support
- Collect usage data for reports and recommendations

2. Logger

The Logger object is used to control logging on the client. It is a DataObjectSet and can be synchronized with the server using MSTP.

Logger contains the following attributes:

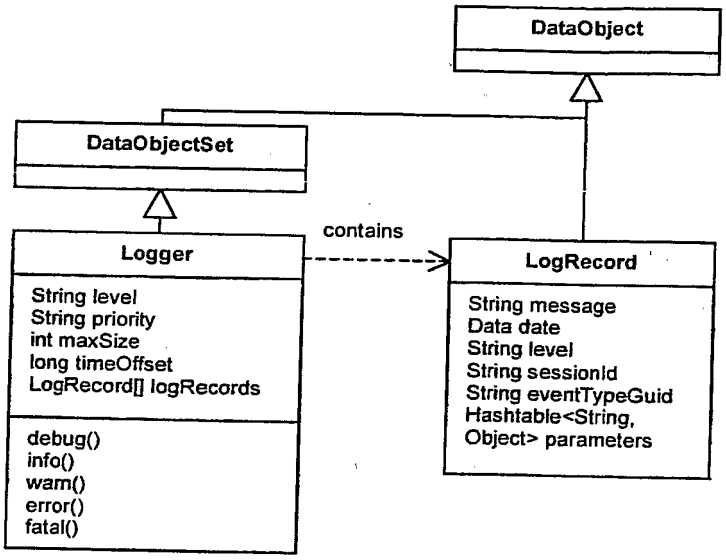
- Level: The level at which logs are stored, events at a lower level are discarded
  - DEBUG: Events that are useful to debug the application
  - INFO: Informational messages that highlight the progress of the application
  - WARN: Indicates that there's a potential problem
  - ERROR: An error occurred but the application managed to continue
  - OFF: Nothing is logged
- Priority: Controls the frequency that logs are sent to the server
  - MIN: When the client next makes a request to the server or when maxSize is reached.
  - NORMAL: Every 5 minutes (or as MIN)
  - MAX: Every 30 seconds (or as MIN)

  This behaviour will be controlled by properties and can be tuned
- MaxSize: The maximum number of records to store on the client
- TimeOffset: The time difference between the server and client
- LogRecords: The logs themselves

The Logger contains a LogRecord for each client log. The LogRecord conatins the following attributes:

- Message: Readable description of what happened

- Level: The level of this log
- Date: The server time calculated using the client time and timeOffset
- SessionId: The server sessionId when this event occurred (if any)
- EventTypeGuid: The identifier in the event_type table for this event (if any)
- Parameters: The parameters that are pertintent for this event



3. Client Debugging

Whilst the client is in testing we need to allow the testers to easily view the client logs so that they can understand what was going on when the error occurred and can include these in Mantis bug reports.

The client will log the following:

- Tasks including all parameters required to run the task
- Commands including all parameters required to run the command
- Exceptions including all relevant information

Each LogRecord will be logged as an incident and can be viewed by the tester using the Incident Monitor's web interface. Because each record is logged using the server time the incidents can be ordered by date to give a list of client and server actions in the sequence that they occurred.

4. Customer Support

When the customer contacts customer support we need to push the Logger object from the client to the server so that customer support can see the last LogRecords generated by the client. The client needs to initiate the push, which it may not do for a while if the log priority is set to MIN. Therefore we need a method for commanding the client to post the Logger object.

Can support ask the user to go to the About screen and this will push the Logger object to the server immediately?

We need to log enough information to recreate the user's situation. This information will be stored in the LogRecord.parameters Hashtable. If the eventTypeGuid attribute is set a record will be inserted

into the customer_event table and the parameters inserted into customer_event_val. We will use a queue to insert into customer_event and customer_event_val so that event logging doesn't delay the response to the client. The exception is when the customer_logger.priority is set to MAX. In this case we want to see events as they happen and these records will be inserted directly into the database.

5. Usage Data

Client usage data is populated using triggers on the event table. So for example when we receive an event for the customer plays a track the customer_track.play_count is incremented.

6. Database Requirements

Customer support needs to be able to control the logging generated by the client and the frequency that it's sent to the server. This is controlled using the customer_logger table.

customer_logger	type	constraint	default	comments
customer_id	NUMBER(10)	Mandatory		Foreign key in customer
level	VARCHAR2(256)	Mandatory Allow values: DEBUG, INFO, WARN, ERROR, OFF	INFO	Level of logging output on client
priority	VARCHAR2(256)	Mandatory Allow values: MIN, NORMAL, MAX	NORMAL	Task priority for sending log records to the server
max_size	VARCHAR2(32)	Mandatory	100	Maximum number of log records stored on client

Whenever this table is changed a record is inserted into customer_object_log so that the updated Logger object can be pushed to the client.

7. Further Considerations

How do we ensure logging is reset after a support call has terminated?

Connected MusicStation Protocol Design

Mike/Steve/Tom/MarkS/MarkK
9 Jun 2006 – 19 June 2006

1. Introduction

MusicStation client applications are required to connect to the MusicStation server to download and upload various data. The protocol that MusicStation uses to connect to the server must be capable of being implemented on a variety of client technologies, e.g. Java, Symbian, and Windows Mobile. It must also solve the issues documented in the document "Connected MusicStation Issues and Requirements"[1]

2. Protocol History

MyFone used HTTP to transfer data. This experience showed up several issues with the HTTP request and response having to pass through operator gateways. Operator gateways and various mobile phones regularly interfered with the HTTP headers, usually by failing to forward them. This is one of the key factors that led to the creation of this protocol.

To transfer several files in one response, this protocol took inspiration from MIME. An earlier revision of this document used MIME like boundaries to separate the different files in the response. This was changed to use the offset and length notation in the header. This allows a client to quickly access the data objects. Only the header needs to be parsed, and not the body contents. (see section 3.2.7)

Previously status codes used a binary representation to allow them to be extensible whilst still being understood by older clients. This has been simplified to use integer values that can be easily understood by humans as well as by the client. The server sending the most appropriate status code to the client addresses the issue of introducing new status codes. The server will only ever send status codes that the connected client version understands.

It was previously not possible to uniquely identify acknowledgments if the same file had been requested several times in a session, or if the acknowledgement was sent in a different session to the session that the data file was sent in. The acknowledgement id principle used in `Sent` and `Put` lines solves this issue.

3. Protocol Overview

The client must initiate all communication because of the way in which mobile phones connect to the internet. As the mobile phone does not have a static IP address, and because it will usually connect via a mobile operator gateway there is no way for the server to initiate the communication. MIDP2.0 handsets could use the Push Registry functionality to send an SMS to the application requesting that the client makes a request to the server, but this functionality may not be available on all target handsets and client platforms, so therefore the MusicStation protocol should be based upon the client initiating the communication.

The protocol must be able to run over HTTP and TCP/IP socket connections. These are the two most commonly available connections made available to us by the client platforms.

The protocol will assume a reliable transport layer. The protocol will not need to be able to re-request individual packets of a particular response. Therefore UDP socket connections will not be a supported

[1] G:\Projects\MusicStation\Architecture & Design\ Connected MusicStation - Issues and Requirements.doc

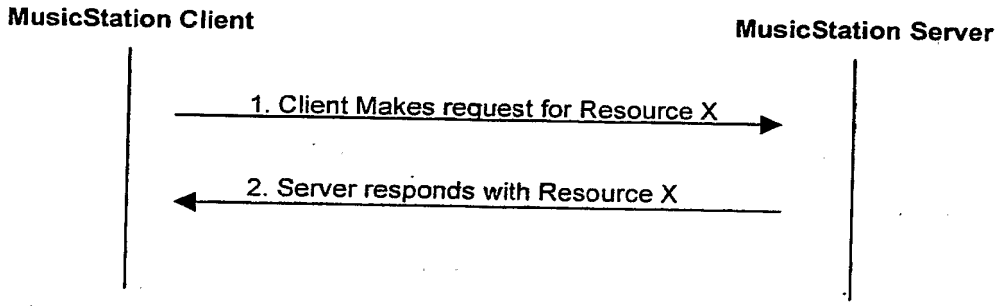
transport mechanism. To support an unreliable transport layer would require a lot of extra functionality in the MusicStation protocol and TCP is available on all clients that have UDP.

The protocol must be able to support the client transferring data to the server as well as the client making requests for data from the server. This is required so that error data, logging data, usage data, playlist information and user related data can be transferred from the clients to the server.

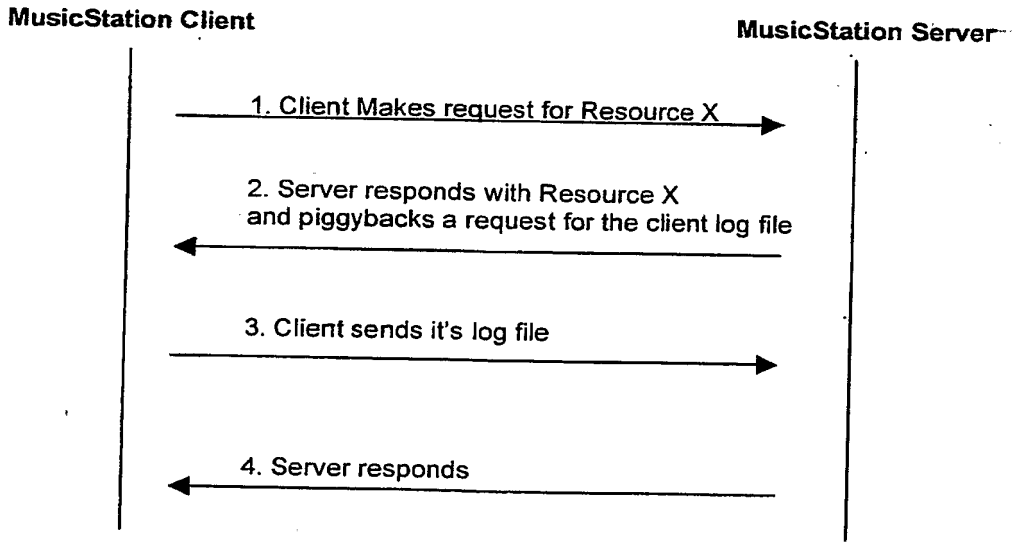
As the MusicStation is a request / response protocol it is modelled closely on HTTP, borrowing several of HTTPs features.

The MusicStation protocol is text based using the ASCII character set only, this is so that it can be implemented on many different client platforms without any of the encoding issues associated with binary data.

The diagram below shows the request/response flow between the client and the server. This is an example of a simple request from the client being fulfilled by the server. All client/server communication happens in this same basic way.



The next diagram shows how the server sends a request to the client. As the client/server communication must always be triggered by a client request, the only way for the server to make a request from the client is for the server to piggyback the request on a response it sends to the client.



Note that in normal operation the server will always respond to a client request, even if there is no data in the response. The response may include just a status code (see 'The Server "Response" protocol' section).

Robot clients and requests exceeding a requests per minute threshold are not normal operation, and the server has no obligation to respond to these requests. Real clients that do not receive a server response are expected to retry the request after a reasonable time.
Like HTTP the MusicStation Protocol uses a header to hold the meta-data about the body of the message, which contains the actual data being transferred. This document describes the protocol, which is concerned with just these headers. The body of the message can differ for the various client implementations. Like HTTP the header and body are separated by an empty line.

3.1 The Client "Request" protocol

3.1.1 The protocol identifier

The first part of any request will be the protocol identifier. This is so that the server receiving this request can validate that the data it has received is indeed from a client. The protocol identifier should be short so that it does not put an overhead on the request. The protocol identifier used by MusicStation is

```
MSTP
```

This stands for **MusicStation Transfer Protocol.**

3.1.2 The protocol version number

Along with the protocol identifier is the protocol version number. This protocol identifier is entirely separate from the client version number, the server version number and the data objects used by the client version number.

There can be many different versions of the client application that will all use the same protocol version number.

The protocol version number will be in the form `major.minor`

The minor number should be increased for incremental changes to the protocol, and the major number should be increased with significant changes to the protocol. Initial development versions of the protocol will have a major number of 0. This will be incremented to 1 on the first production release of the protocol.

The server software should always be able to handle every released version of the protocol so that it is backwardly compatible with all older client versions.

The protocol version number will be on the same line as the protocol identifier, and separated from the protocol identifier by a forward slash.

```
MSTP/0.1
```

This line indicates that this is version 0.1 of the MusicStation protocol.

3.1.3 The request identifier

Each request sent by a MusicStation client will include an identifier. This identifier must be unique to this request in the current session. There is no requirement for the request identifier to be globally unique. This request identifier can be any string up to 32 characters long.

This could be implemented as an integer starting at 1 and being incremented for every request made by the client.

This request identifier is required so that the server can identify duplicate requests from clients. MyFone experience has shown that mobile phone client requests can sometimes be very unreliable. This means that the client must be able to automatically retry a request if it has not received a response within a reasonable amount of time.

When the client has not received a response, this could be because the request never got as far as the server, or it could be because the server's response got lost in the operator gateway on it's way back to the client.

By including the request identifier it is straightforward for the server to identify duplicate requests.

The client must send the same request identifier for any re-tried requests.

The request identifier can come at any point underneath the request identifier and request version number.

```
MSTP/0.1
RequestId: 123456
```

This identifies a request by this client. If the client retries this request, the request id in the retry must by 123456.

3.1.4 Client name and version number

Every request must include the client name and version number. This information can then be used on the server to perform a look up of the abilities of this client. This means that new abilities can be added at any time to the client without having to change the information given in the protocol.
For example, if a client identifies itself as the MIDP version 0.4.6 client, then the server knows which format it need to return the data objects in. The server also knows what music encoding is supported by this client. And the server knows that this client does not support encrypted music files.

```
MSTP/0.1
RequestId: 123457
Client: MusicStation 0.4.6 MIDP Nokia/N70
```

This identifies the client as the Java client version 0.4.6 running on a Nokia N70 handset. The server can then look up which abilities this client has.

The format of this string should always be

The static string "MusicStation" followed by a single space.
Then the version number which is in the format major.minor.micro, followed by a single space.
Then a variant name which will be defined for each variant. Currently this will be one of "MIDP", "Symbian", "WindowsMobile", "MusicMate".
The variant is followed by a space character, and then a platform identifier. The platform identifier is likely to be our device code from the db. This platform identifier will differ for each variant. For MIDP clients it is suggested that "Manufacturer/Model" is used.
Occasionally there may be different releases for the same MIDP handset. The different release may be required to work around firmware specific issues. In cases like these, the platform identifier should include this information. E.g. "MusicStation 0.4.6 MIDP Nokia/6630-pre3.6.0firmware"
This platform identifier may contain spaces, and will generally be used to perform a look up in the database for the properties of that device.

3.1.5 User's globally unique identifier

Every request must include the user's globally unique identifier. The one exception to this is the initial registration request. If a request does not include the user's globally unique identifier, then the server will respond with notification that the client is required to register.

This globally unique identifier allows the server to lookup various information about the user.

The client should not construct the globally unique identifier. The identifier will be created by the server during the registration process, and then assigned to the client. The client must then include this identifier in every subsequent request.

```
MSTP/0.1
RequestId: 123458
Client: MusicStation 0.4.6 MIDP Nokia/N70
UserGUID: AB12YZ
```

```
This identifies the user with the Globally Unique Identifer AB12YZ. The server can use this
information to look up user details such as preferred language, territory, operator and
branding.
```

3.1.6 Data requests

Most of the requests from the client will be a request for data from the client. For example the client may request the latest news from the server.

```
MSTP/0.1
RequestId: 123459
Client: MusicStation 0.4.6 MIDP Nokia/N70
UserGUID: AB12YZ
Get: inbox.data
```

This is an example of a request for the inbox.data data object file.

Data requests may also have path information associated with them. This uses a syntax similar to HTTP URLs. A / (forward slash) character is used as a directory separator.

```
MSTP/0.1
RequestId: 123459
Client: MusicStation 0.4.6 MIDP Nokia/N70
UserGUID: AB12YZ
Get: games/namethattune/question.data
```

This is an example of a request for the question.data data object file which has the path games/namethattune.

Data requests may optionally include parameters that the server will use to construct the data object to be returned to the client. This request data is included by using HTTP query string syntax.

```
MSTP/0.1
RequestId: 123460
Client: MusicStation 0.4.6 MIDP Nokia/N70
UserGUID: AB12YZ
Get: advncedSearch.data?type=artist&query=bon%20jovi&country=uk&language=en
```

This is an example of a request for the advanced search results. The requested resource has a ? (question mark) character to separate the name of the requested resource from the parameters for this resource. The parameters are name/value pairs. Each name/value pair is delimited by a & (ampersand) character, and the name and value part are separated by an = (equals).

The values have been URL encoded, so that the space character in the search term "bon jovi" has been replaced by the URL encoded version %20.

The client may request multiple resources from the server at the same time. To do this, the client sends several `GET` lines, one for each resource requested.

```
MSTP/0.1
RequestId: 123461
Client: MusicStation 0.4.6 MIDP Nokia/N70
UserGUID: AB12YZ
Get: inbox.data
Get: charts.data
```

This is an example of a request for the inbox.data file and a request for a charts.data file. A situation like this may occur when the client is making a request for a resource it requires immediately (in this example inbox.data), and is also required to update a resource in the background (in this example charts.data).
The `GET` lines should be ordered in the order that the client would like the resources ordered in the server response.

Occasionally the client may have a partial response cached, and require only some of the data returned from the server. In cases like this, the client may want to make a request for only a certain part of the data.
The client can do this by using the range parameters on a `GET` line. The range parameters are separated from the requested resource name by a ; (semi colon) character.
If there is more than one range parameter then the range parameters are separated by a ; (semi colon) character.
The range parameters are `from` and `to`. Both of these should be followed by an = (equals) character, and then an integer number of bytes.

```
MSTP/0.1
RequestId: 123462
Client: MusicStation 0.4.6 MIDP Nokia/N70
UserGUID: AB12YZ
Get: inbox.data; from=34
```

This is an example of a partial request for the inbox.data file. The client is requesting all of the inbox.data file from the 34th byte onwards.

```
MSTP/0.1
RequestId: 123463
Client: MusicStation 0.4.6 MIDP Nokia/N70
UserGUID: AB12YZ
Get: charts.data; from=128; to=256
```

This is an example of a partial request for the inbox.data file. The client is requesting all of the inbox.data file from the 128th byte up until the 256th byte.

When making a range request, the client should not expect the returned data to be of the range asked for. The server response will include the details of the range returned, and the client should use the range information in the server response, and not the range information in it's own request for further processing. This is because the server may have a reason for returning a different range of data. For example if the data has changed since the client last requested it.

3.1.7 Sending data to server

On occasions the client may need to send data to the server. For example, to send error information to the server. The client can do this by using the `put` line.
A `put` line has several parts. Each part is separated by a ; (semi colon).
The first part is the name of the resource being sent to the server.
The next part is an acknowledgment id. This is the id that will be echoed back to the client by the server in the acknowledgement line (see section 3.2.6). The client must generate this acknowledgement id in a way that it can uniquely identify which put data a received acknowledgement is for.
The next part is an offset number of bytes. This offset is how many bytes into the body of the message that this data starts.
The next part is a length number of bytes. The length is how many bytes long the data in the body of the message is.
The final part is the content type. This will almost always be the content type for our data objects. Therefore this may be redundant information, but it has been left in the protocol because images being transferred may not necessarily be wrapped in a data object.

```
MSTP/0.1
RequestId: 123463
Client: MusicStation 0.4.6 MIDP Nokia/N70
UserGUID: AB12YZ
Put: error.data; ackId=1; offset=0; length=160; type="application/octet-
stream"

01001110001001001000100010111111000101010
10010100101111111111000001111000000001000
10000100100000010000111000111111000011111
01010101010101101010101101011011101101101
```

This is an example of the client sending error data to the server.
The block of zeros and ones signifies the body of the message. This is the binary data that is being transferred by this protocol. The format of this data is outside the scope of this protocol because the format will differ depending on the client implementation technology.
The data in the body starts at position 0, and has a length of 160 bytes. The offset and length values in the `Put` line reflect this information.
The content type in the `Put` line tells the server how to interpret this data.

The client may be required to send multiple resources to the server at the same time. In a similar way to using multiple `Get` lines, the client may send multiple `Put` lines.

```
MSTP/0.1
RequestId: 123464
Client: MusicStation 0.4.6 MIDP Nokia/N70
UserGUID: AB12YZ
Put: error.data; ackId=2; offset=0; length=160; type="application/octet-
stream"
Put: photo.jpeg; ackId=3; offset=160; length=320; type="image/jpeg"

01001110001001001000100010111111000101010
10010100101111111111000001111000000001000
10000100100000010000111000111111000011111
01010101010101101010101101011011101101101
01000011100010101011011011011010000000010
10101101101101101000000011000011100010010
00111000101010110110110110110100001110000001
00011100000111000001110000011100000011100
01101101101000011100000011011011010000011
01101101101000011100000011011011010000011
10000001101101101000011100000011011011101
11011010000111000000110110110100011011101
```

This is an example of the client sending error data and a photo to the server. In the body of the request the error data is shown in a different colour to the photo data. The length and offset positions in the request tell the server the offset into this data, and the length of the data.

In a similar way to the `Get` line, the `Put` line also supports parameters on a `Put`.

The syntax for this is the same as the Get line syntax, which is modelled on the HTTP query string syntax.

```
MSTP/0.1
RequestId: 123465
Client: MusicStation 0.4.6 MIDP Nokia/N70
UserGUID: AB12YZ
Put: photo.jpeg?name=Fave%20Tracks; ackId=4; offset=0; length=160;
type="image/jpeg"

01001110001001001000100010111111000101010
10010100101111111111000001111000000001000
10000100100000010000111000111111000011111
01010101010101101010101101011101101101110
```

This is an example of the server sending a Jpeg photo with a single parameter (name = "Fave Tracks").

Generally the parameters should be encoded within a data object. So any use of Put line parameters should be looked at closely to see whether the data should really be part of a data object.

Note, although the `Put` line is very similar to the `Get` line, the Put line does not support the range values `From` and `To`. A failed Put will require a full resend of the data. The client will know whether the `Put` has failed because it will not receive an acknowledgment receipt from the server (see 'The server "Response" protocol' section)

3.1.8 Client acknowledgements

This part of the protocol is really a client response to a server request, so it is worth re-reading after reading the 'Server "Response" protocol' section of this document.

So that the server can always have perfect knowledge of what data exists on each client, the client is required to acknowledge the receipt of every piece of data sent to it by the server. This is done by sending one `Ack` line for each data file successfully received and stored.
The Ack line parameter is the ackId assigned by the server when it sends the file.

```
MSTP/0.1
RequestId: 123466
Client: MusicStation 0.4.6 MIDP Nokia/N70
UserGUID: AB12YZ
Ack: 2006061911030001CHARTS
```

This request shows the client acknowledging that it has successfully received and stored the data file which had an assigned acknowledgment id of `2006061911030001CHARTS`.

A request may include multiple acknowledgement lines.

```
MSTP/0.1
RequestId: 123466
```

```
Client: MusicStation 0.4.6 MIDP Nokia/N70
UserGUID: AB12YZ
Ack: 2006061911030001CHARTS
Ack: 2006061911030001INBOX
```

This request shows the client acknowledging that it has successfully received and stored the data files with acknowledgement ids of `2006061911030001CHARTS` and `2006061911030001INBOX`.

The client must only acknowledge fully received files. It must never acknowledge partially received files. If a client partially receives a file, it should make a `Get` range request for the rest of the data. Once all of the data has been received and stored, the client can then send the acknowledgement for this data.

If the client does not successfully receive and store a data file it has requested, it should send a Not Acknowledged notification to the server.

```
MSTP/0.1
RequestId: 123466
Client: MusicStation 0.4.6 MIDP Nokia/N70
UserGUID: AB12YZ
Nak: 2006061911030001CHARTS
```

This request shows the client telling the server that there was a problem with the receiving or storing of the data file with acknowledgement id `2006061911030001CHARTS`. The server will now know that this file does not exist on the client.

Usually when the client sends a `Nak`, it is very likely to have some accompanying error data that explains the reason for the `Nak`. If the server receives a `Nak`, and no error data, it may want to ask the client to send the log file details. If the client persistently sends `Nak`s to the server, the server may want to increase the logging level on the client to help identify the cause.

3.1.9 Session identifier

Each request the client sends to the server should include a session identifier. The client should not remember this session identifier between restarts. On the first request after starting up, the client should not include a session identifier. The server will respond by sending back a new session identifier. The client should then include this identifier in every subsequent request until the user closes the client.

```
MSTP/0.1
RequestId: 123467
Client: MusicStation 0.4.6 MIDP Nokia/N70
UserGUID: AB12YZ
SessionId: FJSKNBKSKSDKFLSH
Get: inbox.data
```

This request shows the client has previously been assigned a session identifier of `FJSKNBKSKSDKFLSH`.
For more details on how the client gets this session identifier see the 'Server "Response" protocol' section.

3.2 The Server "Response" protocol

3.2.1 The protocol identifier

The protocol identifier used in the server response should be identical to the client request protocol identifier. Clients should check this identifier so that they know the response if in the MusicStation Protocol format.

3.2.2 The protocol version number

The server can support many different versions of the protocol at the same time. The server should always respond with the same protocol version number as the client used in the request. This is because this is the only protocol version number the server can be sure that the client supports. Along with the protocol identifier, the client should check the protocol version number in the response to so that they know the protocol version being used is a version that they understand.

```
MSTP/0.1
```

This is an example of the server sending the MusicStation Transfer Protocol identifier and using protocol version number 0.1.

3.2.3 Response Status Codes

With each response the server will send a status code. The status codes are

Success	1000
Unauthorized	4010
Unauthorized IP Address	4011
Forbidden	4030
Internal Server Error	5000
Service Unavailable	5030
Unsupported Version	5050

The status codes are always 4 digits. This is to allow enough codes to allow for future expansion. 3 digit codes have not been used to avoid confusion with HTTP status codes:

The status codes are extensible, new codes can be added at any time. The server will make sure that clients are only ever sent status codes that the client understands.

The status codes are grouped into 2 sections. Codes starting with the digit 1 (i.e. 1000 – 1999) are to be used for codes relating to a successful operation. Codes starting with the digit 2 (i.e. 4000 – 5999) are to be used for a failed operation.

Within the failure range of codes, there are two further groups. Codes starting with the digits 4 (i.e. 4000 – 4999) are to be used for failure when the client is at fault. Codes starting with the digitis 5 (i.e. 5000 – 5999) are to be used for failure when the server is at fault.

```
MSTP/0.1
StatusCode: 1000
```

This example shows a successful response from the server.

3.2.4 The response identifier

So that the client can verify that the response it receives is in response to the request it made, each response from the server will echo the client's request identifier.

```
MSTP/0.1
StatusCode: 1000
ResponseId: 234567
```

This example shows the server response to the client request with a request id of `234567`.

3.2.5 Setting the session identifier

The first request from the client each time it is started up will not contain a session identifier. The server should respond to this request with a newly assigned session identifier.

```
MSTP/0.1
StatusCode: 1000
ResponseId: 234568
SetSessionId: FJSKNBKSKSDKFLSR
```

This response shows the server setting the session id to `FJSKNBKSKSDKFLSR`.

If the client receives any response with a `SetSessionId` line, then the client must start using the new session id immediately. There may be cases where the server assigns a new session id to a client that already has a session id. For example this could happen when the session has timed-out on the server.

3.2.6 Acknowledging sent data

When the client sends data to the server (for example error data), the server must acknowledge the receipt of this data so that the client knows the server has successfully received this data.

```
MSTP/0.1
StatusCode: 1000
ResponseId: 234569
Ack: 3
```

This response shows the server acknowledging the receipt of the data file which the client sent in a `Put` line and the client assigned a acknowledgment id of 3.

Equally the server can negatively acknowledge the receipt of the data if there has been an issue receiving or storing the data. This will allow the client to resend the data.

```
MSTP/0.1
StatusCode: 1000
ResponseId: 234569
Nak: 4
```

This response shows the server acknowledging the failed receipt of the data file which the client assigned an acknowledgement id of 4.

3.2.7 Sending data

Most responses from the server are likely to include at least one data object file. Theses data files are sent in the body of the response.
For each `Get` line sent in the request by the client, the server should return a `Sent` line.
The server must generate an acknowledgement id that it sends along with the data. This is so that when the server receives an `Ack` line, it knows which data is being acknowledged. It is the servers responsibility to generate these acknowledgement ids in a way that uniquely identifies the data file sent.

The Sent line must include the byte offset position into the body of data where the client can find the data, it must also include the length of the data and the content type of the data. Byte offset and length are used in the MusicStation protocol because they make for relatively straightforward processing. This has been used in preference to a boundary parameter as used in multipart MIME.

```
MSTP/0.1
StatusCode: 1000
ResponseId: 234570
Sent: news1.data; ackId=20060619111100NEWS1; offset=0; length=160;
type="application/octet-stream"

01001110001001001000100010111111000101010
10010100101111111111000001111000000001000
10000100100000010000111000111111000011111
01010101010101101010101101011101101101110
```

This response shows the server sending the news1.data file.

The server may also send multiple data files in a single response. This is done with multiple `Sent` lines in the same way as the client sends resources to the server with multiple `Put` lines.

```
MSTP/0.1
StatusCode: 1000
ResponseId: 234571
Sent: news2.data; ackId=20060619111200NEWS2; offset=0; length=160;
type="application/octet-stream"
Sent: news3.data; ackId=20060619111200NEWS3; offset=160; length=160;
type="application/octet-stream"

01001110001001001000100010111111000101010
10010100101111111111000001111000000001000
10000100100000010000111000111111000011111
01010101010101101010101101011101101101110
01001110001001001000100010111111000101010
01010101010101101010101101011101101101110
10010100101111111111000001111000000001000
10000100100000010000111000111111000011111
```

This response shows the server sending the news2.data.and news3.data files. In the body of the response the data is shown in a different colour. The client knows which body data is for which data file because of the offset and length parameters on the `Sent` line.

If a client request was for a certain range of data, and the server sends only this range of data, then the server response must indicate which range of data has been sent.

```
MSTP/0.1
StatusCode: 1000
ResponseId: 234572
```

```
Sent: news1.data; ackId=20060619111200NEWS1; offset=0; from=160; length=40;
type="application/octet-stream"

0101010101010110101010110101110110110110
```

This response shows that the data returned at is the data from byte 160 to the end. There are 40 bytes of this data, and they are positioned at 0 bytes into the body of the data (i.e. the start of the body).

Note that the offset value is an index into the body of the data and is not to do with the range values.

The range `to` value may be used in a `Sent` line to show that the data in the response does not go to the end of the data file.

The client should always read the response headers and use these to process the data rather than the client sent request headers. This is because the range requested might not be the range returned if the server has a reason to return the full data file.

The server may also send `Sent` lines for data is wants to push to the client. This is done by the server sending a `Sent` line that the client had not sent a corresponding `Get` line in the request.

```
MSTP/0.1
StatusCode: 1000
ResponseId: 234573
Sent: news1.data; ackId=20060619111230NEWS1; offset=0; length=40;
type="application/octet-stream"
Sent: command.data; ackId=20060619111230NEWS2; offset=40; length=40;
type="application/octet-stream"

0101010101010110101010110101110110110110
0100111000100100100010001011111000101010
```

This response shows the server sending the news1.data and the command.data file. Any pushed data should always follow the requested data in the response body.

3.2.8 Acknowledgement requests

If the server has sent data to the client, and then in the next request with a different request id from the client the server did not receive an acknowledgement for that data, then the server can ask the client to acknowledge whether it has received the data or not.

This is done by the server sending a `AckRequired` line in the response.

```
MSTP/0.1
StatusCode: 1000
ResponseId: 234574
AckRequired: 20060619111230NEWS2
```

This is an example of the server asking the client to acknowledge the data file that was previously sent with an acknowledgement id of `20060619111230NEWS2`.
Note that the server is not required to ask for acknowledgements of data files, the client should send the automatically. The `AckRequired` line is used when the connection is less than perfect and a previously sent acknowledgement from the client has not reached the server for some reason.

4. Connection levels

Clients will have varying levels of connection speed, reliability, bandwidth and latency.

Each client data object request will have a predefined priority level associated with it.
The client will dynamically change it's connectivity level threshold based upon the available bandwidth and the number of successful connections.

The priority levels are

- IMMEDIATE – The client must send this request immediately, and not queue this request. This should be used for requesting data objects that are required to show the screen requested by the user.
- SOON – This client may send this request immediately if the network speed / bandwidth is available. This information is useful to the server in deciding what data objects to push to the client.
- WHENEVER – The client does not need to need to send this information to the server in any time critical period. The server needs to be informed of this information, but the data can be sent along with the next request.

The client can calculate its bandwidth based upon the time it takes to transfer a large amount of data. This is probably best done when transferring an audio file.
The client can calculate its connectivity threshold based upon the bandwidth and the number of successful connections, and the number of connections that are interrupted to send a higher priority request.

Clients with a good connectivity will have a connectivity threshold that allows all messages of priority SOON or above to be sent immediately.

Client with a poor connectivity will have a connectivity threshold of that only allows IMMEDIATE messages to be sent immediately.

5. Command data objects

In MusicStation MIDP 0.4.6 the only data objects that exist are content data objects. A new type of data object is required in the connected MusicStation version. These are required so that the server can request or send various data to the client. These data objects are sent by the MusicStation Transfer Protocol, but are not part of the MusicStation Transfer Protocol. They are not part of the protocol because different command objects will be used for different client implementations, but the same transfer protocol will be used for all implementations.

5.1 Server command data objects

In addition to content data objects and image files the server needs to be able to send the following commands to the client

- Please send total file space size to the server.
- Please send remaining file space size to the server.
- Please send log file to the server.
- Please send errors to the server.
- Please change client logging level.
- Set property.
- Get property.
- Please delete a file.
- Please send details of what files you have.
- Please send bandwidth details.
- Please change connection level.

- Please request data file.
- Please request audio file.
- Registration data.

Items are likely to be added to this list as new functionality is added to the client and server.

It's worth noting that the server will frequently be more aware of the client's connection details than the client is. For example a MIDP client on a Nokia N80 has no way of knowing whether the HTTP connection is via an operator gateway, or whether it's via a Wireless LAN. The server will know whether the client connection is via an operator gateway because the connection will be coming from a known operator IP address range.

5.2 Client command data objects

The client needs to be able to send the following data to the server

- Playlist data.
- Image files.
- Total file space.
- Available file space.
- Log file data.
- Current logging level.
- Error info.
- Info on which files have been deleted to free up space for other files.
- Current bandwidth level.
- Data file used.
- Screen shown.
- Property value.
- Current client time.
- Registration data.

Items are likely to be added to this list as new functionality is added to the client and server.

5.3 Timings

The server will be recording the time that various client events have occurred.
Clients should report times to the server in the number of seconds since midnight GMT on 1st January 1970.

For Example, in MIDP1.0, this can be obtained by

```
Calendar.getInstance(TimeZone.getTimeZone("GMT")).getTime().getTime()
```

The MIDP specification says that the GMT timezone must be supported, but if for some reason it is not supported, then the handset can simply use

`(new Date()).getTime()` to get the client time.

Each client stores timing data using it's own time settings. When this data is transferred to the server, the server can then convert and store these event timings in it's own format.

The server will do this by comparing the clients local time, with it's own time. The delta between client reported times and server times can then be calculated.

The current client time command object must contain the time that the data is sent to the server so that the server calculated time is as accurate as possible.

6. Example usage

6.1 Client reading the inbox news.

1. Client request.
The client makes the first request of the session with no session id. The request is for inbox data, and also sends some error data to the server.

```
MSTP/0.1
RequestId: 0123456789ABCDEF
Client: MusicStation 0.4.5 MIDP Nokia/N70
Get: inbox.data
Put: error.data; ackId=CLIENT1; offset=0; length=102; type="application/octet-stream"
UserId: Xatq3kf4ImdlE3TD

[ERROR DATA IN BODY]
```

2. Server response.
The server responds with a responseId matching the requestId in the client request. The inbox data is sent as requested. A session id is assigned for use in the rest of the session, and the error data is acknowledged.

```
MSTP/0.1
StatusCode: 1000
ResponseId: 0123456789ABCDEF
Sent: inbox.data; AckId=SERVER1; offset=0; length=102; type="application/octet-stream"
SetSessionId: ZYXWVUTSRQPONMLK
Ack: CLIENT1

[INBOX DATA IN BODY]
```

3. Client request.
The client makes a request for news1.data, and acknowledges the receipt of the inbox data. The session id assigned in the last request is now included in the response.

```
RequestId: 0123456789ABCDF0
Client: MusicStation 0.4.5 MIDP Nokia/N70
Get: news1.data
Put: pageView1Data; AckId=CLIENT2; offset=0; length=102; type="application/octet-stream"
Ack: SERVER1
UserId: Xatq3kf4ImdlE3TD
SessionId: ZYXWVUTSRQPONMLK

[PAGEVIEW1 DATA IN BODY]
```

4. Server response.
The server responds with the news1 data that was requested.

```
MSTP/0.1
StatusCode: 1000
ResponseId: 0123456789ABCDF0
Sent: news1.data; AckId=SERVER2; offset=0; length=102; type="application/octet-stream"
Ack: CLIENT2

[NEWS1 DATA IN BODY]
```

5. Client request.
The client makes a request for news2 data. Note that the client has for some reason not acknowledged the news1.data with acknowledgement id of SERVER2 *that it previously received.*

```
MSTP/0.1
RequestId: 0123456789ABCDF1
Client: MusicStation 0.4.5 MIDP Nokia/N70
Get: news2.data
Put: pageView2Data; AckId=CLIENT3; offset=0; length=102; type="application/octet-stream"
UserId: Xatq3kf4ImdlE3TD
SessionId: ZYXWVUTSRQPONMLK

[PAGEVIEW2 DATA IN BODY]
```

6. Server response.

The server responds with the news2 data that was requested. The server also asks the client to acknowledge the previously send news1 data that it did not receive and acknowledgement for.

```
MSTP/0.1
StatusCode: 1000
ResponseId: 0123456789ABCDF1
Sent: news2.data; AckId=SERVER3; offset=0; length=102; type="application/octet-stream"
Ack: CLIENT3
AckRequired: SERVER2

[NEWS2 DATA IN BODY]
```

7. Client request.

The client makes a request for news3 data and acknowledges the news1 and news2 data files which had the ids SERVER2 *and* SERVER2

```
MSTP/0.1
RequestId: 0123456789ABCDF2
Client: MusicStation 0.4.5 MIDP Nokia/N70
Get: news3.data
UserId: Xatq3kf4ImdlE3TD
SessionId: ZYXWVUTSRQPONMLK
Put: pageView3Data; ackId=CLIENT4; offset=0; length=102; type="application/octet-stream"
Ack: SERVER3
Ack: SERVER2

[PAGEVIEW3 DATA IN BODY]
```

8. Server response.

The server responds with the news3 data that was requested. The server also sends two commands to the client. These commands tell the client to request the news4 and new5 data files because the server thinks that the client may need these files soon. Alternatively the server could have pushed the news4 and news5 data files, but has chosen to tell the client to request them at its convenience. This allows the client to decide whether it has the correct connectivity level available to request these files.

```
MSTP/0.1
StatusCode: 1000
ResponseId: 0123456789ABCDF1
Ack: CLIENT4
Sent: news3.data; AckId=SERVER4; offset=0; length=102; type="application/octet-stream"
Sent: command1.data; AckId=SERVER5; offset=102; length=34; type="application/octet-stream"
Sent: command2.data; AckId=SERVER6; offset=136; length=34; type="application/octet-stream"

[NEWS3, COMMAND1 & COMMAND2 DATA IN BODY]
```

6.2 Client searching for U2

1. Client request.
The request is for search results data, the client also sends some page view data to the server.

```
MSTP/0.1
RequestId: 0123456789ABCDEF
Client: MusicStation 0.4.5 MIDP Nokia/N70
Get: searchData?type=artist&searchTerm=u2
SessionId: ZYXWVUTSRQPONMLK
UserId: Xatq3kf4ImdlE3TD
Put: pageViewData; ackId=98; offset=0; length=102; type="application/octet-stream"


[PAGE VIEW DATA IN BODY]
```

2. Server response.
The server responds with a responseId matching the requestId in the client request. The search results data is sent as requested, and the page view data is acknowledged.

```
MSTP/0.1
StatusCode: 1000
ResponseId: 0123456789ABCDEF
Sent: searchData?type=artist&searchTerm=u2; AckId=U2SEARCHRESULTS; offset=0; length=102;
type="application/octet-stream"
Ack: 98

[SEARCH RESULTS DATA IN BODY]
```

6.3 Client making a playlist public

1. Client request.
The request contains the playlist data object which has been modified, the client also sends some page view data to the server.

```
MSTP/0.1
RequestId: 0123456789ABCDEF
Client: MusicStation 0.4.5 MIDP Nokia/N70
Put: playlist.data; ackId=ABC; offset=0; length=102; type="application/octet-stream"
Put: pageViewData; ackId=DEF; offset=102; length=102; type="application/octet-stream"
SessionId: ZYXWVUTSRQPONMLK
UserId: Xatq3kf4ImdlE3TD

[PLAYLIST AND PAGE VIEW DATA IN BODY]
```

2. Server response.
The server responds with a responseId matching the requestId in the client request. No data is sent, but the received data is acknowledged.

```
MSTP/0.1
StatusCode: 1000
ResponseId: 0123456789ABCDEF
Ack: ABC
Ack: DEF
```

6.4 Client requesting Name That Tune game

1. Client request.
The request contains the request for the name that tune game, the client also sends some page view data to the server.

```
MSTP/0.1
RequestId: 0123456789ABCDEF
Client: MusicStation 0.4.5 MIDP Nokia/N70
Get: nameThatTune
Put: pageViewData; ackId=DEF; offset=102; length=102; type="application/octet-stream"
SessionId: ZYXWVUTSRQPONMLK
UserId: Xatq3kf4ImdlE3TD

[PAGE VIEW DATA IN BODY]
```

2. Server response.
The server responds with a responseId matching the requestId in the client request. The name that tune game data is sent, along with some pushed data which the client will require for the game to be played.

```
MSTP/0.1
StatusCode: 1000
ResponseId: 0123456789ABCDEF
Sent: nameThatTune; AckId=NAMETUNE; offset=0; length=102; type="application/octet-stream"
Sent: 30seconds.mp3; AckId=AUDIO; offset=102; length=1024; type="application/octet-stream"
Ack: DEF

[NAME THAT TUNE AND AUDIO DATA IN BODY]
```

6.5 Client has created a new playlist

1. Client request.
The request contains the newly created playlist.

```
MSTP/0.1
RequestId: 0123456789ABCDEF
Client: MusicStation 0.4.5 MIDP Nokia/N70
Put: newPlaylist; ackId=1; offset=0; length=102; type="application/octet-stream"
SessionId: ZYXWVUTSRQPONMLK
UserId: Xatq3kf4ImdlE3TD

[NEW PLAYLIST DATA IN BODY]
```

2. Server response.
The server responds with a responseId matching the requestId in the client request. The playlist data is acknowledged, and the new playlist object is pushed back to the client with the server GUID property populated.

```
MSTP/0.1
StatusCode: 1000
ResponseId: 0123456789ABCDEF
Sent: newplaylist; AckId=PLAY939; offset=0; length=102; type="application/octet-stream"
Ack: 1

[SERVER STORED PLAYLIST DATA IN BODY]
```

7. Suggested Implementation

The initial implementation of this protocol will be for MIDP clients to communicate with the server. The initial MIDP application will use HTTP as the transport layer for our protocol.

7.1 Client implementation

7.1.1 Data object expiry

Data objects can be expired by the both the client and the server.
Data objects that are to be expired by the client must contain an expiry date as one of their parameters. The client can then check this expiry date to see if the object is valid before using it. This client expiry of data objects is completely external to the MusicStation Protocol. Similarly data objects may have a valid from property, and the client can check this to see if the data object they are about to use is valid yet.
The server may expire data objects by one of three methods.

1) For a data object which has an expires time property, the server can push an updated version of this data object to the client. The client expires functionality will then take care of the expiry.
2) The server can tell the client to delete a data object which is no longer valid.
3) The server push to the client an updated version of the data object which is no longer valid.

Note that all these expiry cases use the protocol to transfer data, but the protocol is not responsible for the expiration.

7.1.2 Connection levels

When the client has data to send to the server that has a connection priority below that of the current connection level, then the client will have to queue this data for sending to the server.

How much of the data can be queued obviously depends upon the particular client.

If the queue of data has become full without a connection to the server being required, then the client will probably want to try an immediate request to the server at this point. This is even though a request at a high enough connection level has occurred. Depending upon the size of this queue the client may want to try and empty this queue at either regular intervals, or when it reaches a certain size. For example if there are five objects on the queue waiting to send to the server. In general, the client should treat several lower priority requests the same as one higher priority request.

If a client with a full queue can not make a successful connection to the server, then it may have to start deleting data from this queue to make space for further data.
The client may choose to delete the oldest data first, or there may be data which is non-mandatory to send to the server which can be deleted.

7.1.3 Music Server connection

The MusicStation Transfer Protocol is not designed for transfer of audio files from the music server to the client.

The MusicStation server will have control over what music files are downloaded from the music server though. This is done by the MusicStation server sending command objects to the client telling the client which files it should pull in from the music server.

7.1.4 Client registration

Client registration is handled by command and/or data object transfer. This is because the data submitted to the server for registration will be different depending upon the client implementation. The server can respond with command and/or data objects signifying the success of failure of the registration.

7.2 Server implementation

7.2.1 HTTP access

The initial server implementation will expose the server functionality via HTTP. The protocol handling functionality should not be tightly integrated to HTTP though, so as to allow for other transport layers to access the functionality with the minimum of change.

A Servlet interface to the protocol may be

```
// Create a MSTP request object from the HTTP data
MSTPRequest request = new MSTPRequest(servlet.getInputStream());

// Get the session for this requestor
MSTPSession session = SessionManager.getSession(request.getSessionId());

// Pass this request and session to our Request Engine, it will return the
response
MSTPResponse response = engine.process(request, session);

// Send this response as the servlet result
servlet.getOutputStream().write(response);
```

The main processing functionality is all in the `engine.process()` method. This same method can be used for access via any other transport layer.

For a HTTP transport layer, it would probably be best to use the Java EE web container's session management (i.e. Tomcat) because it provides all the required functionality.
(See the `findSession()` method in `org.apache.catalina.session.ManagerBase`)

Alternatively the Servlet Session API could be used by duplicating the session id in the URL that the protocol is accessed from. Duplicating the session id in the URL does have the advantage of allowing the HTTP interface to the system to be easily load balanced. Load balancers can usually look at a URL to keep session affinity with the same server. Duplicating the data obviously introduces the possibility of inconsistency, although this should be easy to manage.

The access URL for the HTTP interface will be the same URL each time and for each user.

For example

```
http://musicstation.omnifone.com/mstpHandler.do
```

7.2.2 Pushing objects

The server has two different ways of pushing data to the client.

1) Including a data object in a response will force the client into receiving an updating that resource immediately.
2) Including a command object in the response will ask the client to request the data object at it's convenience.

The server should choose the most suitable method of pushing the data based upon the client's properties, and the data object properties.

Relatively small data objects should almost always be pushed straight to the client.
Data objects that must be updated, should always be pushed straight to the client.
Clients with a low connection level should be asked to pull the data object.
Clients with low memory available should be asked to pull the data object.

We maintain a list of the tracks that are on the memory card in the order that they were last used. This list also contains the total file size and the number of bytes downloaded so we can tell if a file has only partially downloaded. This list is stored on the memory card in the same directory as the track files so that if the user swaps memory cards the list is kept with the tracks files. Also whenever the user opens a playlist, a track list or an album we verify that the list is correct.

We try to leave at least 10MB space on the memory card. This is to protect against the phone incorrectly telling us how much space is actually available. Before we download a track we first check to see if there is enough room. If there isnít room we delete the track files at the end of the list until there is room. We then download the track to the memory card

Data Objects are the basic unit of object that is passed between the server and client and client and server. They encapsulate the representation of some entity which is displayed within the client interface (such as an artist, album, etc) or data which needs to be sent back to the server (like a user-defined playlist)

They are passed between the server and client and are stored securely on the phone

Data objects can be delivered to the client by the server at anytime that it needs to update something on the client

From backward Compatibility doc:

2.2 Data Objects

The data objects are able to write themselves to a file and these are used to transfer data between the client and server. The file header contains the data object version used to write the file. The latest versions of the data objects are able to read and write files in all supported versions. The version is passed to each read and write method and this allows us to switch what gets read or written based on the version.

Using this method the server is able to write data object files for older versions of the client. The target version is set in the file header and then each write method ensures that the output is in the format for that version.

The server is also able to read files written by an older client using the same method. When the file is read into the data object the read method uses the version to switch what attributes are read from the file.

- Each piece of content is "tagged" with the container, format, bitrate and sample rate (e.g. m4a, acc+, 48kbps, 44.1kHz).
- The playback of content is tested on the device using a piece of base content (pink noise) encoded in all variants of container, format, bitrate, sample rate and mime type.
- The results of these tests are sent via the test client back to the server and stored.
- Each container, format, bitrate, sample rate and mime type has a preference when compared to the others stored on the server.
- When the client then makes a request for an additional piece of content the server returns either:
    - A list of links to that piece content encoded in the variations of container, format, bitrate, sample rate and mime type that played back. This is done by matching the "tags" on the content that played back to the available "tags" on the additional piece of content. This list is ordered by preference.
    - A link to that piece of content encoded in the top encoding preference.
- A audio playback quality test is performed on this auto-selected item to confirm that it is of acceptable quality. If it is not then the second preference would be examined, and so on down the list. The audio quality test makes use of audio software to analyse the output of the phoneís headphone socket.

MusicStation Handset Commissioning

Chris Knowles – 13 December 2006

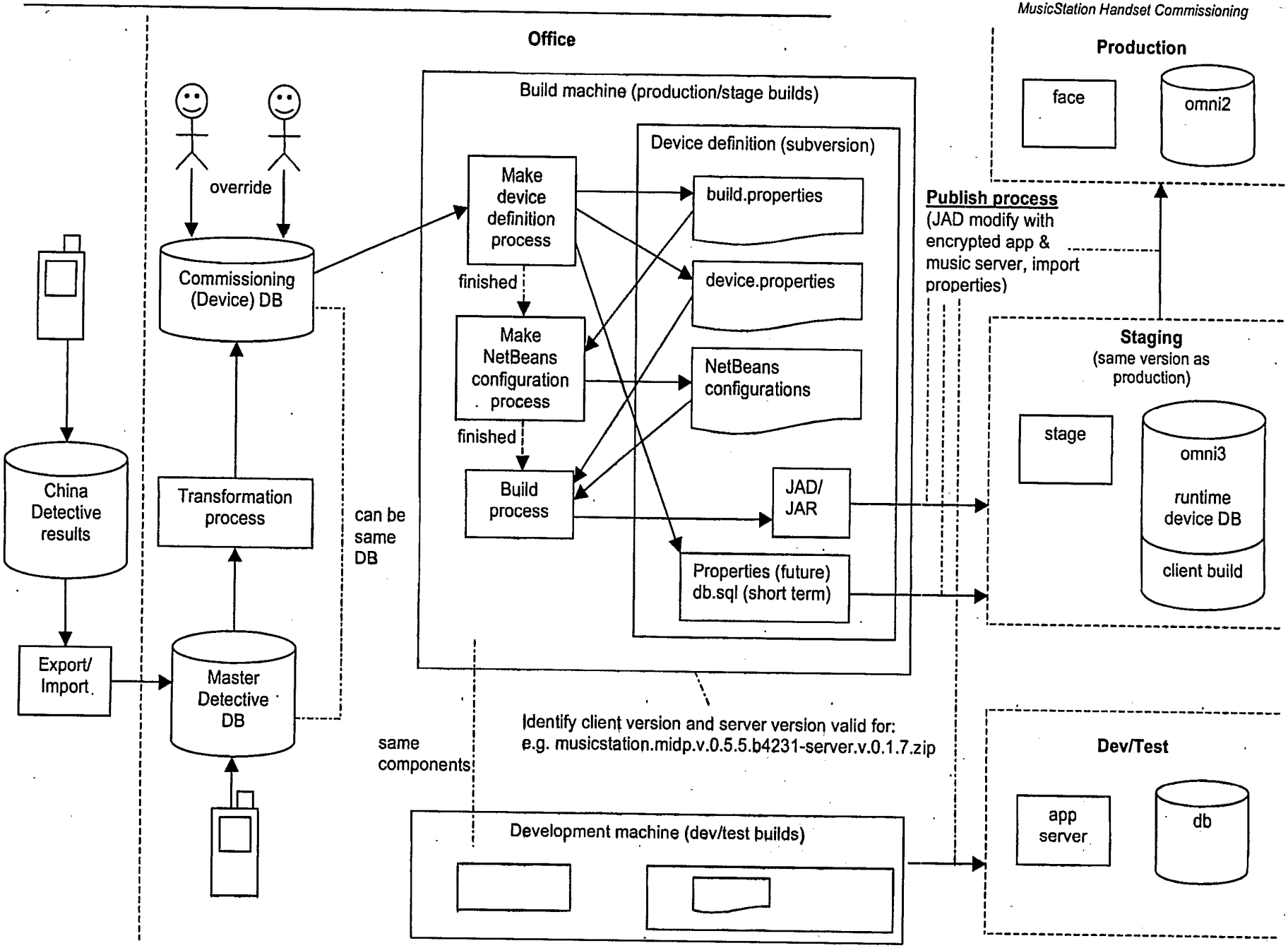
Table of Contents

1. Introduction

This document gives an overview of the Detective data model, the MusicStation device data model as well as transformations involved between the two. It describes the end-to-end process by which data collected by the Detective across handsets is used in order to build MusicStation clients for those handsets.

2. Overview

The following diagram gives an overview how the Detective and MusicStation interact to reach the end results of releasing a build.



3. Overall Process

1. Detectivise
2. Transform process
3. Manual Override
4. Build process
5. Publish to stage
6. Test (if fail, go to step 3)
    a. If fail due to bad configuration, go to step 3
    b. If fail due to bug which needs client code fix, arrange for bug to fixed on the branch associated with the current production release. If we already detect everything required, return to step (4) and build against the branch. Otherwise either:
        i. put in queue waiting for detective fix and return to step(1)
        ii. return to step (3) and enter new details
    c. In unlikely event that fails due to bug which needs server-side code fix, then this handset must be put in the queue to wait for that server-side fix. Once new server release then return to (2).
7. Publish to production

4. Data Model Overview

There are three main parts to the handset data model:

1. Detective data – this is the raw test data collected.
2. Commissioning data – this is a transformed version of the Detective data. It includes everything required by the MusicStation data model plus several other components such as information required at build time.
3. MusicStation data – this is all the data required to identify a handset and anything required by MusicStation at runtime.

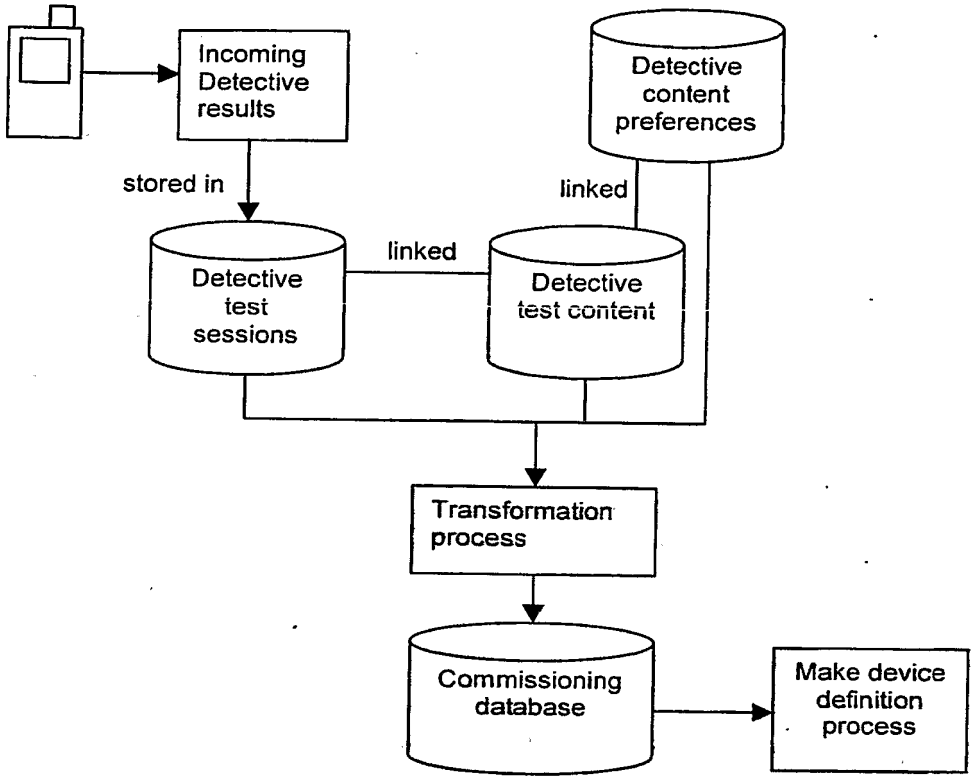
5. Master Detective Database

This database stores the raw detective data in "sessions". Each session represents a set of tests performed at a particular time by a particular tester. Each piece of data is linked to a particular version of the Detective.

This database is centralised (i.e. it is not deployed on separate machines such as development machines for actual commissioning).

6. Detective/Commissioning

The following diagram gives and overview of the device model with regard to the Detective and Commissioning databases.



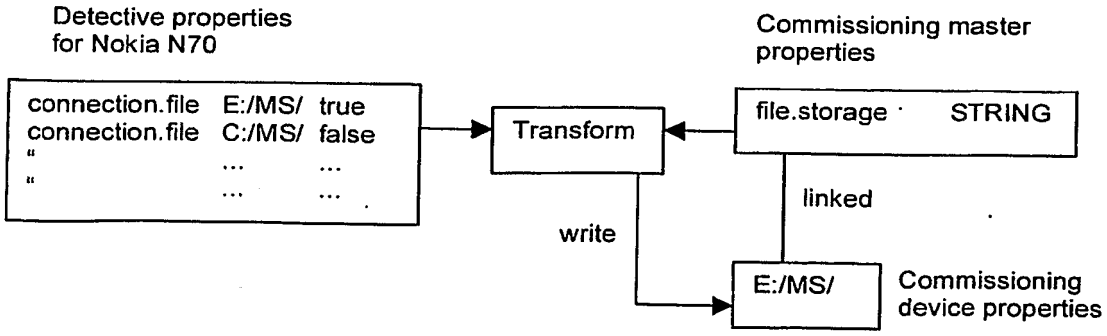
6.1 Transformation process

Transformations have to be performed between the Detective data model and the Commissioning device data model.

For example, the Detective finds out the drives names via Java and then collects the following data:

- Can we write to the root of this drive?
- Can we write to known directories such as "music_files" in this drive?

However, the Commissioning master property is simply "where should we be writing to?" The process for working this out is not just a simply copy. In fact, it may require manual intervention by the commissioner due to the fact that we want to be writing to the memory card, but we only have so many known memory card drive names (i.e. we cannot be certain that there will not be a new one). Another example would be canvas height, which can simply be copied.



This process is internal to the detective. It is therefore centralised. It takes the raw detective data and changes it in such a way as to complete the information required in the commissioning database. This can consist of:

- Raw detective data simply being copied.
- Detective data is transformed (e.g. where we can write to is transformed into where we should write to).
- Raw detective data being aggregated.

There are several outcomes for these:

- Raw data is incomplete and the device must be tested more.
- The process can automatically determine information for the commissioning database.
- Manual intervention is required (e.g. we cannot automatically determine which device drive is the memory card and which is the phone's memory).
- Manual intervention is required because of a device or detective bug or a particular piece of information is not collected yet.

As this process is part of the detective, any revisions to the structure of the master detective database or the data contained within it are also reflected here. Any changes to the commissioning database are also reflected here, but that is also centralised so we are only dealing with a single version.

6.2 Commissioning Database

This database stores two sets of data. A dictionary of properties required for defining a device and the actual properties for all handsets that have been commissioned.

The dictionary consists of:

- The definitions of the properties needed to make a device definition (i.e. including run time properties and build time properties as well as other items such as device_code). This is so the "make device definition process" knows if it has everything it needs to make the definition and also so the "transformation process" knows what it needs to write.
- Which properties relate to particular "bands" of client and server versions. For example, a property "supports.bluetooth" may only be valid on client versions v0.6 and above and server versions v0.6-v0.8. This is similar to the "device_capabilities" from the MusicStation data model, but includes definitions for all device properties, not just runtime ones.

The per device data consists of:

- Values for these properties for particular devices and how they were set. Each property can have one value from each type of source (i.e. one from transformation process, one manual). Each manual value should have a reason of why it is set this way.

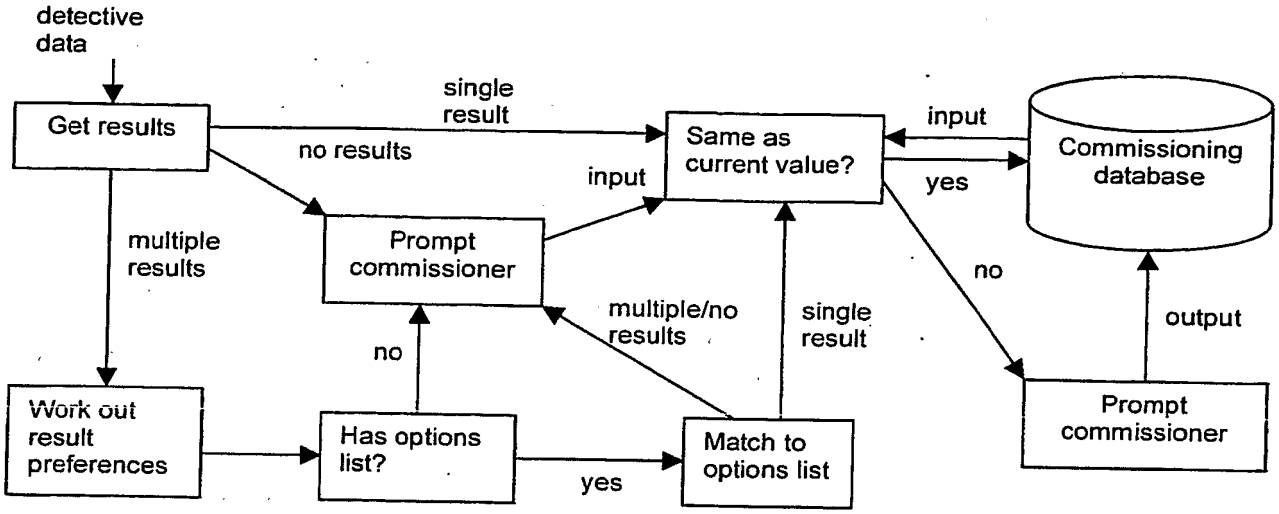
This database is centralised (i.e. it is not deployed on separate machines such as development machines for actual commissioning).

6.3 Transformation Process Implementation

For the time being, the transformation process will be performed in code but we could use a generic data driven process. The process is essentially driven by two pieces of information:

- Mappings between Detective properties and Commissioning ones
- Known options for particular Detective properties (e.g. for memory card we'd want to pick up "E", "Memory Card", "efs" automatically).

The transformation process for a particular Detective property works as follows:



At the start of the process, we can potential have multiple results for the same Detective property. The following preference order is applied in this order:

1. More recent Detective version
2. Higher certificate domain (e.g. manufacturer domain preferred over third party trusted domain)
3. Higher Detective variant (e.g. variants with extra components request extra permissions)
4. Later date performed

Even after going through these preferences for a set of results, it is still possible to have more than a single result left. There are two follow-ups to this:

1. Automatically look at a list of possible options for this result if it exists
2. Prompt the commissioner to choose from the set of results

If we have a list of options for a particular result, we automatically go through our set of results and match to the list of available options, working out from the priorities of the options. There are two outcomes for this automatic process:

1. No results in the set match the options
2. Single top result found

If we have found a single result then we are ok, but if we have no results then this means we cannot automatically choose and we need to prompt the commissioner to choose between the results in the set given to the this automatic process.

Whenever setting a property, the process will look for any current value in the Commissioning database. The following rules then apply:

- If there are no current values then we are safe to write.
- If there is only a one current value and it is identical to the outcome of the transformation process we are safe to write (with a new modified timestamp).
- If there is only one current value and it is different to the outcome of the transformation process then we need to highlight this to the user to check if it seems correct.
- If the current manual and automatic values are identical to the outcome of the transformation process we are safe to write (with a new modified timestamp).
- If the current automatic value is identical to the outcome of the transformation process but the manual one is not then we need to highlight to the user that the property was manually overwritten last time and provide the reason for the overriding given in the database.
- If the current manual value is identical to the outcome of the transformation process but the automatic one is not then we just need to highlight to the user that the property isn't changing but that it has been transformed differently this time.

We will need to have one table that holds the mapping of Detective properties to Commissioning ones, but we also need a generic way to hold a list of options that a Detective property can be (e.g. for memory card we'd want to pick up "E", "Memory Card", "efs", etc). We therefore need another table that links these options to the correct Detective properties.

Some examples of transformation operations are:

- copy – this just copies the Detective property to the corresponding Commissioning one
- result query – this transforms via a query and returns the Detective result (such as E:/MS/)
- content query – this transforms via a query and returns the content linked to the Detective result (such as Sugababes, e-aac+, m4a, 48kbps, 44kHz)

We can perform each query with or without matching to available options. e.g. we can find all connection.file results or all connection.file results that wrote to known places. If a particular property does have options, and the full query returns no results, then we can perform the same query but without matching to available options and provide this set of results to the commissioner to choose from.

Please note, queries below are not exact SQL.

Transformations:

	Example 1	Example 2	Example 3
detective_property	"connection.file"	"display.canvas.height"	"audio.playback"
detective_version	0.57	0.57	0.57
property	"cache.phone.url"	"canvas.height"	"file.format"
operation	"result query"	"copy"	"content query"
query	select detective_property where result is true	*null*	select detective_content where property_value = true and top preference

others operations could be translate, add, etc.

List of options:

	Example 1	Example 2	Example 3
detective_property	"connection.file"	"connection.file"	"connection.file"
value	"E"	"Memory Card"	"efs"
priority	1	2	3

6.3.1 Property Mappings

The actual Detective to Commissioning mappings used by the transformation process are detailed in this section:

detective_property_transformation

detective_property	property	operation	query
connection.file	file.storage	query-result	select r.value from detective_results r where r.result = 'true'
display.canvas.height	canvas.height	copy	*null*
display.canvas.width	canvas.width	copy	*null*
audio.playback	file.container	query-content	select c.container from detective_results r, detective_content c where r.result = 'true' and c *is top preference*
audio.playback	file.format	query-content	select c.format from detective_results r, detective_content c where r.result = 'true' and c *is top preference*
audio.playback	file.rate.bit	query-content	select c.bit_rate from detective_results r, detective_content c where r.result = 'true' and c *is top preference*
audio.playback	file.rate.sample	query-	select c.sample_rate from detective_results r,

		content	detective_content c where r.result = 'true' and c *is top preference*

detective_property_options

detective_property	value	priority
connection.file	E	1
connection.file	Memory card	2
connection.file	e	3
connection.file	Ms	4
connection.file	Nand	5
connection.file	SD	6

7. Building clients

A client build process is initiated when the commissioner wants to build a client.

The process consists of three separate processes, which can be performed independently or one after the other:

1. Make device definition process
2. Make NetBeans configuration process
3. Build process

See the sections below for more detail on these processes.

The information required is:

- Device
- Subversion revision (there are several subversion revisions for each client version – this way we can build with a client-side bug fix included)
  - We should provide a list of versions along with each revision of that version
- Server version
  - We should provide a list of which server are on which versions
- Service id
  - We should provide the names of the services

This process is deployed in two places:

- centrally on the main build machine for current production/stage clients
- on a test build machine for current test builds (latest revision)

The version deployed on the test build machine is backwards compatible with all client versions and the current deployed server version on stage/production (in the same way that the current server release must be backwards compatible with all previously released clients). This process is updated when the current client/server test builds are updated. It changes if any client/server changes require new objects/properties/a change in structure to the device definition. If changes to the commissioning database require changes here, a new version of this process is deployed at the same time as the new commissioning database is deployed.

The version on the main build machine does not change until there is a new release to stage/production and we are happy with the current build process.

The latest build of this process can also be deployed to development machines for use when it is not been released yet to the centralised machine (i.e. the server/client version it is dealing with has not yet been released either).

The device definition files end up on the same machine the process runs on, in a folder named the same as the device code (e.g. sony-ericsson-k800i). Device definitions can only be checked in from the centralised machine. Each resides in a directory structure named as such:

<service id> / <server version> / <client version> / <client build> /

e.g.1/v0.1.7/v.0.5.5/b4231

So we can have the following checked in:

- The definition used for the current stage/production server/client build.
- The definition for the latest server/client test build.

Redundant server versions can be archived at a later date.

7.1 Device definition

The device definition consists of:

- A build.properties file:
  - A list of build time properties such as icon size and screen size
  - Timestamp
  - The subversion revision (encapsulates client version)
  - Used to make NetBeans configuration
- A device.properties file:
  - A list of properties used by the client at runtime
  - Timestamp
  - The subversion revision (encapsulates client version)
  - Packaged inside the JAR at build time
- A NetBeans configuration file – see section below
- A JAD – without server URLs
- A JAR
- A runtime.properties file – a list of runtime information required for the target DB (in the short term, just SQL to import runtime data for the handset into the target DB). This includes device name, code and runtime properties as well as a timestamp and the server/client version and client build numbers generated for.

7.1.1 Examples

build.properties:

```
#Properties for SonyEricssonW550i
#Mon Dec 04 09:17:00 GMT 2006
#Client revision 4837
certificate=verisign
icon.height=32
icon.width=32
screen.height=220
screen.width=176
```

device.properties:

```
#Properties for SonyEricssonW550i
#Mon Dec 04 09:17:00 GMT 2006
#Client revision 4837
AUDIO_CONTENT_TYPE=audio/3gp
AUDIO_FILE_EXTENSION=.3gp
CARD_CACHE_URL=c:/Other/MusicStation/
DEVICE_GUID=d38b1ce80a00006761a3912143c10178
KEYCODE_BACK_KEY=-11
KEYCODE_DELETE=-8
KEYCODE_GAMEACTION_DOWN=-2
KEYCODE_GAMEACTION_FIRE=-5
KEYCODE_GAMEACTION_LEFT=-3
KEYCODE_GAMEACTION_RIGHT=-4
KEYCODE_GAMEACTION_UP=-1
KEYCODE_LEFT_SOFTKEY=-6
KEYCODE_RIGHT_SOTFKEY=-7
PHONE_CACHE_URL=c:/Other/MusicStation/
```

To make the import process as simple as possible to begin with runtime properties will be named directly after the tables and columns of the data (e.g. device.guid). In the future the import process will handle this a bit more intelligently.

7.2 Make device definition process

This process takes the data from the commissioning database and creates three property files:

- build.properties
- device.properties
- runtime.properties (or in the short term, an SQL file for importing the same data)

7.3 NetBeans configuration files

The NetBeans configuration for a particular handset is made using the build.properties for that handset. The configuration files can contain configurations for more than one handset. Individual NetBeans configuration files for each handset will be checked in. If we are building multiple clients, then an overall configuration file will be made encapsulating all these handsets. We may also want to check in some of these overall configuration files for each of our partners (e.g. one for Telenor, one for Vodacom, etc).

7.4 Make NetBeans configuration process

This process either takes the build.properties file for a single device and makes a NetBeans configuration for it, or takes the build.properties files for a set of device and makes an overall NetBeans configuration file. This process may be redundant in the future because the make device definition process may be able to make the NetBeans configurations directly. However, it may also be a good idea to keep the processes separate so we do not have to access the commissioning database to make a new NetBeans configuration.

7.5 Build Process

The build process simply takes a NetBeans configuration file (may be one device or a set of devices) and uses the standard NetBeans ant scripts to build the clients.

8. Publishing Process

In this process the device definition is taken and processed so that it is ready to be used with a particular set of target servers (i.e. a particular MusicStation application server and a particular audio server).

The properties for the target servers are stored in files in a "servers" directory underneath the publishing process. For each target environment, there will be one file. For example:

production.properties:

```
#Properties for production
#Mon Dec 04 10:14:00 GMT 2006
audio=murdock
audio.url=81.89.137.172
musicstation=face
musicstation.url=81.89.137.171
database=omni2
database.url=81.89.137.170
database.username=username
database.password=password
```

8.1 JAD modification

This simply takes a JAD/JAR combination and adds encrypted application and music server URLs to the JAD. This ensures we are using exactly the same JAR for testing as we are for production, just linking it to different servers. The properties are (possibly to change):

- MUSICSTATION_HOST
- AUDIO_HOST

8.2 Deployment

After the JAD is modified, two things need to be done to deploy a client:

1. Copy the JAD and JAR to the target server.
2. Import the runtime.properties file to the target server.

While we are doing part (1), we need to include checksum checks in order to ensure the client is not corrupted while copying across.

In the short term, the runtime.properties is an SQL script which is run on the target server. In the future, this will be an actual properties file and the publishing process will have to know what to do with the different types of property. For example, it will have to know that the device name and code in the file correspond to the device table and that it should insert a new device if there is not already one with the same code. It will also have to know how to deal with device runtime properties, file types, etc.

The runtime.properties will also have to populate the client_build table of the MusicStation database with the information gained at the start of the build process:

client_build

id	integer	PK, NOT NULL
device_id	integer	FK device.id, NOT NULL
service_id	integer	FK service.id, NOT NULL
revision_id	integer	FK client_revision.id, NOT NULL
certificate	integer	FK omnifone_code_signing_cert, NOT NULL
jar_file	string	
jad_file	string	
support_start_date	timestamp	NOT NULL, The date/time that support for this client build started
support_end_date	timestamp	default NULL. The date/time that support for this client ended/will end
desupport_notice_date	timestamp	default NULL. The date/time that users will start being told of planned desupport of their client
support_end_message_set_id	integer	FK message_set.id, NOT NULL
desupport_message_set_id	integer	FK message_set.id, NOT NULL
guid	string	
created	timestamp	
inserted	timestamp	
modified	timestamp	

add any other information that CK has as part of his process

Full list of updates on deployment:

- Add or update a device row (as many columns as possible)
- Add or update device property rows

9. Appendix A

This appendix lists the definitions of the properties in the commissioning database

name	type	use	description	optional
certificate	string	build		no
application.icon.height	int	build	app icon height	no
application.icon.width	int	build	app icon width	no
display.canvas.height	int	build		no
display.canvas.width	int	build		no
AUDIO_CONTENT_TYPE	string	device	e.g. audio/3gp	no
AUDIO_FILE_EXTENSION	string	device	e.g. .3gp	no
cache.card.url	string	device		no

device.guid	string	device		no
key.code.back	int	device	e.g. -11	yes
key.code.delete	int	device		yes
key.code.game.down	int	device		yes
key.code.game.fire	int	device		yes
key.code.game.left	int	device		yes
key.code.game.right	int	device		yes
key.code.game.up	int	device		yes
key.code.softkey.left	int	device		yes
key.code.softkey.right	int	device		yes
cache.phone.url	string	device		no
player.prefetch	boolean	device		no
key.function.zero	string	device		no
key.function.pound	string	device		no

- Empty string *cannot* mean null *and* also empty string in a property file for different properties. The property not being there is null (i.e. PHONE_CACHE_URL should not be present if we are not using it, not the current situation of present but empty)

10. Notes

client_version

id	integer	PK	NOT NULL	
number	integer		NOT NULL	subversion revision number
version	varchar(12)		NOT NULL	The version number as seen by the user
support_start_date	timestamp		NOT NULL	The date/time that support for this client build started
support_end_date	timestamp		default NULL	The date/time that support for this client ended/will end
desupport_notice_date	timestamp		default NULL	The date/time that users will start being told of planned desupport of their client
support_end_message_set_id	integer	FK message_set.id	NOT NULL	
desupport_message_set_id	integer	FK message_set.id	NOT NULL	
comments	varchar(256)			
guid/created/inserted/modified	string			

master_property

id	integer	
name	varchar(12)	
type	property_type_kind	(INTEGER, BOOLEAN, LONG, STRING, FILE)
use	property_use_kind	(BUILD, DEVICE, RUNTIME)
optional	boolean	

guid/created/inserted/modified

master_property_library

id	integer	
master_property_id	integer	FK master_property.id
library_version_id	integer	FK library_version.id

guid/created/inserted/modified

master_property_client_version

id	integer	
master_property_id	integer	FK master_property.id
client_version_id	integer	FK client_version.id

guid/created/inserted/modified

library

id	integer	
name	varchar(64)	

guid/created/inserted/modified

library_version_set_item

id	integer			
library_id	integer	FK library.id	NOT NULL	
number	integer		NOT NULL	revision number
version	varchar(12)		NOT NULL	version

guid/created/inserted/modified

library_version_set

id	integer	
count	integer	
library_version_id	integer	FK library_version.id

guid/created/inserted/modified

client_version

id	integer	
....		
branch	varchar(48)	subversion branch
library_version_set_id	integer	FK library_version_set.id

server_version

id	integer	
...		
library_version_set_id	integer	FK library_version_set.id

MusicStation Client & Server R&D Workflow

Steve Pocock – 12 Jan 2007

Table of Contents

1. Introduction

Sets out the workflows of the processes within R&D involving developing new client and server functionality. The outputs of these processes are new client and server versions, some of which will be made available to operational staff for commissioning new handsets against, which will then be published to live services from which they may then be provisioned.

This document covers the R&D processes. The operational processes are covered in the document MusicStation Client Commissioning and Provisioning Workflow (MusicStation Client Commission & Provision Workflow vX.X.doc)

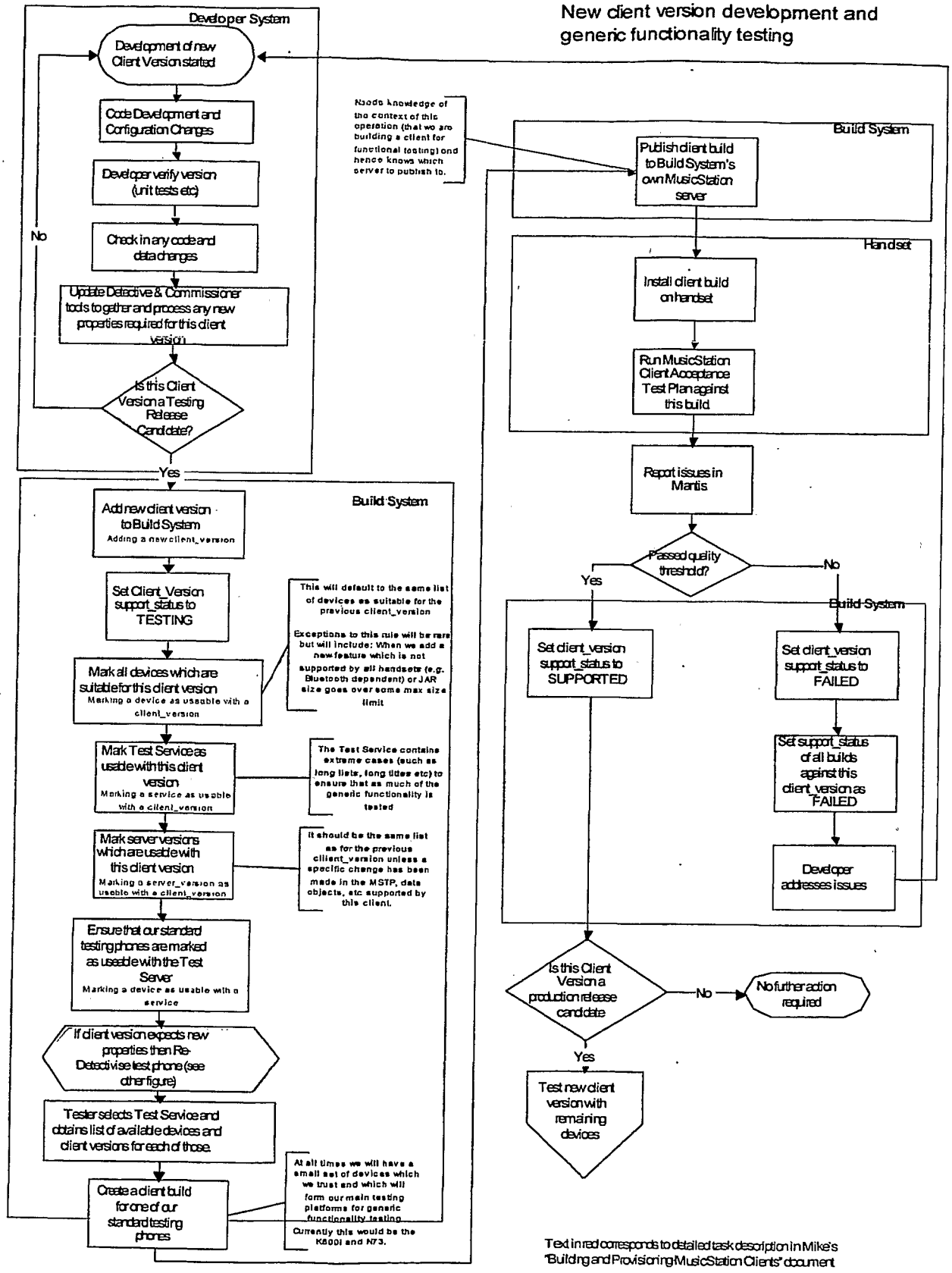
2. Overview

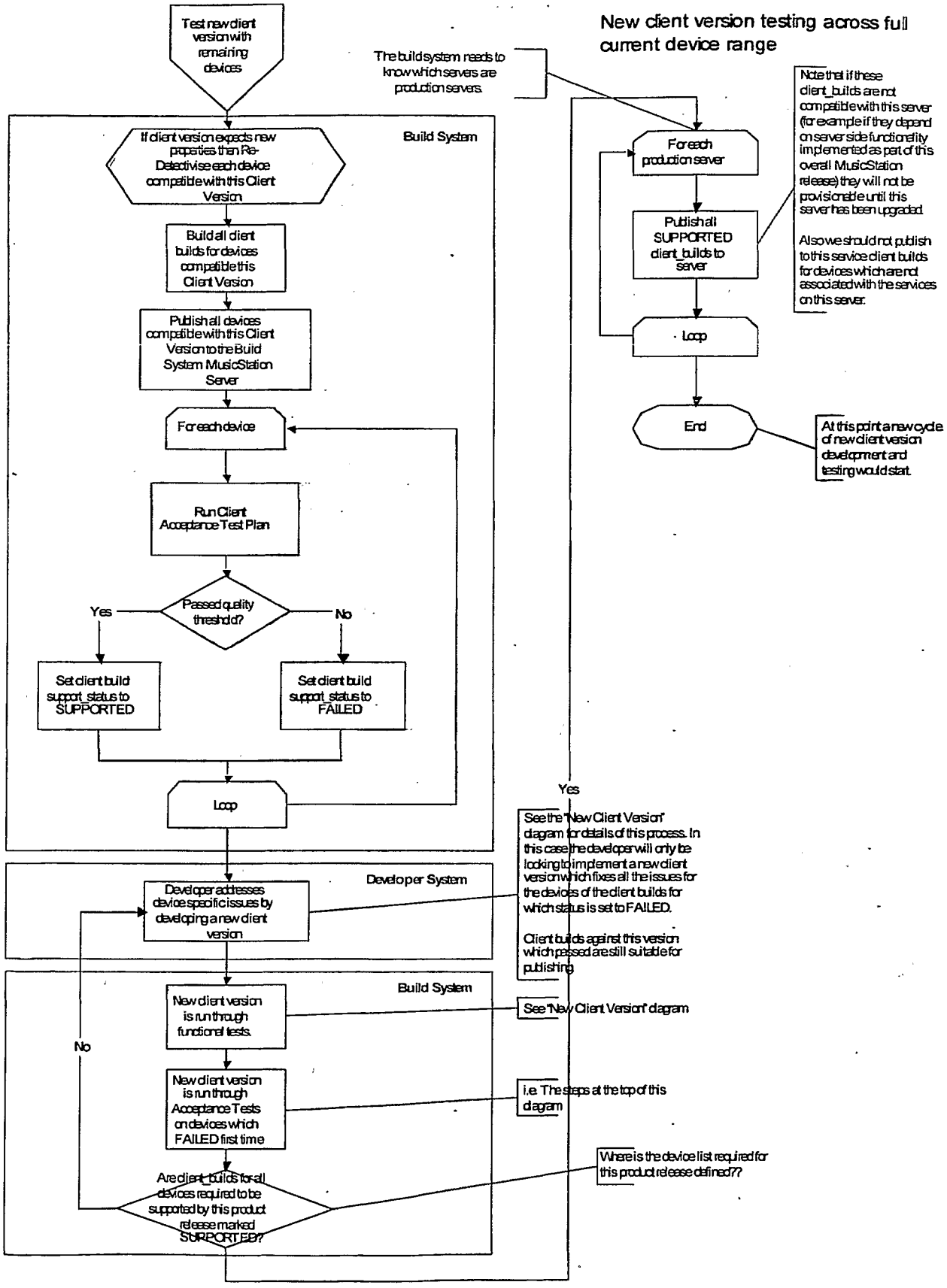
Figure 1 – The process by which a new client version is developed. A new client version is created to either introduce new functionality or to fix either generic or possibly device-specific bugs.

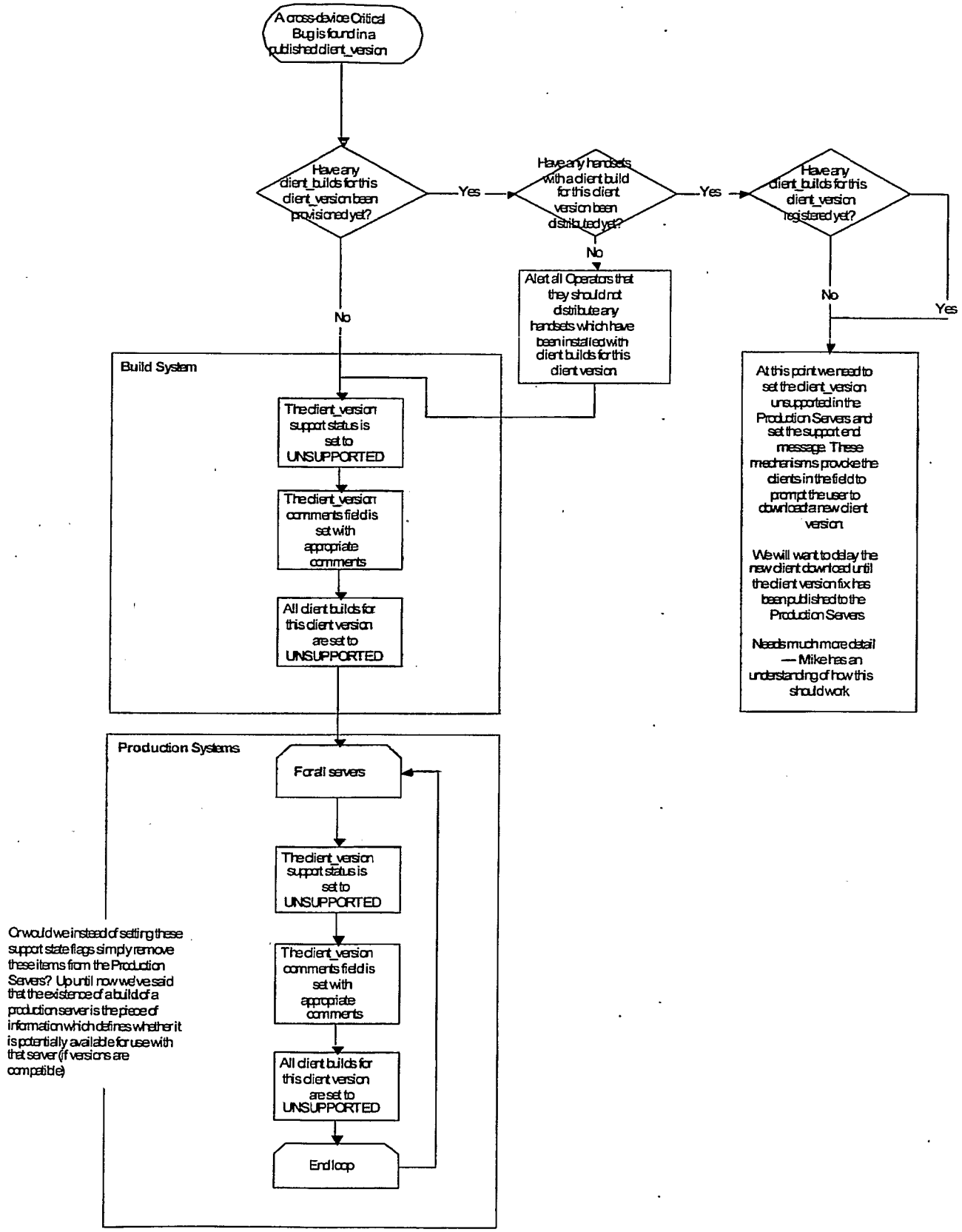
Figure 2 – A new client version is tested across the range of currently supported handsets

Figure 3 – A critical bug is found in a client version which affects all client builds.

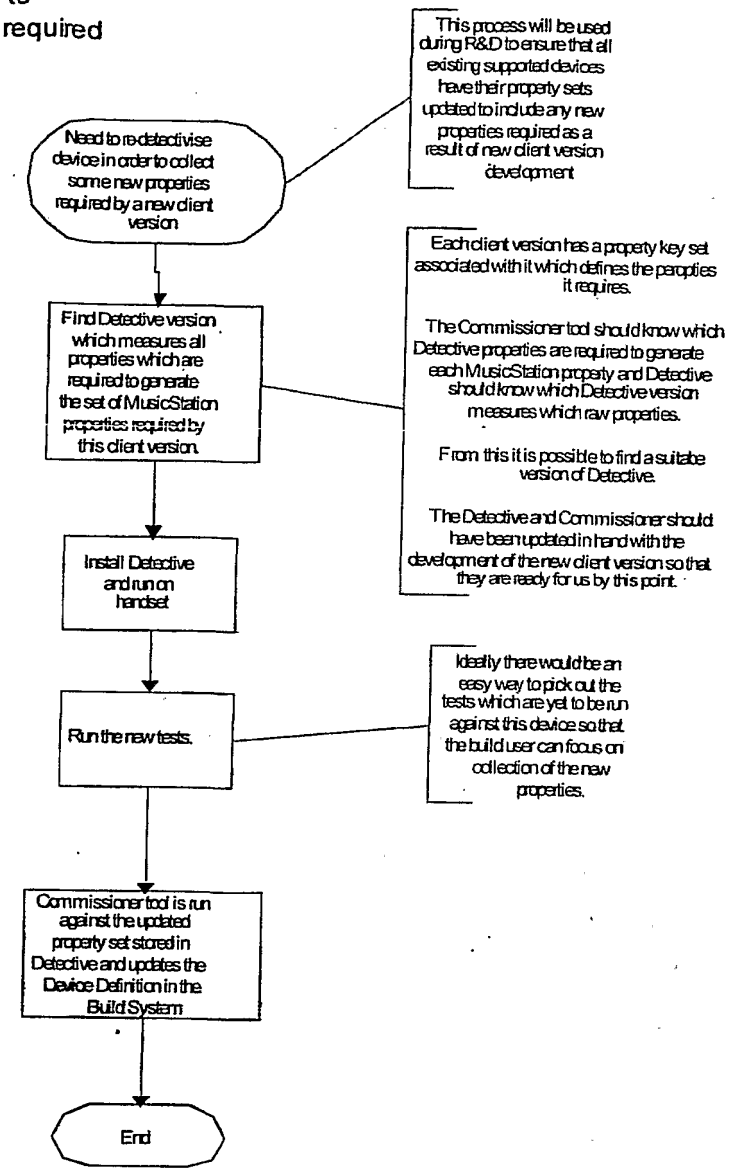
Figure 4 – Re-detectivising a previously Detectivised device in order to gather properties which are required for a new client version.







Re-Detectivise an existing device to gather new Detective properties required for a new client version



Client Builds and Multi-Language Support

Mark Sullivan – 11 Jan 2007

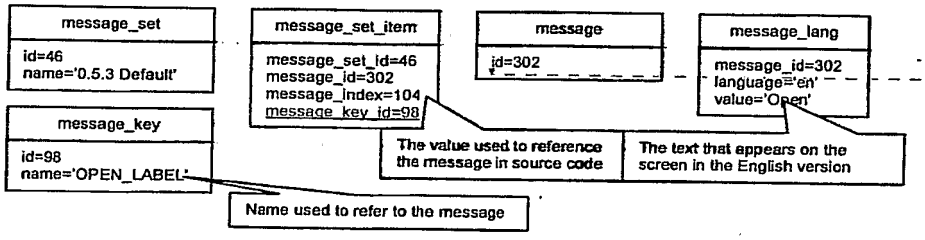
Table of Contents

1. Introduction

This document describes how we manage and use messages to build a client build for a specific device, service and client version.

2. Development

Each client version released by Development has a default set of messages used by the client. In development the default set would contain English messages and possibly messages in a few sample languages required for development testing. This message set is maintained during the development of a release by the developers. Each message in the message set is text or a label that appears somewhere in the client.



spocock 12/1/07 11:04 AM
Comment: Would we expect the default message set for a client version to include multiple languages? Or would the multiple languages only be introduced higher up in the hierarchy?

I think later you are saying that we would only expect English in this message set. Can you clarify that here pls.

For testing purposes our Test Service would have multiple languages.

MSullivan 14/1/07 5:39 AM
Deleted: message_key_id=98

Roland Zwikker 9/2/07 1:25 PM
Deleted: 15-Jan-07 13:57

MSullivan 18/1/07 1:45 PM
Inserted: 15-Jan-07 13:57

Roland Zwikker 9/2/07 1:25 PM
Deleted: 12-Jan-07 12:09

A message is added to the default message set by adding a record to message_set_item with the next available message_index. The message index is used in the source code to access messages in the message set. The index is defined as a constant in the Message object:

```
public static int OPEN_LABEL_INDEX = 104;
```

This constant can then be used to get the message in the currently selected language:

```
openCommand.setLabel(messageSet.getMessage(OPEN_LABEL_INDEX));
```

This message set is set as the default message set for a client version.

Images that are packaged in the build are defined in the default image set. Images are selected from this set based on the image role.

image_set
id=58 name='0.5.3 Default'

image_set_item
image_set_id=58 image_id=34 image_set_item_role='ICON'

image
id=34 source_url=/icons/default.png

The client version is released with the default message and image sets

client_version
default_message_set_id=46 default_image_set_id=58 version='0.5.3'

3. Client Version Release

The following records are packaged with a client version release from development to the Build System:

- The default message set and message set items
- The messages and message keys used by the default message set
- The message langs in English and any other test languages
- The default image set and image set items
- The images used by the default image set

4. Message Translation

Translated messages can be loaded into the Build System at any time. A language is available for selection by the build user when each message in the default message set has a message lang for that language.

message
id=302

message_lang
message_id=302 language='en' value='Open'

message_lang
message_id=302 language='fr' value='Ouvrir'

message_lang
message_id=302 language='it' value='Aprire'

The message is translated into all languages supported by the build system

A translated message can be added at any time

spocock 12/1/07 11:04 AM
Comment: IS there still such a thing as message key set associated with a client version which defines all the messages that it expects? F so can you cover that please.

MSullivan 12/1/07 11:14 AM
Comment: Message key set is no longer relevant, it's the default message set that defines the messages for a build.

MSullivan 14/1/07 8:34 AM
Formatted: Bullets and Numbering

spocock 12/1/07 11:49 AM
Comment: If a service is defined with a set of languages then we could enforce that you are not allowed to add a message set unless you have all the languages covered.

spocock 12/1/07 11:49 AM
Comment: Similarly a device message set would not be selectiable unless it contained message langs for all messages in it matching the full list of service languages

Similarly for service message sets

MSullivan 14/1/07 8:40 AM
Deleted: message_key_id=98

Roland Zwikker 9/2/07 1:25 PM
Deleted: 15-Jan-07 13:57

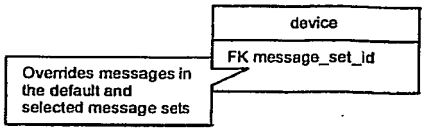
MSullivan 18/1/07 1:45 PM
Inserted: 15-Jan-07 13:57

Roland Zwikker 9/2/07 1:25 PM
Deleted: 12-Jan-07 12:09

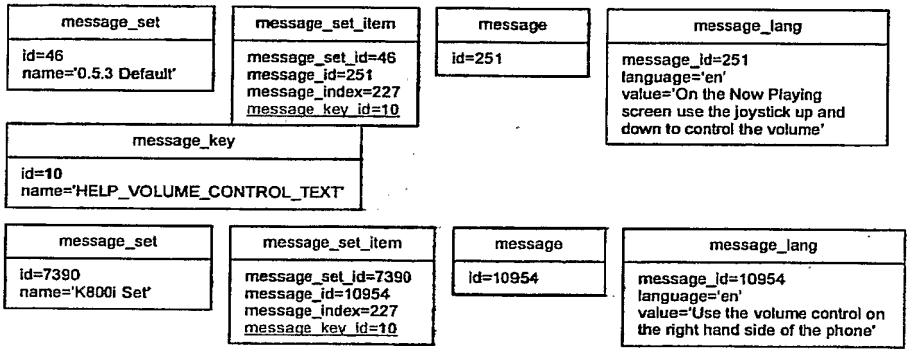
When adding a message to a service message we should enforce that a message lang record exists for all languages supported by the service. Similarly if a build user selects a device to use with that service we should ensure that all device messages have a message lang for al languages supported by the service.
Because a client build is built for multiple languages but we can only include one icon, logo and splash screen in the jar there's no need for translation of images. The images defined for a service should be in the default language for that service. If we wanted to support multiple languages for images we'd need to add an image_lang table.

5. Device Messages

A message set can be defined for a device. This allows us to override messages in the default message set for the selected device.



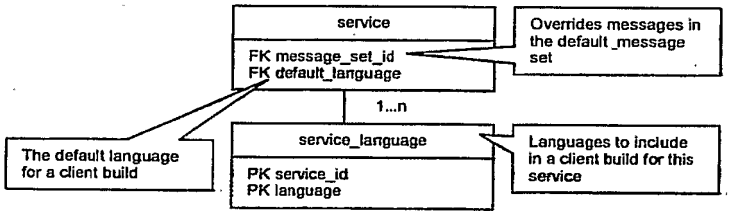
For example a help message may be specific to a particular device:



At time of build, messages defined for the selected device override messages in the default message set that have the same message key.

6. Service Messages

Messages can also be defined for a service. These messages override the default and device message sets although in practice messages should be either device specific or service specific and not both.



Roland Zwikker 9/2/07 1:25 PM
Deleted: 15-Jan-07 13:57
MSullivan 18/1/07 1:45 PM
Inserted: 15-Jan-07 13:57
Roland Zwikker 9/2/07 1:25 PM
Deleted: 12-Jan-07 12:09

A service also has a default language and a set of service languages. These are set as the default language and supported languages for the client build however the build user is able to edit these before doing the build if the build needs a different default language or only a sub-set of the languages.

7. Service Images

MSullivan 12/7/07 8:51 AM
Formatted: Heading 1

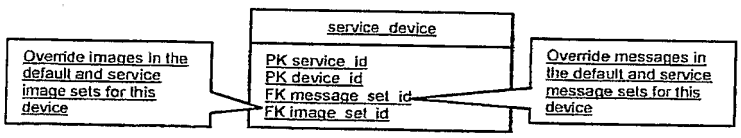
Images can also be defined for a service. These images override those images in the default image set with the same image_set_item_role.

We'd only want to package one splash screen and icon in the jar whereas for messages we intend to package more supported languages. Also messages are over the air updateable whilst images are not. For these reasons there is no image_lang table. If they contain any text then service images will be in the default language for the service.

8. Service And Device Specific Message And Images

MSullivan 12/7/07 1:28 PM
Formatted: Heading 1

In some instances we want to specify that a message or an image is specific to a particular device and a particular service. For example we may want to use an service icon that has been manually resized on a set of devices.



9. Message Substitution

MSullivan 12/7/07 2:17 PM
Formatted: Bullets and Numbering

Any service or device property that can be referenced in the database is available for substitution into the default message set. For example to substitute the customer support phone number:

To get help please call ${service.company.companyAddress.customerSupportTelephone}

The default message set supports substitution and this is hidden from the build user. When they view the default message it will have the phone number already substituted in.

Device and service messages also support substitution. The tools that manage device and service messages should hide the syntax from the build user.

If a substituted value isn't defined for a device or service the build user is required to set the value before the build can proceed.

10. Client Build

MSullivan 12/7/07 2:17 PM
Formatted: Bullets and Numbering

The user has chosen the client version, device and service. The default message set for the version provides the base for the messages selected for the build. These messages are then overridden by the device and service messages sets respectively. These are then overridden by any messages specified in the service_device message set.

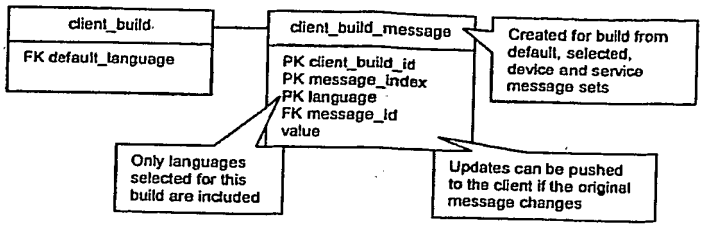
Roland Zwikker 9/2/07 1:25 PM
Deleted: 15-Jan-07 13:57

MSullivan 18/1/07 1:45 PM
Inserted: 15-Jan-07 13:57

Roland Zwikker 9/2/07 1:25 PM
Deleted: 12-Jan-07 12:09

The selected languages for this build are then used to filter the message lang records for the supported languages.

A client build message is created for each message in each language and copied into the client build message table for this build.



Taking a copy of the message at build time allows us to:

- Keep a record of any substitutions made
- Update messages without having to duplicate locked messages

A client build image is created for each image in the default image set and then overridden with any images in the service image set. These are then overridden by any images specified in the service_device image set. These images are then resized and renamed and packaged in the jar. Mike, how is the resizing and renaming done?

Client build messages and images form part of the client build definition and are published to a Production Server when that client build is published to jil.

11. Publishing Client Builds

For each client build the following message related tables are released to the Production System:

- Client_build: The record for this client build
- Client_build_message: The records for this client build
- Message: Each message referenced in client_build_message
- Message_key: The key for each message
- Message_lang: The message_lang for each message in each supported language.
- Client_build_image: The records for this client build
- Source image files: Each image file referenced in client_build_image

MSullivan 4/1/07 8:58 AM
Formatted: Highlight

MSullivan 4/1/07 8:58 AM
Formatted: Highlight

Spocock 12/1/07 11:04 AM
Comment: Would we need to also publish all the other structures discussed above or only a sub-set? We need to get very clear on these things.

MSullivan 12/1/07 2:17 PM
Formatted: Bullets and Numbering

MSullivan 4/1/07 8:52 AM
Formatted: Bullets and Numbering

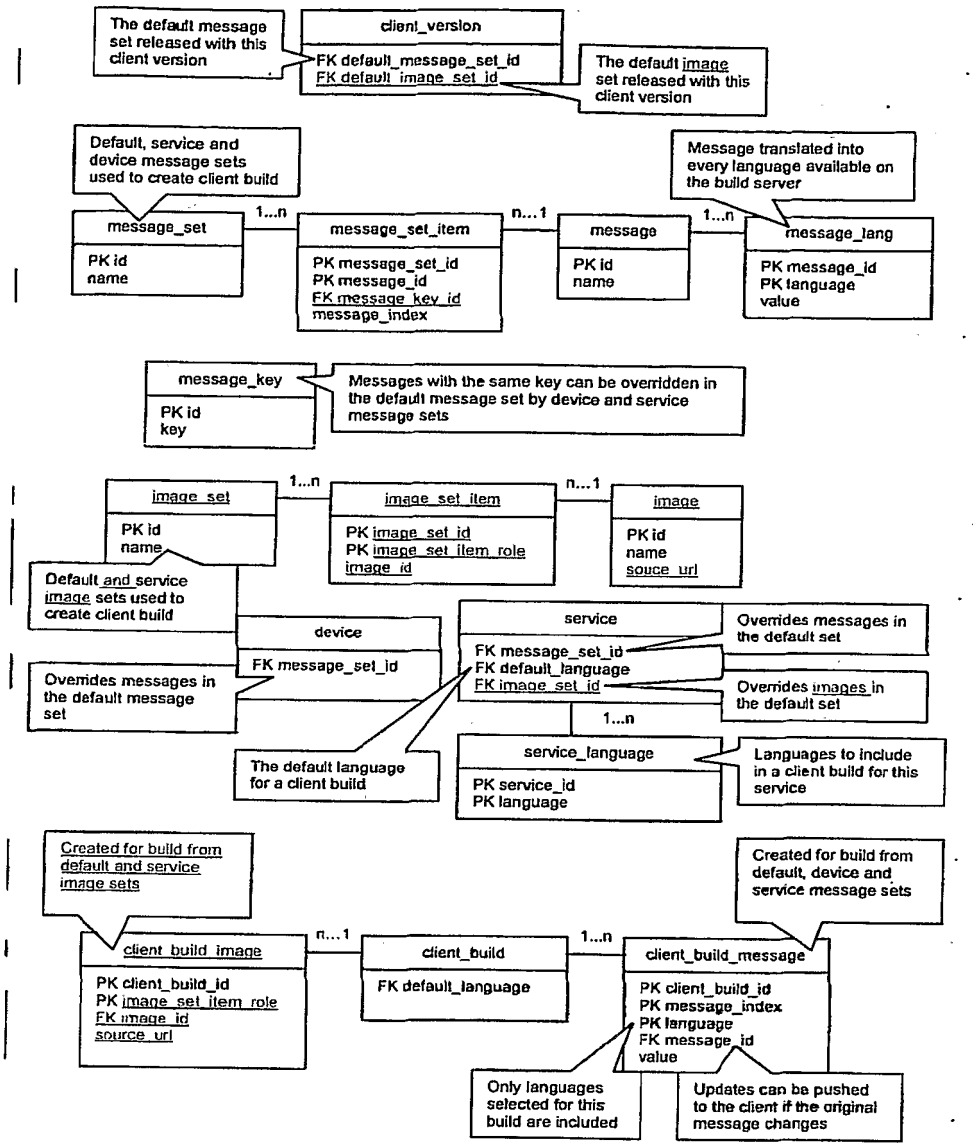
Roland Zwikker 9/2/07 1:25 PM
Deleted: 15-Jan-07 13:57

MSullivan 18/1/07 1:45 PM
Inserted: 15-Jan-07 13:57

Roland Zwikker 9/2/07 1:25 PM
Deleted: 12-Jan-07 12:09

12. Client Build Message Tables

MSullivan 2/1/07 2:17 PM
Formatted: Bullets and Numbering



Roland Zwikker 9/2/07 1:25 PM
Deleted: 15-Jan-07 13:57
MSullivan 8/1/07 1:45 PM
Inserted: 15-Jan-07 13:57
Roland Zwikker 9/2/07 1:25 PM
Deleted: 12-Jan-07 12:09

13. Database Requirements

MSullivan 12/4/07 2:17 PM
Formatted: Bullets and Numbering

client_version

default_message_set_id	FK	number(10)	not NULL

message_key

id	PK	number(10)
name		varchar(96)
description		varchar(256)

MSullivan 14/1/07 8:56 AM
Deleted: message_key_id . . FK . number(10) .

message_set_item

message_set_id	PK	number(10)
message_key_id	PK	number(10)
message_index		

unique index (message_set_id, message_index)
replace name in PK with message_key_id

MSullivan 18/1/07 4:24 PM
Formatted: Font:12 pt, Bold

MSullivan 14/1/07 11:25 AM
Formatted

image_set_item

image_set_item_role	add constraints ('ICON','LOGO')

MSullivan 14/1/07 8:25 AM
Formatted: Font:12 pt, Bold

message_lang

comments	varchar(256)

MSullivan 18/1/07 4:24 PM
Formatted: Font:12 pt, Bold

client_build

default_language	FK	char(2)

client_build_message

client_build_id	PK	number(10)
message_index	PK	number(10)
language	PK	char(2)
message_id	FK	number(10)
is_small_value		number(1)
small_value		varchar(256)
large_value		clob

client_build_image

client_build_id	PK	number(10)	
image_set_item_role	PK	varchar(32)	in ('SPLASH','ICON','LOGO')
image_id	FK	number(10)	
source_url			

MSullivan 18/1/07 4:17 PM
Formatted: Font:12 pt, Bold

service_device

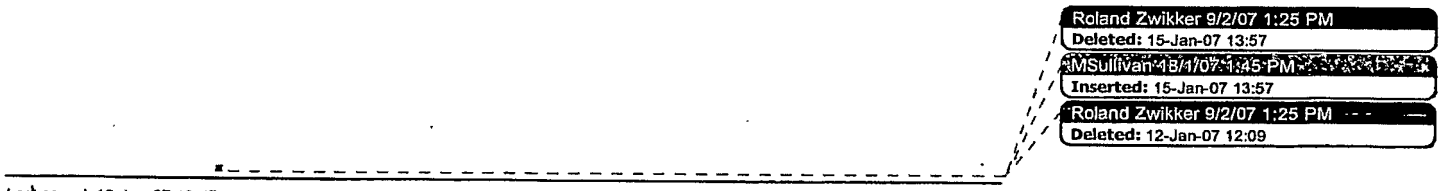
service_id	PK	number(10)
device_id	PK	number(10)
message_set_id	PK	number(10)
image_set_id	PK	number(10)

MSullivan 15/1/07 1:54 PM
Formatted: Font:12 pt, Bold

Roland Zwikker 9/2/07 1:25 PM
Deleted: 15-Jan-07 13:57

MSullivan 18/1/07 1:45 PM
Inserted: 15-Jan-07 13:57

Roland Zwikker 9/2/07 1:25 PM
Deleted: 12-Jan-07 12:09



MusicStation Device Commissioning

Chris Knowles – 17 Jan 2007

Table of Contents

1. Introduction

2. Detective Properties

Each version of the Detective maps to a particular set of properties that is detects.

3. MusicStation Properties

Each client version of MusicStation maps to a particular set of MusicStation properties that it requires (build and runtime).

We will only ever add new MusicStation properties. We will never remove existing MusicStation properties.

4. Detective - MusicStation Property Mappings

The commissioning system contains mappings between Detective and MusicStation properties. There may be multiple mappings for particular properties. E.g. an old version of Detective may detect "height" but a new one may detect "canvas.height" which both may be mapped to the MusicStation property "display.canvas.height". The following rules are obeyed:

- If a Detective property key is renamed, the current mappings are copied and changed accordingly (to keep the old relationships).
- If a Detective property key changes meaning, the current mappings are deleted (and possibly backed up).

- If a Detective property key is added that is required by MusicStation, a mapping is added.
- If a MusicStation property key is renamed, the current mappings are copied and changed accordingly (to keep the old relationships).
- If a MusicStation property key changes meaning, the current mappings are deleted (and possibly backed up).
- If a MusicStation property is added, a mapping is added.

5. Roles

The commissioner will have different user levels (lowest to highest):

- Handset Specialist – client build tester – never able to override.
- Senior Handset Specialist – client build tester with some knowledge of device properties – able to override current values with automated ones.
- Administrator – creator of Detective, Commissioner code or client build approver – able to override current values with any value after automated process had finished.
- Developer – creator of MusicStation client code – always able to override.

These range on the experience the users have with the code for the Detective, Commissioner and MusicStation and the properties passed around the whole system.

6. Commissioning Database

The commissioning database just contains properties that have been manual overridden (including the special file type properties). This can be for several reasons:

- Bug in Detective.
- Something that Detective tries to find out but is never going to be completely accurate.
- Detective does not detect this particular property yet.
- Detective tests have not yet been run.

We should not manual override to work around a MusicStation bug. We can possibly allow this for testing purposes but may want to block it altogether.

Logically, when we override a property during commissioning, we are overriding a MusicStation property (we are overriding after we transform). Therefore, the commissioning database will store MusicStation properties and not Detective properties.

If the user has permissions, they will be able to override at the following points in the process:

- If there is no mapping between a Detective property and MusicStation property.
- If there is no value for a Detective property.
- After the transformation process has produced its outcome (MusicStation may or may not have a current value).

7. Build database

The key parts that the build database contains that are relating to this document are:

- The definition for each device / MusicStation client version combination that has been commissioned.
- The set of property keys required for each MusicStation client version.

The commissioner will write directly to the Build Database. We will not use the data mover mechanism.

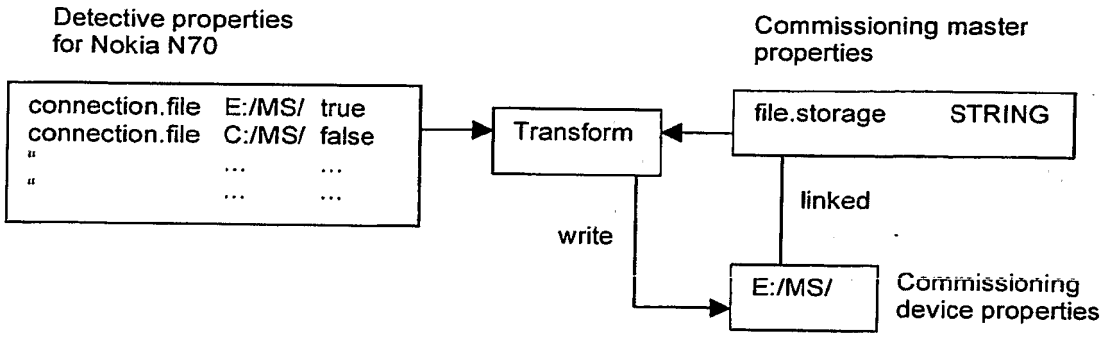
8. Transformation Process

Transformations have to be performed between the Detective data model and the MusicStation device data model.

For example, the Detective finds out the drives names via Java and then collects the following data:

- Can we write to the root of this drive?
- Can we write to known directories such as "music_files" in this drive?

However, the MusicStation master property is simply "where should we be writing to?" The process for working this out is not just a simply copy. In fact, it may require manual intervention by the commissioner due to the fact that we want to be writing to the memory card, but we only have so many known memory card drive names (i.e. we cannot be certain that there will not be a new one). Another example would be canvas height, which can simply be copied.

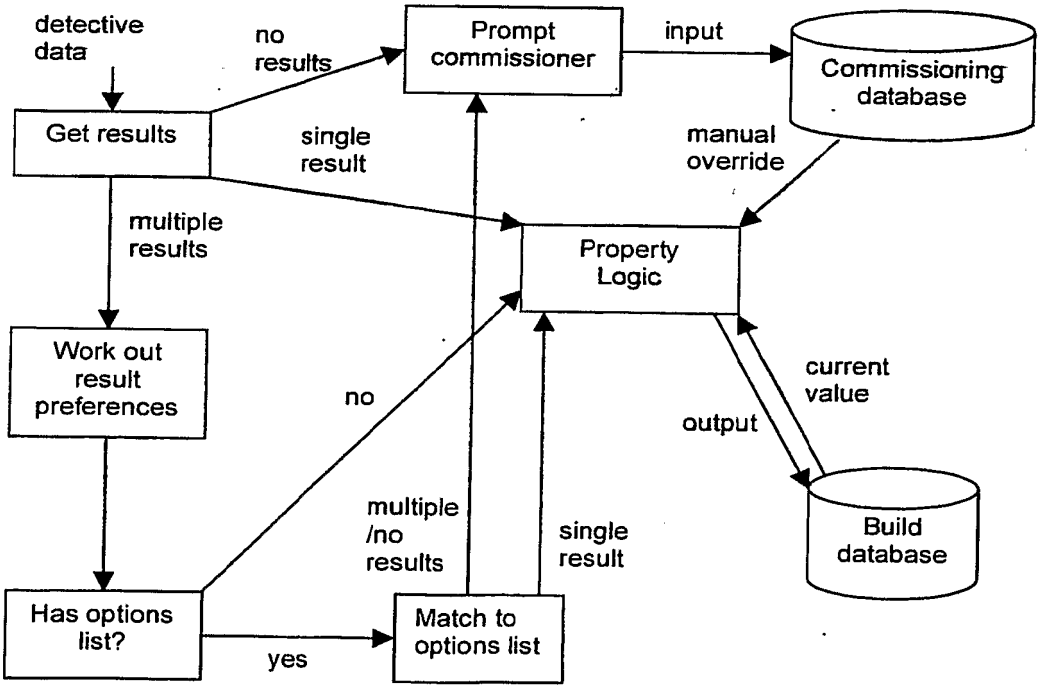


8.1 Transformation Process Implementation

The process is essentially driven by two pieces of information:

- Mappings between Detective properties and MusicStation properties (this is a many to many relationship).
- Known options for particular Detective properties (e.g. for memory card we'd want to pick up "E", "Memory Card", "efs" automatically).

The transformation process for a particular Detective property works as follows:



At the start of the process, we can potential have multiple results for the same Detective property. The following preference order is applied in this order:

1. More recent Detective version
2. Higher certificate domain (e.g. manufacturer domain preferred over third party trusted domain)

3. Higher Detective variant (e.g. variants with extra components request extra permissions)
4. Later date performed

Even after going through these preferences for a set of results, it is still possible to have more than a single result left. There are two follow-ups to this:

1. Automatically look at a list of possible options for this result if it exists
2. Prompt the commissioner to choose from the set of results

If we have a list of options for a particular result, we automatically go through our set of results and match to the list of available options, working out from the priorities of the options. There are two outcomes for this automatic process:

1. No results in the set match the options
2. Single top result found

If we have found a single result then we are ok, but if we have no results then this means we cannot automatically choose and we need to prompt the commissioner to choose between the results in the set given to this automatic process. Each mapping will have a description of what it should be which we will display to the user to aide them in choosing the correct value.

Whenever setting a property, the process will look for any current value in the commissioning database and build database. The following rules then apply:

- If a MusicStation property has a current value in the build database and an overridden value in the commissioning database they should be identical because the build database should contain the overridden value from the last run of the commissioner. If not then something has gone wrong and we should inform development/admin!
- If there are no current values then we are safe to write.
- If there is a current value and it is identical to the outcome of the transformation process we are safe to write (with a new modified timestamp). If the current value is in the commissioning database (a manual overridden one), then we are safe the delete that because the outcome of the transformation process is now the same.
- If there is a current value and it is different to the outcome of the transformation process then we need to highlight this to the user. If the current value is in the commissioning database, we can give a reason why it was overridden last time. We probably only want the administrator and developer roles to be able to choose this new value. If we choose the new value, we will need to delete the current value if it is in the commissioning database.

We will need to have one table that holds the mapping of Detective properties to MusicStation ones, but we also need a generic way to hold a list of options that a Detective property can be (e.g. for memory card we'd want to pick up "E", "Memory Card", "efs", etc). We therefore need another table that links these options to the correct Detective properties.

Some examples of transformation operations are:

- copy – this just copies the Detective property to the corresponding Commissioning one
- result query – this transforms via a query and returns the Detective result (such as E:/MS/)
- content query – this transforms via a query and returns the content linked to the Detective result (such as Sugababes, e-aac+, m4a, 48kbps, 44kHz)

We can perform each query with or without matching to available options. e.g. we can find all connection.file results or all connection.file results that wrote to known places. If a particular property does have options, and the full query returns no results, then we can perform the same query but without matching to available options and provide this set of results to the commissioner to choose from.

Please note, queries below are not exact SQL.

Transformations:

	Example 1	Example 2	Example 3
detective_property	"connection.file"	"display.canvas.height"	"audio.playback"
detective_version	0.57	0.57	0.57
property	"cache.phone.url"	"canvas.height"	"file.format"

operation	"result query"	"copy"	"content query"
query	select detective_property where result is true	*null*	select detective_content where property_value = true and top preference

others operations could be translate, add, etc.

List of options:

	Example 1	Example 2	Example 3
detective_property	"connection.file"	"connection.file"	"connection.file"
value	"E"	"Memory Card"	"efs"
priority	1	2	3

8.2 Property Mappings

The actual Detective to Commissioning mappings used by the transformation process are detailed in this section:

detective_property_transformation

detective_property	property	operation	query
connection.file	file.storage	query-result	select r.value from detective_results r where r.result = 'true'
display.canvas.height	canvas.height	copy	*null*
display.canvas.width	canvas.width	copy	*null*
audio.playback	file.container	query-content	select c.container from detective_results r, detective_content c where r.result = 'true' and c *is top preference*
audio.playback	file.format	query-content	select c.format from detective_results r, detective_content c where r.result = 'true' and c *is top preference*
audio.playback	file.rate.bit	query-content	select c.bit_rate from detective_results r, detective_content c where r.result = 'true' and c *is top preference*
audio.playback	file.rate.sample	query-content	select c.sample_rate from detective_results r, detective_content c where r.result = 'true' and c *is top preference*

detective_property_options

detective_property	value	priority
connection.file	E	1
connection.file	Memory card	2
connection.file	e	3
connection.file	Ms	4
connection.file	Nand	5
connection.file	SD	6

9. Overall Publish Device Process

The inputs to this process are device (selected from the Detective database) and MusicStation client version range. The default "from" version will be the oldest client version. This allows use to ensure that we are being delivered properties for all client versions which are currently marked supported and are available for distribution to operators.

We then do the following:

1) Check the set of MusicStation properties required by this client version of MusicStation.

2) Check we have mappings for all these MusicStation properties from Detective properties.
    a. If we do, continue.
    b. If we don't, prompt the user with the mappings that are missing.
3) Check we have Detective property values for all these mappings.
    a. If we do, continue.
    b. If we don't, and there are Detective tests that produce these Detective properties prompt the user to run these tests (or possibly override if allowed – see section 5).
    c. If we don't, and there aren't Detective tests that produce all these Detective properties:
        i. Tell the user that they will have to wait! (Unless they are allowed to override).
        ii. Prompt the user to run any Detective test that produces some of these Detective properties (to complete the data a bit more).
4) Run the transformation process (write to build database).
5) Run the build process from the build database.

10. Other Publishing Processes

10.1 Adding a new device to the build system

1) Check we have the manufacturer in the build database and update accordingly.
2) Add device record (display name, user_agent, guid).
3) Add file types for this device (a special property mapping).
4) Add device property set values.
5) Add device image set values.
6) Add device message set values.

10.2 Modifying a device in the build system

1) Find device guid in build system and modify name/manufacturer id.
2) Update file types for this device.
3) Update device property values.
7) Update device image set values.
4) Update device message set values.

10.3 Finding devices which need tests to be run

This follows the same logic as in the first 3 steps of section 0, apart from the user selects by client version and the information is shown for all devices instead of just working on a single device.

10.4 Override a property

Depending on user permissions, we may be allowed to override MusicStation properties during commissioning. The commissioning database stores these overridden MusicStation properties. There are three situations for overriding:

- We don't have a mapping for a MusicStation property.
- We have a mapping for a MusicStation property but don't have the correct Detective property.
- We have a mapping for a MusicStation property and have the correct Detective property.

5) In every case if we are overriding, we write the MusicStation property to the commissioning database (not the Detective property).

11. Developer Processes

11.1 Adding a new client version

1) Copy the property set for the last client version.
2) Add to client version table (linking to this new property set).
3) Add/remove properties as required.

4) After this the commissioner still follows the same logic (section 0).

11.2 Adding a MusicStation property

New MusicStation properties are added into the build system. The Overall Device Publish Process (section 0) can then be run again for a device and the same rules apply.

Building and Provisioning MusicStation Clients

Mike Lamb – 22 Dec 2006

Table of Contents

1. Introduction

This document details the design for the creation and management of the MusicStation installation files. For the Java client, this is the JAD and the JAR, and for the Symbian client, the SIS file.

Section 2 lists the database tables required.
Section 3 lists the processes required.

2. Tables

All the following tables include the standard fields listed below. These fields have been omitted from the table descriptions for clarity.

data_classification
created
inserted
modified

2.1 Client build

The client build table is shown in full. Any fields that differ from the 0.26 data model are shown.

client_build		
Field	**Type**	**Comments**
id	int	PK
service_id	int	FK service.id
device_id	int	FK device.id
client_version_id	int	FK client_version.id
guid	guid	NOT NULL
message_set_id	int	FK message_set.id
property_set_id	int	FK property_set.id
image_set_id	int	FK image_set.id
omnifone_supported_status	varchar(16)	Constraints "SUPPORTED", "TESTING", "DESUPPORTED", "DEVELOPMENT"
service_support_status	varchar(16)	Constraints "SUPPORTED", "TESTING", "DESUPPORTED"
support_start_date	timestamp	
support_end_date	timestamp	
support_end_notice_date	timestamp	This is a renaming of the current field is named "desupport_notice_date"
support_end_message_set_id	int	FK message_set.id This combines the current fields support_end_message_set_id and desupport_message_set_id
guid	varchar(32)	
build_key	varchar(512)	The current field is only a varchar(4)
build_key_algorithm	varchar(4)	The current field is

		unnecessarily a varchar(512)
descriptor_file	varchar(1024)	This is a renaming of the current field "jad_file"
file	varchar(1024)	This is a renaming of the current field "jar_file"

Notes:
The "certificate" field has been removed from this table. Certificate is now just a property.

2.2 Client version

The client version table is shown in full. Any fields that differ from the 0.26 data model are shown.

client_version		
Field	**Type**	**Comments**
id	int	PK
guid	guid	NOT NULL
branch_name	varchar(64)	This is the name of the branch in source control, or the special case of "TRUNK"
revision_number	int	This is the source control revision number.
version	varchar(16)	This is the version number as shown to the user. (e.g. 0.5.2)
message_set_id	int	FK message_set.id
property_set_id	int	FK property_set.id
image_set_id	int	FK image_set.id
event_type_set_id	int	FK event_type_set.id
exception_set_id	Int	FK exception_set.id
support_state	varchar(16)	Is thisversion enabled and/or supported.
support_start_date	timestamp	
support_end_date	timestamp	
support_end_message_set_id	int	FK message_set.id This combines the current fields support_end_message_set_id and desupport_message_set_id
comments	varchar(256)	

2.3 Message

The message table gains one new field.

message		
Field	**Type**	**Comments**
locked	bool	Whether this message is editable or not.

Message has the 2 fields "name" and "comments" replace by

message_key_id	Int	FK message_key.id

2.4 Image

The image table gains one new field.

image		
Field	**Type**	**Comments**
locked	bool	Whether this image is editable or not.

Message has the 2 fields "name" and "description" replace by

image_key_id	Int	FK image_key.id

2.5 Property

The property table is a new table. It is very similar to the message table in that it stores name/value pairs.

property		
Field	**Type**	**Comments**
id	Int	PK
locked	bool	Whether this property is editable or not.
guid	guid	
value	varchar(1024)	

2.6 Message set

The message set table gains one new field.

message_set		
Field	**Type**	**Comments**
locked	bool	Whether this message is editable or not.

2.7 Image set

The image set table gains two new fields.

image_set		
Field	**Type**	**Comments**
locked	bool	Whether this message is editable or not.

2.8 Property Set

The property set table is a new table. It is very similar to the message set table in that it models a collection of properties.

property_set		
Field	**Type**	**Comments**
id	int	PK
enabled	bool	
locked	bool	Whether this property_set is editable or not.
guid	guid	
name	varchar(96)	
description	varchar(256)	
comments	varchar(256)	

2.9 Property Set item

The property set item table is a new table. It is very similar to the message set item table in that it maps a property set to the properties contained in it.

property_set		
Field	**Type**	**Comments**
property_set_id	int	PK and FK property_set.id
property_key_id	Int	PK and FK property_key.id
property_id	int	FK property.id

2.10 Client Version Set

The client version set table is a new table. It is very similar to the other _set tables in that it models a collection of client versions.

client_version_set		
Field	**Type**	**Comments**
id	int	PK
enabled	bool	
guid	guid	
name	varchar(96)	
description	varchar(256)	
comments	varchar(256)	

2.11 Client Version Set item

The client version set item table is a new table. It is very similar to the other _set_item tables in that it maps a client version set to the client versions contained in it.

client_version_set_item		
Field	**Type**	**Comments**
client_version_set_id	int	FK client_version_set.id
client_version_id	int	FK client_version.id

2.12 Device

The device table gains two new fields.

device		
Field	**Type**	**Comments**
property_set_id	int	FK property_set.id
client_version_set_id	int	FK client_version_set.id

Notes:
The "tested" and "tested_date" fields should be removed from device. A "device" is not tested, it is a client_build which is tested.
Also "enabled" should be removed from device. A "device" is not enabled, it is a build which is enabled for a device.

2.13 Service

The service table gains three new fields.

Service		
Field	**Type**	**Comments**
property_set_id	int	FK property_set.id
client_version_set_id	int	FK client_version_set.id
device_set_id	int	FK device_set.id

2.14 Server version

The server version table is currently named "version". It should be renamed to avoid any potential confusion with the client_version table. The server version table gains one new field

server_version		
Field	**Type**	**Comments**
client_version_set_id	int	FK client_version_set.id

Notes:
The server version table contains several fields which may not be used, and we may want to take this opportunity to remove these fields.

"version_major", "version_minor", and "version_micro" is a very explicit way of modelling a label that would be shown on start up of the server.
The "version_revision" number should be changed to be named the same as the client_version table. i.e. "branch_name" and "revision_number" are used in the client_version table.

2.15 Device Set

The device set table is a new table. It is very similar to the other _set tables in that it models a collection of devices.

device_set		
Field	**Type**	**Comments**
id	int	PK
enabled	bool	
guid	guid	
name	varchar(96)	
description	varchar(256)	
comments	varchar(256)	

2.16 Device Set item

The client version set item table is a new table. It is very similar to the other _set_item tables in that it maps a client version set to the client versions contained in it.

device_set_item		
Field	**Type**	**Comments**
device_set_id	int	FK device_set.id
device_id	int	FK device.id

2.17 Message Key

The message key table is a new table.

message_key		
Field	**Type**	**Comments**
id	int	PK

name	varchar(96)	
description	varchar(256)	

2.18 Property Key

The property key table is a new table.

property_key		
Field	**Type**	**Comments**
id	int	PK
name	varchar(96)	
description	varchar(256)	
type	varchar(16)	Constrained to "MIDP_BUILD", "MIDP_RUNTIME".
value_regexp	varchar(1024)	A regular expression which can check the values given to this property

2.19 Image Key

The image key table is a new table.

image_key		
Field	**Type**	**Comments**
id	int	PK
name	varchar(96)	
description	varchar(256)	

2.20 Message Set Item

Message set item has the name field replaced by

image_key		
Field	**Type**	**Comments**
message_key_id	int	FK message_key.id

3. Processes

Some assumptions are made for these processes

1) That a method exists to move data and between servers which have the same database schema.
2) That a default test service exists.

3.1 Marking a device as useable with a client_version

When a new device or client_version has been added to the system, it is likely that this process will be required.

1) The device record should be identified.
2) The client_version record should be identified.
3) The device.client_version_set_id property should be used to identify the client_version_set.
4) A new client_version_set_item record should be created that links the client_version_set to the identified client_version.

3.2 Marking a device as useable with a service

When a new device or service has been added to the system, it is likely that this process will be required.

1) The device record should be identified.
2) The service record should be identified.
3) The service.device_set_id property should be used to identify the device_set.
4) A new device_set_item record should be created that links the device_set to the identified device.

3.3 Marking a service as useable with a client_version

When a new service or client_version has been added to the system, it is likely that this process will be required.

1) The service record should be identified.
2) The client_version record should be identified.
3) The service.client_version_set_id property should be used to identify the client_version_set.
4) A new client_version_set_item record should be created that links the client_version_set to the identified client_version.

3.4 Marking a server_version as useable with a client_version

When a new server_version or client_version has been added to the system, it is likely that this process will be required.

1) The server_version record should be identified.
2) The client_version record should be identified.
3) The server_version.client_version_set_id property should be used to identify the client_version_set.
4) A new client_version_set_item record should be created that links the client_version_set to the identified client_version.

3.5 Adding a new device

A device will be created by the Detective process.

From the server where the device was created, the following data needs to be moved to the build server

Device record
Manufacturer record (optional)
Device_file_type records
The Message_set from device.message_set_id (and all related message_set_items and messages)
The Image_set from device.image_set_id (and all related image_set_items and images)
The Property_set from device.property_set_id (and all related property_set_items and properties)

It is likely that the "Marking a device as useable with a client version" process would then be used to mark the most recent client_version and useable with this new device.

Also the "Marking a device as useable with a service" process would be used to mark the default test service as useable with this new device.

3.6 Adding a new service

A service will be created by the service creation tool.

From the server where the service was created, the following data needs to be moved to the build server. The service may also have related content data, but this is not required on the build server.

Service record
The Message_set from service.message_set_id (and all related message_set_items and messages)
The Image_set from service.image_set_id (and all related image_set_items and images)
The Property_set from service.property_set_id (and all related property_set_items and properties)

It is likely that the "Marking a device as useable with a service" and "Marking a service as useable with a client_version" processes would then be used to mark the most recent client_version and a device as useable with this service.

3.7 Adding a new client_version

A client version will be created by the R&D team.

From the server where the client_version was created, the following data needs to be moved to the build server.

Client version record
The Message_set from client_version.message_set_id (and all related message_set_items and messages)
The Image_set from client_version.image_set_id (and all related image_set_items and images)
The Property_set from client_version.property_set_id (and all related property_set_items and properties)

The event_type_set from the client_version.event_type_set_id (and all related event_type_set_items and event_types)
The exception_set for the client_version.exception_set_id (and all related exception_set_items and exceptions)
The Message_key_set from client_version.message_key_set_id (and all related message_key_set_items and message_keys)
The Image_key_set from client_version.image_key_set_id (and all related image_key_set_items and image_keys)
The Image_key_set from client_version.image_key_set_id (and all related image_key_set_items and image_keys)

It is likely that the processes "Marking a device as useable with a client_version", "Marking a service as useable with a client_version" and "Marking a server_version as useable with a client_version" would then be used to mark this client_version as useable with a device, the default_service and the current server_version.

3.8 Creating a new build

All the data required for this new build should exist before this process is started. If the new build is required for a service, device or client_version that do not exist in the build server database, then the appropriate process should be run to create this data first.

3.8.1 Identifying the service, device and client_version records to use

1) The "build user" is presented with a list of services where the service.enabled flag is true.
2) The service record to use is selected by the "build user".
3) The service.device_set_id field is used along with the device.enabled flag to find the buildable devices for this service.
4) If zero devices are found, the build process should exit.
5) If only one device is found, the build process should make the choice for the "build user".
6) If more than one device is found then the "build user" should be presented with a list of devices. The "build user" may choose one device to build. To aid with the choice the "build user" should be shown any relevant information from device.
7) The service.client_version_set_id and the device.client_version_set_id data is used to identify 2 client_version sets. An intersection of these client_version sets are the potential client_versions that can be build for this service/device combination. The client_version.enabled flag is used to filter this list further.
8) If zero client_versions are found, the build process should exit.
9) If only one client_version is found, the build process should make the choice for the "build user"
10) If more than one client_version is found then the "build user" should be presented with a list of client_versions. The user may choose one client_version to build. The "build user" may choose one client_version to build. To aid with the choice the "build user" should be shown any relevant information from client_version (for example, version, revision_number, and branch name).

3.8.2 Preparing the Message, Property and Image data.

11) The build system should use the client_version.message_set_id to identify the base set of messages that will be used with this build. This becomes the working message set.
12) The build system should then override any messages in the working message_set with messages of the same name that are present in the device.message_set_id.

13) The build system should then override any messages in the working message_set with messages of the same name that are present in the service.message_set_id.
14) Any device or service specific substitutions should be done at this point. After the substitutions have been done, the build system should see whether these messages (with substituted text) already exist in the database as "locked" items. If a message does exist, then the working message_set should use the existing message, otherwise the message should be created in the database.
15) The client_revision.message_key_set_id is used to identify the set of message keys that must exist for this client revision. The working message_set is then compared against this message_key set to make sure that all the messages required by the client_revision are present in this working message_set.
16) The working message_set is then looked for in the database. If the same data as the working message_set exists in the database, then the message_set from the database will be used. If the working message_set does not exist in the database, then the message_set should be created.
17) If more than one message_set exists in the database, then the "build user" should be given the choice of which language message_sets to use. More than one message_set
18) The same process as used for identifying/creating the message_set should be used for identifying/creating the property_set and the image_set, apart from the language choice in step 17) is not required.

3.8.3 Doing the build

19) If the build system hasn't already checked out the client build sandbox at the given client_version.branch_name and client_version.revision_number, then the sandbox should be checked out at this point.
20) The ant task that puts the messages, images, properties, event_types and exceptions into the correct format for this build should then be run. This ant task will have been checked out along with the source code, and the ant task's functionality may change over tim, but it will always be invoked in the same way. Given the message_set_id, image_set_id, property_set_id, event_type_set_id and exception_set_id, the ant task will produce the input as required by the current MIDlet source code. This input may be data object files, property files or even generated source.
21) The ant task to create the netbeans configuration files is invoked.
22) The actual build is done (i.e. code pre-processing, compilation, obfuscation, packaging and signing)
23) A new client_build record is created and inserted in to the build server's database.

3.9 Critical bug found for a client version

This only considers the implication on the build system. If we find a critical bug on a released client_version then a more in depth process is required to handle the upgrade of deployed clients.

1) The client_version record is identified
2) The client_version.enabled field is changed to false.
3) The client_version.supported field is changed to false.
4) The client_version.comments field is updated with appropriate comments.
5) Any client_build records that use this client_version have the omnifone_supported_status updated to "DESUPPORTED"

3.10 Critical bug found for a device / client version combination

This only considers the implication on the build system. If we find a critical bug on a released device / client_version combination then a more in depth process is required to handle the upgrade of deployed clients.

1) The device record is identified
2) The device.client_version_set_id field is used to find the client_version_set.
3) The client_version_set has the client_version removed from it.
4) Any client_build records that use this device_id and client_version_id have the omnifone_supported_status updated to "DESUPPORTED"

3.11 Server release process

In addition to the actual deployment of the server software, the following needs to be done

1) A new record created in server_version
2) The "Marking a server version as compatible with a client version" process should be used for each client_version that this server version is compatible with.

3.12 Client build support state check

A client build supported state can be checked by checking the following fields.

- client_build.omnifone_supported_status
- client_build.support_start_date
- client_build.support_end_date
- client_build.client_version.enabled
- client_build.client_version.supported
- client_build.service.enabled

3.13 Publishing process

1) A client_build (or collection of client_builds) are chosen to be released.
2) The "client build support state check" process is used to make sure the client_build is publishable.
3) The client_build record and all related data is moved to the target server's database.

3.14 Provisioning process

1) A client build is chosen
2) A check is done to see whether this client_build is compatible with the current server. (i.e. is client_build.client_revision in the current server_version.client_revision_set)
3) The "client build support state check" process is used to make sure the client_build can be provisioned.
4) The server modifies the JAD file to point to itself at the MusicStation host.

MusicStation (Java Edition) Handset Compatibility Rules

Steve Pocock – v3 – Jan 2007

Table of Contents

Changes

V4 – Typo corrections

1. Introduction

This document defines the tests we perform in determining the compatibility of a handset with MusicStation based on analysis of Detective results data. These rules are used in generating both our standard **Handset Summary Report** for each device, and **Handset Compatibility Report** which lists current compatibility status of every handset known to us.

For a detailed description of many of the capabilities referred to in this document, see the **Handset Capabilities Checklist** document.

2. Capabilities

A capability is some attribute or behaviour of a particular device. This might be associated with a single property of the device or some combination of them. Capabilities can have a number of different levels of significance:

- **Critical** – This capability is required for MusicStation to run acceptably on this handset
- **Expected** – This capability is expected and will be used by MusicStation if present, but its absence does not preclude MusicStation support

When a capability is measured for a particular handset using Detective it will compared with criteria which define whether it meets MusicStation's requirements. The result of this comparison for a particular capability will be in one of four states:

Detective Pass	The device passes the Full Pass criteria defined for the capability. (Properties for the device have been gathered directly by Detective).
Detective Acceptable Pass	The device passes the Acceptable Pass criteria defined for the capability but not the Full Pass critieria. (Properties for the device have been gathered directly by

	Detective).
Detective Unknown	Insufficient information has so far been gathered by Detective in order to determine the status of this particular capability.
[redacted]	Sufficient information has been gathered but the device fails the Acceptable Pass critiera. (Properties for the device have been gathered directly by Detective).

Before we get our hands on a handset we can make an assessment of predicted support yb examining the specification of the handset. These criteria are less strict and make some assumptions in order for us to make an assessment of the likely supportability of a particular handset before it is Detectivised.

Specification Likely Pass	Based on its published specification information the device meets the Specification Likely Pass criteria. A specification pass means that we consider it likely that when we process the device it will pass the Full Pass criteria for this capability.
Specification Unknown	No specification information is available or insufficient information available to make an assessment.
[redacted]	Based on its published specification information the device does not meet the Specification Likely Pass criteria.

3. Handset Compatibility

A handset can have compatibility measured at different levels of reliability:

- Detective – The handset has been in our hands and has been verified ourselves. The compatibility rating based on this data is 100% reliable.
- Specification – The handset has been evaluated only through its specification
- Group – The handset is related to other handsets which have passed

Before a handset is made available to us then a handset can have its compatibility assessed from the Specification and Group data we gather. When it has been brought in house then it can be assessed based on Detective data.

The overall compatibility rating of a handset is derived from its individual criteria. For a handset which has been processed through Detective and at least some of the properties have been collected then its compatibility rating will be one of the following.

Detective Pass	All Critical criteria are marked Pass
Detective Acceptable Pass	All Critical criteria are marked either Pass or Acceptable Pass
Detective Unknown	One of the Critical criteria are yet to be determined.
[redacted]	One or more of the Critical criteria are marked Fail

For a handset which is yet to be Detectivised but has had its specification assessed it can be in one of the following compatibility states:

Specification Likely Pass	All of the Critical criteria are marked Specification Likely Pass
Specification Unknown	One or more of the Critical criteria are yet to have specification information located.
[redacted]	One or more of the Critical criteria are marked Specification Likely Fail

For a handset which has simply been placed into a group with existing handsets the compatibility states can be:

Group Reliable Pass	This handset is in a group which generally pass. The group is from a handset manufacturer who implements consistent platforms. See section 7 for full details.
Group Likely Pass	This handset is in a group which generally pass. The group is from a less reliable manufacturer. See section 7 for full details.
Group Unknown	Group is unknown
	The handset is a member of a group which generally fail.

4. Handset Compatibility Reports

It is important to note that the above compatibilities are for internal use to provide a clear picture of each the status of each handset. Internal Handset Compatibility Reports would show a list of handses and their three compatibility states:

Device	Detective	Specification	Group
Nokia 1234	Pass	Likely Pass	Reliable Pass
Sony Ericsson K677	Pass	Likely Pass	Group Unknown
Samsung D455	Fail	Likely Fail	Group Unknown

These compatibities may be consolidated for external presentation.

External reports displaying the compatibility of a handset for an external audience can interpret these ratings as they deem appropriate. For example, if a report is required which lists as many handsets passing as possible, a Group Reliable Pass might be reported as a Pass alongside all handsets which passed the full Detective tests to increase the numbers of handsets which pass.

5. Critical Capabilities

The following capabilities are critical for a MusicStation implementation on a particular handset. A handset is not compatible with MusicStation unless it has all these capabilities marked as Acceptable Pass or Full Pass.

5.1 Core Capabilities

Capability	Description	Criteria to meet for a Specification Likely Pass	Criteria to meet for an Detective Acceptable Pass	Criteria to meet for a Detective Full Pass
MIDP v2.0 (with CLDC v1.0 or v1.1)	Provides the basic Java platform on which MusicStation is implemented.	Specification states support for MIDP v2.0	None	MIDP v2.0 library is present This is confirmed by checking whether the system property "system.properties.microedition.profiles" contain "MIDP-2.0".
MM API	The Mobile Media API (JSR-135) provides the basic functionality that MusicStation requires for music playback.	Specification states support for MMAPI (JSR-135)	None	MM API is present. This is confirmed by the system property "system.properties.microedition.media.version" contain "1.1" or "1.0" OR the object javax.microedition.media.Player exists
File Connection	The file connection part of FileConnection and PIM API (JSR-75) is used by MusicStation for reading and writing files to the phone and memory card.	Specification states support for FCAPI (JSR-75)	None	File Connection API is present. This is confirmed by checking whether the the system property "system.properties.microedition.io.file.FileConnection.version" contains anything OR the object "javax.microedition.io.file.FileConnection" exists

Capability	Description	Criteria to meet for a Specification Likely Pass	Criteria to meet for an Detective Acceptable Pass	Criteria to meet for a Detective Full Pass
Music Playback Support	Handset supports playback of a full quality music through MMAPI. The playback is of acceptable audio quality.	(1) Specification states MMAPI support for eAAC+ or AAC (2) Specification states that handset native player supports eAAC+ or AAC	The music file format which plays back with appropriate quality is: AAC 64 kbits/s @ 44.1Khz (Or less preferably):[1] AAC+ 48 kbits/s @ 44.1Khz eAAC+ 32 kbits/s @ 44.1Khz AAC+ 32 kbits/s @ 44.1Khz The format must playback when encoded in the m4a container.	The music file format which plays back with appropriate quality is: eAAC+ 48 kbits/s @ 44.1Khz The format must play-back when encoded in the m4a container.
Handset or subscriber identification	Access to system properties which provide access to unique device identification information.	Assume this will be accessible if specification states MIDP v2.0 support	None	Access to either IMEI or Bluetooth ID of the handset via a system property.
User Interface Navigation	Handset provides access to the MIDlet for all keys/buttons required by the interface. The list of keys/buttons depends on whether the handset interface is keyboard or touch-screen based.	Assume this will be accessible if specification states MIDP v2.0 support	None	Keyboard handset list of keys: Up, Right, Down, Left, Fire, Left Soft Key, Right Soft Key Touch-screen handset list of keys: *None required*
Memory Card Support[2]	The device supports a memory card. A removable memory card is specified as mandatory as part of the music pre-load process.	Specification states support for a removable memory card	None	The device supports a user-replaceable memory card.
Memory Capacity[3]	The memory space that is	Maximum size of the	The handset supports a	The handset supports a memory

[1] Our current expectation is that AAC support is going to be wider than AAC+ support and since we want to minimise the number of music formats our audio supplier must provide, we believe we have . AAC+ 48kbits/s will be slightly smaller than AAC 64kbits/s but less compatible across handsets. We may remove some of these criteria when the final supported formats are agreed with the audio supplier.

[2] During Detectivisation we assess the memory card that is provided with the handset. It is our expectation that when deploying a MusicStation handset an operator will typically install a larger memory card than the usual minimal memory card provided by default with the handset.

[3] Currently we will use the memory card for storage since that eases the pre-load process. We will use the handset internal memory only on handsets which do not support a memory card (e.g. Nokia N91).

Capability	Description	Criteria to meet for a Specification Likely Pass	Criteria to meet for an Detective Acceptable Pass	Criteria to meet for a Detective Full Pass
	available for storing music on the handset, with a preference for storing the music on the memory card. We only consider the handset memory when the handset does not support a memory card. The available memory limits the number of tracks which the user can have available at any one time to listen to while disconnected.	specified format of removable memory card would support 30 tracks of 1.5MB OR if the handset does not have a memory card then the internal memory is sufficient for 30 1.5MB tracks.	memory card which can store upto 30 typically sized downloaded tracks. We assume a typical track is 1.5MB. OR The handset does not support a memory card but has sufficient internal memory to store upto 50 1.5 MB music files.	card which can store upto 50 typically sized downloaded tracks. We assume a typical track is 1.5MB. OR The handset does not support a memory card but has sufficient internal memory to store upto 50 1.5 MB music files.
JAR Size	The phone supports a suitably large JAR size	Specification states JAR size >= 400kb[4]	Max jar size supported is >= 400kb[4]	Max jar size supported is >= 500kb[4]
Heap Size	The phone supports suitably large maximum heap	Screen size is 128x160 and max heap >= 1MB Screen size is 240x320 and max heap supported >= 2 MB Best to define the rules for this for a range of screen sizes.	Needs refining Best to define the rules for this for a range of screen sizes.[5]	Needs refining Best to define the rules for this for a range of screen sizes.

[4] For border-line cases these tests could be made more accurate by taking into account screen-size. For example, the 128x160 screen size JAR is currently 390K. The 356x414 screen size JAR is currently 460K

[5] To calculate required heap will be a formula like
BASE_HEAP_REQUIRED + (SCREEN_HEIGHT x SCREEN_WIDTH x COLOUR_DEPTH x IMAGE_PIXELS)
Where BASE_HEAP_REQUIRED and IMAGE_PIXELS are static for each client_version, and SCREEN_HEIGHT, SCREEN_WIDTH and COLOUR_DEPTH come from the device.

Capability	Description	Criteria to meet for a Specification Likely Pass	Criteria to meet for an Detective Acceptable Pass	Criteria to meet for a Detective Full Pass
Play from a stream	Can a music file be played from a stream as it is decrypted. This capability is important to provide security for the music file once it has been downloaded to the phone. Without this capability the music file would be exposed decrypted on the phone's filing system.	Assume will be able to if MMAPI present	None	The MIDlet can create and start a Player object from a stream. [6]
MIDlet OMA Forward Lock	The ability to apply OMA v1 forward lock to the MIDlet to prevent it from being transferred off the handset if the user gains access to it.	Assume will be available	None	A MIDlet can have OMA Forward lock applied to it and the handset refuses to allow that MIDlet to be copied off or sent from the handset

5.2 Performance Capabilities

Memory card read/write performance	The performance of the MIDlet in reading from and writing to the removable memory card used for storing music.	Assume will be acceptable.	Read a 1.5 MB file from memory card in under 3 secs Write a 1.5 MB file to memory card in under 3 seconds	Read a 1.5 MB file from memory card in under 2 secs Write a 1.5 MB file to memory card in under 2 seconds
Decryption performance	The performance of the MIDlet in running the decryption algorithms required.	Assume will be acceptable	Decrypt a 1.5MB music file in under 6 seconds	Decrypt a 1.5MB music file in under 4 seconds
Data download performance	The performance of the handset in downloading data over the network under a good	Assume will be acceptable	A typical 1.5MB music file can be downloaded to phone memory in less than	A typical 1.5MB music file can be downloaded to phone memory in less than

[6] There are potential technical solutions to avoiding the explicit requirement for this capability. This solution would involve creating a player from a file and decrypting the file just ahead of the play point and encrypting it behind the play point. This solution can be investigated if priority handsets fail this criteria.

	connection.		T_1 secs (GPRS handset) T_2 secs (EDGE handset) T_3 secs (3G handset)	T_4 secs (GPRS handset) T_5 secs (EDGE handset) T_6 secs (3G handset)
Data upload performance	The performance of the handset in uploading data to the network under a good connection.	Assume will be acceptable	A typical 64kb data file can be uploaded to the network in less than T_7 secs (GPRS handset) T_8 secs (EDGE handset) T_9 secs (3G handset)	A typical 64kb data file can be uploaded to the network in less than T_{10} secs (GPRS handset) T_{11} secs (EDGE handset) T_{12} secs (3G handset)
Screen drawing performance	The rate at which images can be drawn to the handset's screen	Assume will be acceptable	The number of frames/second drawing a typically sized MusicStation screen containing a typical number of individual images is greater than X frames/second	The number of frames/second drawing a typically sized MusicStation screen containing a typical number of individual images is greater than X frames/second

5.3 Development Procedures Capabilities

The following capabilities are connected with Omnifone's development procedures. They are required to pass for the handset to run through the commissioning and testing process. However they are not directly related to a production release where we would expect a different signing process.

Capability	Description	Criteria to meet for a Specification Likely Pass	Criteria to meet for an Detective Acceptable Pass	Criteria to meet for a Detective Full Pass
Ad Hoc Signing Supported	The handset supports one of the available methods for ad hoc signing available to Omnifone for use during development and testing. For example, the handset has a Verisign class 3 (3rd party trusted) certificate.	Assume this will be accessible if specification states MIDP v2.0 support	None	The handset supports Ad Hoc signing using one of Omnifone's available methods. These include various handset manufacturer signing mechanisms.
Ad Hoc Signing supports no prompt file connection permission	The ad hoc signing method causes the handset to either set the file connection permission to a permission which	Assume this will be accessible if specification states MIDP v2.0 support	ASK EVERYTIME permission is automatically set	NEVER ASK permission is automatically set Or NEVER ASK permission

	allows file access or once the signed application is on the handset that permission can be manually set.		Or ASK EVERYTIME permission can be manually set	can be manually set Or ASK ONCE permission is automatically set Or ASK ONCE permission can be manually set
Ad Hoc Signing supports no prompt HTTP connection permission	The ad hoc signing method causes the handset to either set the HTTP connection permission to a permission which allows file access or once the signed application is on the handset that permission can be manually set.	Assume this will be accessible if specification states MIDP v2.0 support	ASK EVERYTIME permission is automatically set Or ASK EVERYTIME permission can be manually set	NEVER ASK permission is automatically set Or NEVER ASK permission can be manually set Or ASK ONCE permission is automatically set Or ASK ONCE permission can be manually set

6. Expected Capabilities

The following capabilities are expected and will be used if present. However, the lack of support for any of these capabilities does not cause a handset to be incompatible with MusicStation. Due to lack of detailed information is generally not possible to make a specification-based assessment of these capabilities.

Capability	Description	Criteria to meet for a Detective Acceptable Pass	Criteria to meet for a Detective Full Pass
Handset Play Control Keys	Music control keys on the handset are available to the MIDlet	Some of the keys are accessible by the MIDlet	The MIDlet gets events from the following handset keys where they exist: Stop, Play, Pause, Next, Previous
Headset Play Control Keys	Music control keys on the headset (headphones) are available to the MIDlet	Some of the keys are accessible by the MIDlet	The MIDlet gets events from the following headset keys where they exist: Stop, Play, Pause, Next, Previous
Handset Volume Control Keys	Volume control keys on the handset are available to the MIDlet	None	The MIDlet gets events from the following keys: Volume Up, Volume Down
Headset Volume Control Keys	Volume control keys on the headset are available to the MIDlet	None	The MIDlet gets events from the following headset keys: Volume Up, Volume Down
Support for progressive playback	The MIDlet is able to playback a track in progressively – downloading, decrypting and playing all in one	None	(1) The handset does not support progressive playback

	action.		OR (2) A track can be downloaded, decrypted and played directly without the music file having to be temporarily stored in the phone's filing system in a decrypted form. Writing of the decrypted file to the filing system makes it insecure since the file could be accessed outside the MusicStation application.
MIDLet can run in background	The MIDlet can be minimised and continues to play music in the background when it is.	None	MIDlet can be minimised whilst playing music in the preferred format and music continues playing.
MIDlet can run when clamshell or slider is closed	For clamshell handsets, the MIDlet should continue to run when the clamshell or slider is closed, with an event notification to say that this has occurred.	Not a clam shell and not a slider	(1) Not a clamshell. OR (2) Is a clamshell and music continues playing when the clamshell is closed and opened OR (3) is a slider handset and music continues playing when the slider is closed and opened
Access to the External Screen		No external screen	(1) No external screen OR (2) External screen and MIDlet can write to it

7. Predicted Group Compatibility

We classify handsets into groups, which typically match up with the different handset platforms that a manufacturer deploys in their handsets (for example, Nokia s40 3rd Edition FP1).

Where an individual analysis of the specifications of a device are not performed then it is sometimes possible to make an assessment of likely compatibility based on the group membership of a handset. The reliability of this manner of assessing compatibility is dependent on the robustness of the handset manufacturer's approach to deployment of consistent handset platforms.

	Criteria to meet for a Group Likely Pass	Criteria to meet for a Group Reliable Pass
Group membership	If handset is in a group which contains handsets which have previously passed Detective. Previous	If handset is in a group formed by a platform supplied by a platform-based manufacturer

	experience needs to have shown that grouping of handsets for that manufacturer is a reliable predictor of support. Currently we only use this criteria for Motorola and recent Samsung handsets.	(currently limited to Nokia and Sony Ericsson) and other handsets in that group have passed Detective.

Where device specification information has been collected that device specification information should always override the group information.

The user interface seen on the phone is made up of many small images pieced together to give the desired appearance.

For example the scrollbar image is constructed from three distinct sections. These three sections are the top of the scrollbar, the center of the scrollbar, and the base of the scrollbar. The scrollbar shown on the screen will always be made up of one ìtop of the scrollbarî image, one ìbase of the scrollbar imageî and many ìcenter of the scrollbarî images. The number of ìcenter of the scrollbarî images depends on how long the scrollbar is required to be.

The reason that 3 smaller images are used rather than several larger images is because mobile devices have limited memory and donít always have enough memory to store several larger images.

If we want to change the visual appearance of the scrollbar, then we would need change these three images.

Each handset which has a different screen resolution requires a different set of these three images because the images need to change size for each handset.

Most mobile devices only display raster images, and not vector images. Raster images do not resize very well. Vector images resize well.

What we do is to take a Vector graphic representation of the scrollbar. Resize this vector graphic to the correct size for the target screen resolution, rasterize the resized vector graphic, and then cut up the rasterized version of the image into the component parts. (e.g. the top, center and base of the scrollbar)
These component parts are then recombined by the mobile phone application to create the desired appearance.

This is done for each image that makes up part of the User Interface.

If a raster image was resized for different devices it would give a pixellated appearance. And if the image was not cut up into component parts then the images would use up too much memory on the mobile phone.

There are around 100 raster images used in the MusicStation user interface, these are generated from around 50 vector graphics. There are currently 10 different screen sizes.
This means to change the User Interface appearance we would only need to change 50 vector images, rather than having to change 1000 (100 images x 10 different screen size) raster images.

MusicLoader™ Data Procurement and Ingestion Services High Level Design

Alex West/Mark Knight – Last Update 3rd October 2006

Status: Draft

Table of Contents

1. Terms of Reference

This document sets out to define a first-cut ELT (Extract Load and Transform) process for data from a variety of sources which will be used to describe music and associated content for the Omnifone MusicStation™ product. The document is intended to provide a jump-off point for discussions and review towards establishing an approach for the required data services.

The document status is Draft and feedback and comments are welcome.

It is intended that the document should grow and change to reflect actual decisions taken during this early design and implementation stage to become a documented reference for the service.

2. Scope

The initial scope will be limited to the Muze data feed but is mindful of, and will expand to include, data feeds from 24/7 Music Shop and others. Data Merge services for 24/7 have not been included within the scope at this point but will be in due course.

Four areas of processing and services have been identified (see overview PPT). These are GRAB, FEED, MERGE and PUBLISH. The PUBLISH service has only been given minor treatment here as it partly resides within the domain of the MusicLoader GUI and partly within the domain of the MusicStation production database environment.

3. Objectives

The principal objective at present is to design a service capable of procuring and processing data from two main sources and to consolidate the data into a useable set for the product launch. This needs to be accomplished in a relatively short timeframe for the initial suppliers; Muze™ and 24/7 Music Shop™.

Additional objectives are:

- To provide a flexible and robust platform for data procurement that will allow Omnifone to easily handle both good and anomalous data from external sources.
- To seamlessly handle both full and incremental updates from each supplier.
- To automate data feeds and minimise human intervention wherever possible.
- To procure data from a variety of disparate sources while minimising impact on the service, the production environments and the MusicStation product.
- To establish a common format and process plan for all data wherever possible.
- To build the GRAB, FEED, MERGE and PUBLISH (GFMP) components within a modular architecture that permits any component to execute in isolation or as part of a suite.
- To keep a data archive of all feeds for potential future use.
- To facilitate analysis and reporting of all available data to help establish which data is most useable for the MusicStation™ product in both the short and long term.

4. Overview

NB Please also see the PowerPoint document MusicLoader Overview (2).ppt
As well as appendices that record aspects relevant to specific suppliers, the document is divided in to various sections that discuss the following main areas of handling data for MusicStation:

- Evaluation of third party products
- Data GRAB and FEED Services
- Data MERGE Services
- The MusicLoader Application (GUI)
- Data PUBLISH services

5. Due diligence and evaluation of third party products

In order to assess the degree of in-house development versus the suitability of existing platforms and ELT tools, a variety of current products were evaluated. Below is a brief explanation of findings.

5.1 Microsoft Biztalk Server

MS Biztalk Server was investigated, but not heavily, as it quickly became evident that it is far too heavy for the job and much better suited to a very wide set of data and product suppliers, or a high volume of transaction data and requires a very significant investment in the technology to make it worthwhile.

See Also:
http://www.topxml.com/conference/wrox/2000_vegas/Powerpoints/stephen_biztalk.ppt#1
http://www.microsoft.com/biztalk/evaluation/what-is-biztalk-server.mspx

5.2 Microsoft SQL Server 2005 SSIS (formerly Data Transformation Services DTS)

Investigation of this ETL (Extract, Transform and Load) toolset led to a more in-depth study and the compilation of separate report on its capabilities (See associated SSIS Features and Functionality Report). It ticks all of the boxes that we identified for GRAB FEED and PUBLISH and is fully extensible. It is therefore recommended that SSIS is used at least for these three MusicLoader services and for some of the MERGE services as where appropriate.

Because SSIS ships with SQLServer 2005 and has functionality tightly coupled with that environment, it is further recommended that MS SQLServer 2005 is used as the platform for MusicLoader as a whole and that all data operations are performed within that environment prior to moving the data to an Oracle pre-production (staging) environment from where it can be loaded very rapidly to the production database.

SQLServer 2005 also provides a further comprehensive set of components which includes Analysis Services (SSAS) and Reporting Services (SSRS), as well as an integrated set of development and administration tools, such as the Business Intelligence Development Studio and the Server Management Studio. In combination with SSIS and the DB engine, these components and tools allow integrated development and managed scalability and should afford MusicLoader a speedy start and a rich set of native functionality to build an interoperable solution with maximum leverage that will deal with the considerable challenges of matching, integrating, pruning, analyzing, correcting and ultimately using data from a varying set of sources.

See Also:
http://msdn2.microsoft.com/en-us/library/ms137795.aspx

5.3 Altova Mapforce

Altova Mapforce is a powerful tool for mapping XML to XML schemata and XML to SQL schemata. It also handles data transformations but is weak on outputs directly into SQL databases, creating SQL scripts instead which must be run separately. Nevertheless it could prove useful later as a documentation tool for complex mappings. Additionally XMLSpy™ may prove useful for producing schemata that Omnifone wishes it's suppliers to use and DiffDog™ may be useful for identifying schema changes from suppliers.

See Also:
http://www.altova.com/products/mapforce/data_mapping.html

5.4 The Oracle XE ELT tools

These were examined with a view to automating them. The Data Load/Unload wizards can only be run from a browser based GUI but the SQL*Loader, Data Pump and Generic Import/Export utilities can all be run from the command line and therefore from a scheduler, script, NT service or executable program. The **Oracle Data Pump** should prove particularly useful for the proposed link between the MusicStation Oracle pre-production and production databases

See Also:
http://download-uk.oracle.com/docs/cd/B25329_01/doc/admin.102/b25107/impexp.htm#BCEJBFDF
http://download-uk.oracle.com/docs/cd/B19306_01/server.102/b14215/toc.htm

5.5 Powerbuilder and the PB XML Sax Parser

These routines can load XML based on DTD or XSD and may be useful for identifying unexpected schema changes from suppliers, but we might not need this particular functionality if we are using SSIS. We do however propose to use Powerbuilder to construct the MusicLoader management GUI application and some of the MERGE service components. PB 10 can talk COM which is well suited for object library interaction as it allows communication between libraries of software routines written in any language. Furthermore, PB11 will be able to talk .net which opens up a world of interoperability. We will use PB to develop the client and any required server components. Using extensively Microsoft software through COM, .net and c/c# library calls directly in PB, will allow us to call any executable part of the Windows environment and allow basic OS control through external software components. Moreover, PB's rich datastore/datawindow handling will be a powerful toolset for moving or changing data and it is perfectly capable of talking to the things we need to talk to including third-party tools, a clear benefit being that a harness can easily be used to develop and test a whole server component on a windows machine before taking the finished exe and setting it up as an NT service if required.

6. MusicLoader GRAB and FEED Services

6.1 Muze Data

Muze are contracted to provide the following data to Omnifone:

Core

Database/File Set	Available Formats	No. Files	Size	Delivery	Rows	Freq	Content
Muze Music UK data (all formats)	ASCII/DBF	15	180 Mb	Full/Delta	Full=250K Delta=1K	Full=Mthly Delta=Wkly	Main database of all bands and releases
Muze Music UK Images	jpeg	10	3.5 Gb	Full/Delta	Full=140K Delta=1K	Wkly	Albums & singles cover art supplied as a 170 x 170 pixel jpeg (*.jpg)
Muze Essential Artists	ASCII/XML/ACCESS	1	1.5 Mb	Full	Full=50K Delta=100	Mthly	Additional rich content on artists
Muze Tunes		1	9 Mb	Full	Full=1.6M Delta=1.5K	Wkly	Text database of URLs that link to preview sound clips
EPM	ASCII	1	14 Mb	Full	Full=16K Delta=100	Mthly	Encyclopedia of Music - Biographies and addition rich content for > 9,000 performers

Additional

Muze Essential Artists Images	??	??	200 Mb				Have we licensed this product ?

Muze can supply data in a variety of data formats (FoxPro, ASCII, and XML) but only in flat file ASCII delimited by the pipe character for the "Muze Music All Formats" database at present. It is therefore recommended that we standardise on this format for all feeds until Muze are able to provide us with XML for all feeds.

The data is supplied as flat files which are simply data dumps from their relational databases. The relational structure and the keys are maintained within the data.

For the larger databases the data is available as full data sets (database dumps) or incremental (delta) data sets. Incremental sets have flags to indicate Inserts, Updates and Deletes. The smaller databases are only available as full (replacement) data sets. Some sets are updated weekly; others monthly. Set schedules are published and as far as we know adhered to. Previous incremental updates are available as FTP Pull.

6.2 Muze Data Delivery

During development Muze will send the larger initial databases by post on DVD and the incremental and smaller full sets by email. These are also available as FTP "Pull" feeds. Muze have declared that they are willing to "Push" files to our own FTP area and have indicated that they are converting all feeds to XML but will not commit to a timeframe for the latter.

Our goals are therefore:

- Take all feeds on DVD to get started
- Take subsequent deltas and small database reloads as FTP "pull" (only if required) or email
- Set up an Omnifone Isolated User FTP site and get Muze to "Push" deltas and small incremental reloads to Omnifone
- Work with Muze in the longer term to receive XML feeds either by SOAP or FTP

6.3 GRAB and FEED Services for Muze

It is recommended that the MusicLoader GRAB and FEED services are run entirely under the aegis of SSIS and that any missing functionality should be either scripted or use external components such as UNZIP utilities or PB server components. Please see the separate report on SSIS functionality (SSIS Features and Functionality Report.doc).

For Muze data minimally the following tasks and associated logic will be scheduled in SSIS

- FTP pull or local pickup of FTP pushed files at predetermined intervals from specific locations.
- External utility will UNZIP the files.
- A separate task will load them while logging any errors and notifying relevant personnel of any problems encountered.
- Full or Partial back-outs and reloads of data based on business constraints and errors.
- Move loaded files to designated archive locations after successful GRAB and FEED

SSIS will administer all of the relevant variable data parameter values and run specific components of the GRAB & FEED services in the required order.

A simplistic high level pseudo-code for an SSIS package might be:

```
==========================================================================
        Start or run service
                Check parameters for enrolled suppliers
                While more suppliers to process
                        Process supplier n (in this case Muze) as
                                Get supplier data from parameter location
                                Log Status as 'Processing'
                                Check for files
                                Unzip files
                                Count files
                                While more files to process and no errors
                                        Open File
                                        Count Columns
                                        Load file n
                                        Log stats or errors
                                        Log Status as 'Processed'
                                        Archive files to parameter driven area
                                        ....
                                End
                        ....
                End
        End or idle
==========================================================================
```

6.4 Keys

We will load and use Muze primary and foreign key values for all Muze data in this process. The rationale behind this is that if we keep the Muze keys it's easier to perform any referential integrity checks within the data as is comes in, or shortly thereafter.

6.5 Supplier Data

We will keep all supplier data both in archived file formats as well as in database tables. Rules will be applied so that no subsequent feed can be loaded if prior feeds did not load thus avoiding missing updates.

6.6 Load Order

- Files need to be loaded in specific order if we want to use integrity checks on loading or afterwards.
- Every feed will be identified by a combination of a supplier prefix, a serial ID and a timestamp to enable a thorough audit, data recognition and data correction operations.

6.7 Supplier Data Structure

The Service will not be data driven from the perspective of the data structures that are being fed and checks must be automated to ensure that the data received is structurally what we expect. The service will not attempt to load data that is the wrong shape. Instead an admin functionary will be alerted to review any issues with aborted loads. It is operations such as this that will require a GUI to run specific SSIS operations. Additionally SSIS can check XSD or DTD documents ahead of a data feeds and can also make checks on delimited files for the correct number of fields.

6.8 Graphical Files

For graphical files we will set up a separate SSIS task chained to a load of the relevant URL data for the image prior to moving the files to a live access location. This task may also need to be chained to the PUBLISH service for MusicLoader.

6.9 Data Management

Certainly during development and early deployment and perhaps perpetually, we will keep complete database back-ups before and after full feeds so that these can be restored in the even of serious processing errors. These back-ups will typically be separate from normal daily back-ups. As the data volumes grow we can look at applying transaction log restores instead of full database dump and restore operations.

6.10 Design

The design and construction of the GRAB and FEED must be modular enough to allow a GUI to run and back-out individual steps relevant to a specific operation.

7. MERGE Services

7.1 Overview

For maximum flexibility it is recommended that the MERGE service is implemented as a combination of the following:

1. A package of SQL Server T-SQL procedures.
2. A PB NT service with rich datawindow functionality that can powerfully and quickly execute major and minor operations.
3. A PB GUI (called MusicLoader) application that can operate and control the above in a variety of ways and additionally perform a range of other operations on data as required.

The modularity will allow the service to be either:

1. Run as a scheduled service.
2. Called independently by the MusicLoader GUI Application.
3. Triggered by a successful FEED service completion.

As the complexity of, and familiarity with, data from various sources grows, so too can the functionality of this service, offering a number of data transformation tasks that can be run from the MusicLoader application.

The service will attempt to process all data that is successfully loaded into the FEED tables and aggregate this data into a set of tables similar in structure to the MusicStation Production Database while logging errors and moving data that cannot or should not be processed into a holding or pending area. It is this merged data that will be published to the Oracle staging and production environments.

It may be that several manual operations are required to fully process the data especially if data is missing or malformed. Rules will also need to be established to sideline data that will never be required or is not required at present e.g. data relating to classical recordings or data about music only available on non-digital media.

The PB service will typically comprise of a single core process (MusicLoader Core) which is always running and a set of child processes (MusicLoader Services) which on starting read meta-data about what they are supposed to do - hence we can run any stage in any process. Child processes will use status priorities and communication with the core process through COM at start-up and throughout via any methods we care to write.

This service will not initially have the capacity and functionality to load a massive amount of continually changing data but we must design for linear scalability if we eventually want to handle considerable throughput. Because the MusicLoader services are decoupled from the production environment, the criticality of the service is eased but the environment will need appropriate backup and break-fix cover to allow fast disaster recovery. A distributed environment will allow us to run multiple NT GRAB/FEED servers at the same time, all inter-operating, based on configuration db tables to storing control information about which service is handling what.

Component Overview
MusicLoader Client App (PB 10/11) – windows - talks COM to COM
MusicLoader Core (PB 10/11) – NT
MusicLoader Com (PB 10/11) – NT - started and controlled by Core – provides API server for Client
MusicLoader Stage (PB10/11) – NT - started and controlled by Core – performs a stage in the process
MusicLoader DB (SQL Server) – NT

7.2 Data Consistency

A major concern with receiving overlapping data from different sources that refers to the same subject matter is consistency and how to maintain it. The simplest way to overcome this is to distinguish between:

- Music content-data i.e. all artists, albums, tracks and other data such as biography, reviews etc.
- Music meta-data i.e. sellable albums and tracks and where they are located

Using a Library and Bookshop analogy we can refer to music data content as LIBRARY data and define it as the data that will ultimately be seen and used by MusicStation users to explore and identify music.

We can refer to the music meta-data as BOOKSHOP data and define it as the data that is primarily used to enable users to download or play that music from wherever the physical files reside.

A typical example of LIBRARY data is that supplied by Muze and an example of BOOKSHOP data is that supplied by 24/7 Music Shop. Wherever possible, a given artist should only be associated with a single LIBRARY data supplier. In the short term therefore, to ensure consistency in this model, we should endeavour to only use LIBRARY data from Muze and only use BOOKSHOP data from 24/7 so that even if for example Muze identifies the artist Frank Zappa as Frank Zapa and 24/7 identifies him correctly, we would still use the incorrect data from Muze and work with them to correct it.

This of course falls down if there are artists which 24-7 provides but which Muze do not - in that case 24/7 would be considered the LIBRARY for such artists. This can be handled by rules e.g. "on 24-7 load if an artist does not exist; create it" – At a later point if the artist shows up on a Muze feed then we would update from Muze and alter the artist's LIBRARY tag to Muze

The main advantages in this approach are:

- For a given artist we only have to deal with a single supplier for LIBRARY data issues.
- We do not have to maintain nearly so much information about the source of any LIBRARY data.

The above paradigm will only work on an ideal but simplistic data supply model and where our LIBRARY supplier for a give artist can supply all of the data that MusicStation needs. Where this is not possible we need to associate LIBRARY data at a lower level. Ideally this would be at an entity level lower than Artist e.g. we take recording data for an artist from Supplier A and biography data from supplier B.

The worst case would be to have to associate LIBRARY data at the column/data item level and this would only occur if we had multiple suppliers giving us nominally the same data with very high levels of currency. For entity or lower level audit we would have to introduce precedence constraints to the data Merge services as well as rules and heuristics, e.g.

> If we received feeds A1 B1 A2 B2 B3 B4 B5 B6 B7 B8, we might decide or discover that A is out of date and that we should take B's input instead. A typical rule would therefore be that if supplier A normally has a higher precedence but has not supplied an update for x weeks and in the meantime supplier B sends an update, then we use supplier B for the update to that particular entity for a given artist.

This data driven approach might require the construction of dynamic SQL with which there are processing overheads.

In future, requirements will grow to include the following:

- Bulk transfer of artists from one LIBRARY supplier to another.
- Removal of all data from a supplier to conform to contractual obligations. This is covered in the short term by associating the LIBRARY source for an artist and if we one day wanted to cancel the Muze contract and replace them with 24/7 as the new LIBRARY supplier, then we might do this as follows:
    - Find the last full update feed from Muze
    - Apply it, treating it as a complete incremental update, but instead of updating or inserting data into the production db, remove or blank it
    - Perform the same process for any subsequent incremental processes
    - Find the last full data feed from 24-7
    - Apply it with 24-7 set as the LIBRARY
    - Apply any subsequent incremental feeds from 24-7

7.3 Use of Keys

MusicStation Keys
It is recommended that the MusicLoader database is the system of record for Primary Keys and Foreign Keys for MusicStation data and that these are created as part of the Data Merge Services. The rationale for this is that if they are created and maintained in MusicLoader then there will be consistency with all downstream systems without having recourse to mapping and maintaining mapped keys. This will provide us with a uniform means of identifying Omnifone data throughout.

Other advantages of this are:

- We can apply updates to production or staging without mapping keys
- We can load data faster without key generation in production
- We can load data faster to production without referential integrity checks

There will need to be a way of fast-tracking changes or new data to production so that we can deal with emergency situations e.g. where missing data means that we are failing in a promotional contract obligation. In such circumstances there needs to be a way to short circuit the Merge process but still generate the appropriate keys and data in the right place before quick-fire propagation to either pre-production and/or production.

Industry Keys
One challenge will be matching data across suppliers and this will eventually be mitigated by consistent and accurate ISRC, UPC and EAN codes from suppliers. SQLServer 2005 has some very useful tools that employ fuzzy logic to help identify related records that cannot be accurately linked using standard industry codes or where industry keys not being used as primary keys are inaccurate. It should be noted that although Muze are supplying ISRC codes with all data, for reasons of expediency we are taking this data ahead of their 'cleaning' operation so there will be anomalies.

7.4 Auxiliary Operations

Data Cleansing
It is foreseeable that we will have to make modifications to data in order to get it production ready in situations where either we recognise a problem or it's brought to our attention and the correction cycle of the LIBRARY supplier is too slow. In such circumstances we need to be careful that any good updates that we make are not written by further bad updates from LIBRARY suppliers. We could, for example use a timestamp to tell us that the system has modified such records, and if a LIBRARY feed then attempted to update such a record, we would need to implement something to stop it such as a timestamp and a check of values prior to our update.

Actions such as this can be handled but are complex to implement. Wherever possible we should marshal our supplier's data but, by and large, we have to trust that our supplier knows better than us as editing and compiling meta-data is not our job: there will always be excellent companies providing these services and so we will always be using one of them.

For transformations that we know how to handle, such as expletives for legal conformance, we can implement rules that will always transform content in the same way.

For a specific service, we may rely on a supplier (hypothetical e.g. - DX3 in China) for data and not our LIBRARY which would be ok, as there is probably no real overlap between them.

7.5 Minimum Usable Data Sets

We need to establish the minimal sets of MusicStation data for artists. This has already been established within MusicStation with attendant rules such as cardinality e.g. can an artist not have a biography? These rules need to be encapsulated into the logic for Merge processing.

7.6 Technical Constraints

If we are to build a modular set of services that can be run from the GUI as well as a scheduled then we will need some status indicators to prevent duplicate simultaneous operations.

7.7 Audit and Reporting

Log tables are required for audit purposes as well as for logging processing errors and data errors. Additionally, data versions will be logged to allow us to understand what data has been processed.

8. The MusicStation GUI Application

It is envisaged that the MusicLoader application will grow in functionality together with the needs of MusicStation especially with regard to managing anomalous data and allowing it's users to transform data prior to Aggregation. Some short and longer term functionality such as that below should feature in the App:

Short Term

- Manually control or trigger elements of SSIS for Grab and Feed services on an ad hoc basis such as clearing down reception tables, reloading current and previous data sets and initiating a complete Grab or Feed cycle process.
- May need (subject to SSIS functionality) to report the errors to the suppliers through a simple inbox-style web interface whereby they can view the errors we have found in their data feeds
- Control manual operations in the Merge Services as required.
- Perform checks and status reviews of all services and processes
- Perform Data transformations and corrections as required
- Control Publish Service data loads and complete reloads into pre-production and production including trigger of Oracle Data Pump or other third-party software.
- Control the data loads through a staged process where each stage can be re-executed and all data is stored for each re-executable stage (maybe not all). E.g. for each stage we create a window into its operation and if possible a direct view of the data as it is coming in or being transformed. Tremendously impressive for demonstrations!
- It talks to PowerBuilder Com through .com hence the PB Core can be on any machine on the network. It does not talk to the DB directly.
- There may also be sub-stages for each of the above which may be re-executed using the MusicLoader Client.

9. PUBLISH Services

The MusicLoader PUBLISH service will again be a set of components that can be run as a complete process either controlled by the MusicLoader application or by SSIS or by a combination of the two.

Control parameters will therefore either reside in a table or within SSIS and this information will permit the application to load compatible and complete sets of data as well as specific revisions as required.

It is recommended that the service is decoupled from the production environment by the introduction of a pre-production (staging) environment that will afford both rapid deployments to production as well as verification that the data is complete and accurate. The PPT overview shows the staging and production environments separately for clarity and is not a recommendation.

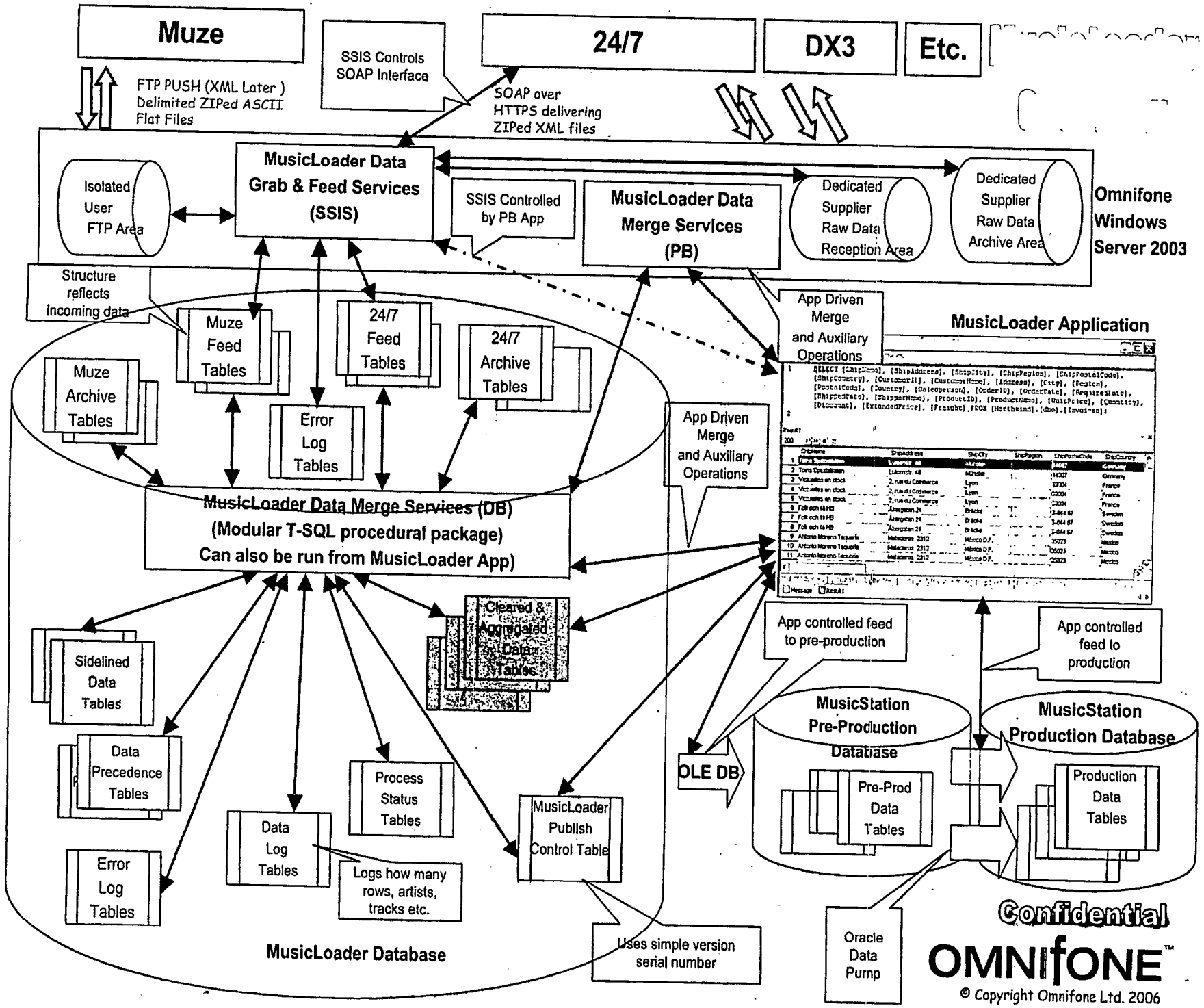
10. Development Steps

- MK/AW iterate design until MK happy
- Present to PS/SP/SG/TB/AD for iterative review until SP happy
- V1 – Muze full load
- V2 – Muze incremental
- V3 – 24-7 full load
- V4 – 24-7 incremental

11. Appendix 1

Technical Notes

FTP
IP addresses/Active directory options
Firewall considerations
Access = Read Write & Log
Opinion on policy for access control with IP range or DNS restrictions or basic account authentication (login) with port blocking



From: Chris Evans
Sent: 19 January 2007 16:31
To: Steve Pocock
Subject: Requirements for PC client

Steve

Please have a quick glance at my rather brief requirements for the PC client below.

What are the many things that I have missed?!!

Thanks

Chris

Phase 1

-Ability to play music (with FFWD, REWD etc...)

-Playlist management.

2-way Playlist synchronisation with mobile app (N.B. Synchronisation from PC client to the handset is invoked by a button press (allowing the customer to experiment with playlists and music before synching with the handset). In the other direction it is suggested that synchronisation is automatic.)

People could live without this. Phase 1.5.

CE: I donít agree. What really is the point of the PC client if not to provide more usable way of managing your playlists etc? All MNOs think this functionality will be there as well.

-Search (plus Advanced Search and Classical Search ??)

-Side-loading.

Is side-loading the actual functionality that operators want (in which case this should stay in phase 1) or is it simply giving their users access to music which is on their PC? If the latter then this is a phase 1.5 item and the iTunes etc sync stays as a phase 1 item.

CE: All MNOs are asking for side-loading to cover music that is not in their catalogue.

-Pc Client has 3 states: sys tray, desktop player, full tool

Napster gets away with only the full screen player so the others could be phase 1.5

CE: Think this should all be phase 1

-iTunes & WM player integration (Q: How does your list of music in iTunes get represented on the mobile? MyTracks for example now onlyb list Tracks that are on the handset, whereas we realy need a list of all tracks that you have listened to)

Needs to include presenting any iTunes playlists you may have defined

-Initial bonding to mobile client (e.g. send SMS containing random code to user's phone number which then has to be entered and validated in PC client (and subscription status confirmed) before it is allowed to be used).

Unbonding and handling of the user upgrading their handset

-Subscription management

Some minimal acceptable level of branding

Simple download and installation

CD based installation

Simple help (or link to the Warrenís FAQ stuff to start)

Phase 2

All the features of the mobile application. These would be too many to list (and quite difficult given that we don't actually have a list for the mobile client), but would be everything that you find in Play, Inbox, Buzz (should we allow messaging)?

CD burning?

Rating

Full branding

Full help

-----Original Message-----
From: Ajaipal Johal
Sent: 08 February 2007 15:22
To: Lucien Rawden
Subject: Big Trees

"A method to enable the deep analysis of hierarchical data structures through the dynamic creation of a forever-expanding object tree"

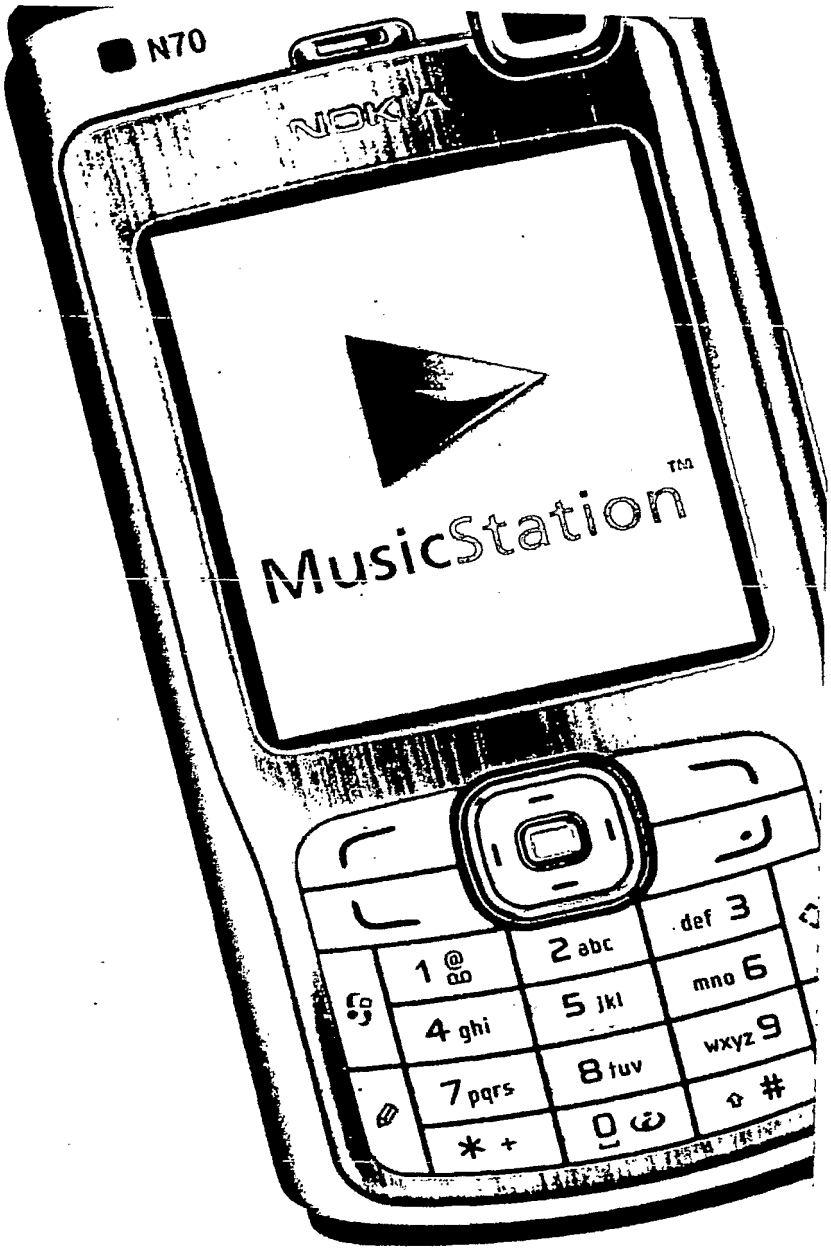
The PC Tools, which we have built, comprise data-driven objects, which enable the quick build of dynamically, and forever exploding tree structures of hierarchically organised objects. The concept relies on the fact that the tree registers an object and callbacks about how to expand it.

The initial registration of the object to the tree links a type of object to the root of the tree. The business relationships of the objects are described in a series of callbacks to the data store. This data store can take any form, be it a traditional database, an application server or SOAP server.

These relationships thus define the outcome of the expansion of the item on the tree.
Hence, when an object can be placed on any tree and expands, it shows both the things it contains and those that reference it.

Each of these things are themselves objects with their own defined callbacks. E.g. a playlist will have a set of tracks, the tracks will have associated artists, the artists will have albums, ad infinitum. Thus there is the path followed need never end.

Hence it is a forever-expanding tree with no depth limitation. This allows deep data-mining to occur without having to repeatedly re-open branches of the tree.



OMNIfONE™
Global mobile phone applications

MusicStation J2ME Client DRM Whitepaper

December 2006 – Version 1.3

Contacting Omnifone

Island Studios, 22 St Peter's Square, London W6 9NW
Email: info@omnifone.com
Web: www.omnifone.com

Change History

31st October 2006	Version 1.0	Initial document release.
27th November 2006	Version 1.1	Minor corrections.
1st December 2006	Version 1.2	Minor update.
4th December 2006	Version 1.3	Minor update.

Contents

Definitions

The following table provides a definition for each of the terms and acronyms used in this document.

AES	Advanced Encryption Standard is a symmetric key cryptography standard adopted by the U.S. government. AES is the encryption method used to protect the sensitive parts of Rights Objects.
API	An Application Programming Interface (API) is the interface that a computer system, library or application provides in order to allow requests for services to be made of it by other computer programs, and/or to allow data to be exchanged between them.
All-You-Can-Eat (AYCE)	An unlimited download access model which allows users, within valid subscription period, access to the entire music catalogue to play anything they want as much as they like. Effectively a time limited music rental model.
Certificate Authority (CA)	In cryptography, a Certificate Authority or Certification Authority is an entity which issues digital certificates for use by other parties. It is an example of a trusted third party. CAs are characteristic of many public key infrastructure (PKI) schemes.
Client Certificate	The MusicStation Device's public key certificate, signed by the MusicStation CA therefore attesting to the authenticity of the Device and it's Client Public Key.
Client Private Key	The private part of the RSA 1024-bit PKI public/private key pair. Only known to the Device allowing messages encrypted with the Client Public Key to be opened only by the Device.
Client Public Key	The public part of the RSA 1024-bit PKI public/private key pair. Messages encrypted with the Client Public Key can only be opened with the Client Private Key. Communications from the Device can be confirmed if signed by the Client Private Key as only the Client Public Key will be able to make sense of the signature.
Content Encryption Key (CEK)	The 128-bit RC4 cryptographic key used to unencrypted protected DRM content (protected music files).
Content Server (CS)	The networked server computer farm which stores and delivers, upon request, DRM content to MusicStation Devices.
Device Adaptive Architecture (DAA)	Omnifone's patented mobile handset commissioning architecture. Enables unique builds of the MusicStation application for a specific phone to be generated extremely rapidly and, in the main, automatically. This reduces engineering costs, vastly reduces the time to commission a new handset and enables Omnifone technical resources to move the MusicStation product forward.
Digital Rights Management (DRM)	A method employing cryptography to control and facilitate the legitimate distribution and use of digital media such as music.
Domain Key (DK)	A 128-bit AES symmetric key used to protect the Rights Encryption Keys (REKs) of the Rights Objects delivered to a specific MusicStation Device.
DRM Content	A content file (e.g. music track) which has been encrypted using a Content Encryption Key.
DRM Content Format (DCF)	A secure content package for DRM Content, with its own MIME content, also containing information such as content description (original content type, vendor, version, etc.), Rights Issuer URI (a location where a Rights

	Object may be obtained), etc.
IMEI	International Mobile Equipment Identity, a number globally unique to every GSM and UMTS mobile phone. It is usually found printed on the phone underneath the battery and can also be found by dialing the sequence *#06# into the phone.
IMSI	International Mobile Subscriber Identity, a globally unique number that is associated with all GSM and Universal Mobile Telecommunications System (UMTS) network mobile phone users. The IMSI is stored in the Subscriber Identity Module (SIM).
J2ME	Java Platform, Micro Edition or Java ME (formerly referred to as Java 2 Platform, Micro Edition or J2ME), is a collection of Java APIs for the development of software for resource-constrained devices such as PDAs, cell phones and other consumer appliances.
Mobile Network Operator (MNO)	Those companies that implement mobile telephone networks, provide handset connection, services & billing.
MusicStation	Omnifone's mobile phone based software application which allows users to discover, manage and listen to music on their phone on the move using the mobile network
MusicStation Device (Device)	Reference to a mobile phone handset with an installation of the MusicStation handset application.
MusicStation Root CA Certificate	The PKI public key certificate of the MusicStation Server Certificate Authority. This is the Trust Anchor of the entire PKI system in the MusicStation network.
MusicStation Server (Server)	A server farm of powerful networked computers offering MusicStation network services to and servicing the MusicStation handset client applications and it's requests.
MusicStation Service (Service)	An instance of a MusicStation server-side application for a particular MNO or handset vendor in a particular territory. E.g. Operator XYZ in UK is one MusicStation Service.
Over-The-Air (OTA)	Pertains to the delivery of an application (such as MusicStation) or content (such as full track music) to a handset using the MNO network.
PKI	Public Key Infrastructure is an arrangement that provides for trusted third party vetting of, and vouching for, user identities. It also allows binding of public keys to users. This is usually carried out by software at a central location together with other coordinated software at distributed locations.
RC4	In cryptography, RC4 (also known as ARC4 or ARCFOUR) is the most widely-used software stream cipher and is used in popular protocols such as Secure Sockets Layer (SSL) (to protect Internet traffic) and WEP (to secure wireless networks).
Right Issuer (RI)	The networked service which registers MusicStation handset applications, assigns Domain Keys and delivers Rights Objects.
Rights Object (RO)	An electronic document specifying permissions and constraints associated with a piece of DRM Content and governing how DRM Content may be used.
Rights Encryption Key (REK)	A 128-bit AES symmetric key used to encrypt sensitive parts of the Rights Object, such as the Content Encryption Key.

Rights Expression Language (REL)	An XML definition contained within a RO describing exactly how the content in the DCF associated with this RO can be consumed and used by the user.
RSA	An algorithm for public key encryption. It was the first algorithm known to be suitable for signing as well as encryption, and one of the first great advances in public key cryptography. RSA is still widely used in electronic commerce protocols, and secure given sufficiently long keys (e.g. 1024-bit).
Software License (License)	A 512-bit unique number embedded in a MusicStation application. Used by the MusicStation server to identify which Service the application is related to.

1. Introduction

MusicStation is a mobile phone based software application which allows users to discover, manage and listen to music on their phone on the move using the mobile network. Omnifone takes MusicStation to market primarily in partnership with Mobile Network Operators (MNOs) whilst working closely with the music industry to ensure the widest and best range of music is available to MusicStation users. Such vast libraries of digital music media are extremely valuable and need to be protected from theft and abuse whilst enabling valid paying users seamless access. Digital Rights Management (DRM) provides a method to control and facilitate the legitimate distribution and use of digital media.

The primary handset technology platform for MusicStation is Java 2 Platform Micro Edition (J2ME). This platform was chosen because it provides the widest mobile phone handset reach. This document describes the methods used by Omnifone's J2ME MusicStation handset application and associated network services to distribute protected content and securely issue the rights to use that content.

MusicStation's DRM is an implementation of the Open Mobile Alliance (OMA) DRM v2 specification. This specification has been widely adopted by both the mobile & music industries as their preferred method of protecting content for mobile devices. Whilst OMA DRM v1 has been widely adopted by handset vendors, at the time of writing, there are very few handsets which support OMA DRM v2. For this reason the OMA DRM v2 implementation discussed in this document is that which Omnifone has built into the MusicStation handset application and the associated MusicStation network services.

1.1 DRM Overview

Before content is delivered, it is packaged to protect it from unauthorised access. A Content Server (CS) delivers DRM Content, and a Rights Issuer (RI) generates and delivers associated Rights Objects. The Content Server and Rights Issuer embody roles in the system. Depending on deployment they may be provided by the same or different actors, and implemented by the same or different network nodes. For example, pre-packaged protected content can be distributed across multiple Content Servers for efficient delivery of content.

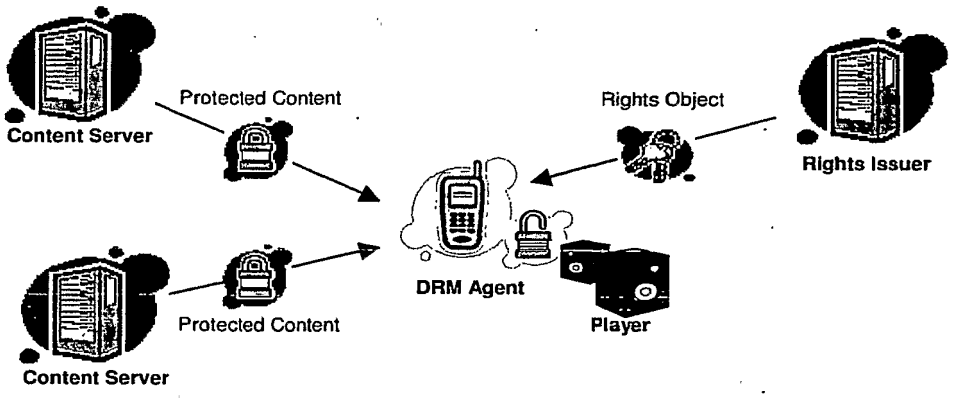


Diagram: DRM Overview

A Rights Object governs how DRM Content may be used. It is a document specifying permissions and constraints associated with a piece of DRM Content. DRM Content cannot be used without an associated Rights Object, and may only be used according to the permissions and constraints specified in a Rights Object.

Like all OMA v2 systems, MusicStation DRM makes a logical separation of DRM Content from Rights Objects, known as "separate delivery". DRM Content and Rights Objects may be requested separately or together, and they may be delivered separately or at the same time. For example, a user can select a piece of content, pay for it, and receive DRM Content and a Rights Object in the same transaction. Later, if the Rights Object expires, the user can go back and acquire a new Rights Object, without having to download the DRM Content again.

Rights Objects associated with DRM Content have to be enforced at the point of consumption. The DRM Agent, inside of the MusicStation handset application, embodies a trusted component of the application, responsible for enforcing permissions and constraints for DRM Content on the device, controlling access to DRM Content on the device, and so on.

A Rights Object is cryptographically bound to a specific DRM Agent, so only that DRM Agent can access it. DRM Content can only be accessed with a valid Rights Object, and so can be freely distributed. This enables, for example, "super-distribution", as users can freely pass DRM Content between them. To access DRM Content on the new device, a new Rights Object has to be requested and delivered to a DRM Agent on that device.

1.1.1 Protection of Content Objects

The DRM Content Format (DCF) is a secure content package for DRM Content, with its own MIME content type. In addition to the encrypted content it contains additional information, such as content description (original content type, vendor, version, etc.), Rights Issuer URI (a location where a Rights Object may be obtained), and so on. This additional information is not encrypted and may be presented to the user before a Rights Object is retrieved. Only the media content (e.g. music file) is encrypted in the DCF file.

The Content Encryption Key (CEK) needed to unlock DRM Content inside a DCF is contained within the associated Rights Object. Thus it is not possible to access DRM Content without a Rights Object. DRM Content can only be used as specified in a Rights Object. MusicStation DRM includes a mechanism allowing a DRM Agent to verify the integrity of a DCF, protecting against modification of the content by some unauthorised entity.

1.1.2 Protection of Rights Objects

A Rights Object is protected using a Rights Encryption Key (REK). The REK is used to encrypt sensitive parts of the Rights Object, such as the Content Encryption Key. During delivery, the REK is cryptographically bound to the target DRM Agent. In this way only the target DRM Agent can access the Rights Object, and thus the CEK. Rights Objects are therefore inherently safe.

2. MusicStation on the Handset

Irrespective of which provisioning method was used, the MusicStation application ends up being resident on the user's mobile phone handset. Each MusicStation handset application installation is a tailored build of software potentially unique to each different phone model and handset firmware version. The software builds are created and managed by Omnifone's patented Device Adaptive Architecture (DAA) and delivered to the correct handset using Omnifone's sophisticated application provisioning software described below.

2.1 MusicStation Provisioning

Working with the MNO there are two ways that the MusicStation mobile handset application is "provisioned" onto the phone, both of which are dealt with in detail in this chapter. The preferred method for distributing the MusicStation application to a handset is to preload (preinstall) the application on the device before it reaches the end-user. Experience of delivering this type of application has shown that discovery by end-users can be as high as 93% when preloaded in the most desirable manner with a hard-key (music button) on the phone which starts the service. Similar experience in OTA provisioning of this type of application by MNOs has shown success rates (i.e. ratio of successfully connected users to requestors) to be an order of magnitude lower than when the application is preloaded.

2.1.1 Application Preload (Preinstall)

Typically this type of device customisation is done by the handset vendor at the request of the MNO and performed before the handset leaves the vendor's premises. It can also be performed by handset distributors, such as Mobiltron, who have customisation capabilities in the supply-chain or perform the same in cells at the warehousing facilities of the MNO.

Wherever this preload is performed it is supported by Omnifone's Preload Provisioning tool, the Preloader. The Preloader is a network-connected desktop application used by staff at the preloading facility. Access to the Preloader is controlled by a Software License, a userid & password and filtered by a list of authorised IP addresses. Access to the Preloader can be revoked at any time either by user, Software License or by organisation.

The Preloader provides an authorised party with access to the latest and most appropriate MusicStation client software builds. Omnifone can control which software builds any Preloader has access to by vendor and model. The Preloader enables the easy location, download and local storage of the correct MusicStation client software build for integration into the handset customisation tools and processes of the installing party.

In-built into the Preloader is a notification system which can alert installers to the fact that new software builds are available for download.

2.1.2 Over-The-Air Delivery (OTA)

Due to an open OTA API, Omnifone supports a number of touch points and mechanisms by which a user might acquire MusicStation OTA. These include, but are not limited to:

- (MNO) WAP portal request.
- SMS text in request.
- Web based request.
- IVR acquisition.
- Web services link.
- Deep MNO network integration such as when SIM is first seen on network.

Irrespective of request mechanism, the MusicStation application is delivered by offering the end-user a WAP download page either directly inside a WAP portal they are already surfing or via WAP push if the application was requested via another method.

2.1.3 Application Reinstall

The MusicStation application contains the ability to force a full reinstall of the application if so instructed to do so by the Server. In this instance the MusicStation application is re-downloaded using OTA WAP download. If the application reinstall is mandated by the Server the old version of the application will not run.

2.2 MusicStation Handset Application Embedded Metadata

Inside each MusicStation handset application is a set of information and metadata automatically inserted and used for a variety of purposes as described here.

2.2.1 Handset Vendor, Model, Version and Firmware Revision

Every MusicStation handset application is built for a specific handset vendor, model, version and firmware revision combination. Metadata identifying this combination is embedded in every MusicStation application build. As such the Server knows exactly what type of handset configuration each MusicStation application is running on. This is the case even if the Server has never communicated with this particular MusicStation application before.

2.2.2 Software License

Every MusicStation handset application is built specifically for a particular MusicStation Service. To affect this every MusicStation handset application has a "Software License" embedded inside it. The Software License is a 512 bit random number which, when presented to the Server, is used to relate this application instance to a particular MusicStation Service. Each Service is either locked or unlocked, only unlocked Services are usable by end-users.

2.2.3 MusicStation Root CA Certificate

Each MusicStation application has the MusicStation Root CA Certificate embedded within the application. This certificate, described in much more detail in later sections of this document, is used to sign and validate messages sent between the MusicStation handset application and the Server.

2.3 Application Permissions & Signing

J2ME implements a security model which means that certain functions that you would normally expect a software application to have access to (e.g. accessing memory/file systems, or accessing the network) are actually restricted. Clearly an application like MusicStation makes extensive use of such features and as such needs access to these common but security protected features of a handset.

In order to provide the MusicStation application with access to these restricted functions, the application is "signed". The signature and resulting PKI certificate of the signer is stored in the JAD file of the application. When the MusicStation application is run, this signature is examined and the certificate is validated to one of the protected domain root certificates already on the handset for these purposes. If the application is correctly signed the restricted features become available.

The root certificates already on the phone are generally either root certificates from the phone manufacturer, mobile network or certificate authority such as Verisign.

2.4 DRM Pertaining to the MusicStation Application Itself

There are a number of ways in which hackers attempt to break DRM systems. One of these ways is to reverse engineer the software code which implements the DRM. It is for this reason that the MusicStation handset application is always installed using the DRM resident on the phone to protect the software from being removed.

Although advanced DRMs such as OMA v2 are not present on many handsets, OMA v1, which supports the required "forward-lock" content control mechanism, is present on the majority of handsets. Forward-lock does as it suggests, it disables the forwarding or transferring of the content item, in this context the MusicStation handset application, from the phone. Whether MusicStation is preloaded or OTA installed, it is installed as an OMA v1 forward-lock protected file.

To further secure OTA deliveries of the MusicStation application, only OTA requests for application downloads confirmed to issue from the MNO network gateways are supported. This ensures that the application code is only ever downloaded over a particular MNO's mobile Internet to a phone rather than being issued from the general Internet. This is implemented by confirming that the source or routing IP addresses found in the network communications headers and metadata are those gateways stored in the Service database and known to be those of the MNO.

2.5 Preloading Music

Music content can be preloaded on to a phone at the same time as the application is installed. This content is either free for promotion and might not be DRM'd, or it is for purchase and subject to the same DRM as would have been applied if the music were downloaded OTA via MusicStation. Preloaded content enables MusicStation to come out of the box playing.

3. Using MusicStation For The First Time

Before a MusicStation application can be used by its owner it must first connect to the MusicStation Server so that it can be registered with the appropriate MusicStation Service and issued with a Client Certificate (and an associated Client Private Key) so that it may access the DRM protected music content which it downloads. In order to be issued with Rights Objects (containing the access rules and the keys to access the DRM protected content) the MusicStation application must also register with the Rights Issuer, this two-step registration process is described in this chapter.

3.1 MusicStation Service Registration

The first time MusicStation starts it knows that it needs to connect to the MusicStation Server in order to register with a Service and be equipped with a Client Certificate and the Client Private Key so that it may access DRM protected content. In order for the registration to occur the Server needs to be able to uniquely identify the device. The "2-pass" MusicStation Service Registration Protocol is the protocol by which this is achieved. This protocol includes identification of the device and the subscriber followed by the secure transfer of the Client Certificate and the associated Client Private Key from the MusicStation Server (the Certificate Authority) back to the Device. As it is imperative that only this MusicStation Device can access the Client Private Key, the registration protocol uses HTTPS secure communications.

3.1.1 Service Registration Request

The MusicStation application attempts to access the handset's IMEI, Bluetooth Address, IMSI and the subscriber's MSISDN so that it might provide information to the Server to uniquely identify the Device and the user. The request parameters sent to the Server are described in the table below.

Parameter	Req'd?	Description
Version	Yes	A <major.minor> representation of the highest MusicStation protocol version supported by this device.
Software Licence	Yes	The 512 bit number installed on the device relating the application to a particular MusicStation Service.
Application GUID	Yes	The Application GUID is embedded in the MusicStation handset software at build time. It is a unique identifier of the specific build of the MusicStation application particular to the device's manufacturer, model, version and firmware revision level.
IMEI	No†	International Mobile Equipment Identity number, globally unique idenitifer of every mobile phone. Some handsets provide access to the IMEI from Java, where this is the case it is retrieved and sent to the Server as part of the MusicStation Service Registration Request.
Bluetooth Address	No†	The unique address used to identify a Bluetooth device and therefore a globally unique method of identifying the handset. The J2ME Bluetooth API (JSR-82) provides a method to obtain the Bluetooth Address. Where this is supported it is retrieved and sent to the Server as part of the MusicStation Service Registration Request.
IMSI	No†	International Mobile Subscriber Identity number, unique to every mobile phone user, stored in the SIM. The IMSI is available as a system property on some J2ME devices. Where this is supported it is retrieved and sent to the Server as part of the MusicStation Service Registration Request.
MSISDN	No	MSISDN is the full international number associated with a mobile phone starting with the country code. The MSISDN is available as a system property on some J2ME devices. Where this is supported it is retrieved and sent to the Server as part of the MusicStation Service Registration Request.

Table: Service Registration Request Parameters

† One of IMEI, Bluetooth Address or IMSI must be supplied to identify the device or SIM card at the server.

3.1.2 MNO Added Metadata

As communications from the MusicStation handset application to the MusicStation Server are routed through the networking equipment of the MNO the following subscriber and potentially also handset identifiers are added to the HTTP request headers. This information is extracted from these headers and used by the MusicStation Server for added identification purposes.

Metadata Item	Req'd?	Description
MSISDN	No†	As above.
Party ID	No†	Internal MNO unique identification for subscriber. Generally used if MSISDN is deemed too sensitive to place inside communications headers which go outside of the MNO network.
IMEI	No	As above.

Table: MNO Added Metadata Parameters

† One of MSISDN or Party ID must be supplied to identify the subscriber at the server.

3.1.3 Service Registration Process

When the MusicStation Server receives a Service Registration Request message these steps are followed.

3.1.3.1 Registration from MNO Home Network?

When the Server receives a registration request it checks that the mobile data network that the MusicStation handset application is currently being used on is the MNO's home network. This is done using a set of database stored records of the IP addresses of the MNO's home network gateways and Internet traffic routing equipment.

The normal setting is to only allow Device registrations on the MNO's home network or on other specific networks such as that of a third-party MNO with whom there is a roaming agreement.

3.1.3.2 Customer Credentials Verification

Upon receipt of a request to register a new MusicStation handset application with a MusicStation Service the server will perform the following tests:

- Confirm that the Software License is for a valid and active MusicStation Service.
- Confirm that the subscriber has been identified, e.g. by MSISDN or Party ID.
- Confirm that the MSISDN or Party ID is a customer of this MNO (if the API exists at the MNO).
- Optionally confirm that the device has been identified, e.g. by IMEI or Bluetooth ID.

Once these credentials have been confirmed, the server moves on to the PKI stage below.

3.1.3.3 MusicStation & Public Key Infrastructure (PKI)

After a MusicStation Service Registration is successfully completed the Device will need to register with the Rights Issuer so that it may request Rights Objects and in turn access DRM content. The Rights Issuer, however, only registers Devices which it can positively identify. This identification is facilitated by the MusicStation Server acting as a PKI Certificate Authority (CA) and generating a public key certificate, the Client Certificate, for each registered MusicStation handset application and thus attesting to the authenticity and identity of each Device. The MusicStation Rights Issuer trusts the CA, it has a copy of the MusicStation Root CA Certificate so that it can confirm that the Client Certificate presented to it by a MusicStation handset application was actually issued by the CA.

Public Key Infrastructure (PKI) is the arrangement used which provides for trusted third-party vetting of, and vouching for, user identities, or in this context MusicStation handset application identities. It allows the binding of public keys to users. This is usually carried out by software at a central location, in this case the MusicStation Server, together with other coordinated software at distributed locations, i.e. the MusicStation handset applications.

PKI arrangements enable users (MusicStation applications, MusicStation Servers, MusicStation Rights Issuers, etc) to be authenticated, and to use the information in PKI certificates (i.e. each other's public keys) to encrypt and decrypt messages traveling between parties in the system. In general, a PKI consists of client software (MusicStation handset application), server software (MusicStation Server) such as a Certificate Authority and operational procedures. A user may digitally sign messages using his private key, and another user can check that signature (using the public key contained in that user's certificate issued by a CA within the PKI). This enables two (or more) communicating parties to establish confidentiality, message integrity and user authentication without having to exchange any secret information in advance.

The authenticity of the CA's signature, and whether the CA can be trusted, can be determined by examining its certificate. This chain must however end somewhere, and it does so at the MusicStation CA Root Certificate, so called as it is at the root of a tree. Root certificates are implicitly trusted (they are sometimes called the Trust Anchor) and are included with many software applications such as web browsers, or in this case the MusicStation Rights Issuer and the MusicStation handset application.

3.1.3.4 Client Certificate & Client Private Key Generation

The first step in issuing a new Client Certificate is to generate a new public and private key pair for the MusicStation handset application which is registering. This implementation of PKI uses the RSA 1024 bit public key algorithm.

Once the key pair has been generated the public key is used by the MusicStation CA to build, then issue the Client Certificate. The Client Certificate states that the CA attests that the public key contained in the Client Certificate belongs to the MusicStation handset application noted in the certificate. A CA's obligation is to verify an applicant's credentials, so that users (relying parties, such as the MusicStation Rights Issuer) can trust the information in the CA's certificates. The idea is that if the user trusts the CA and can verify the CA's signature, then they can also verify that a certain public key does indeed belong to whomever is identified in the Client Certificate.

The Client Private Key is not stored on the MusicStation Server, only the Client Public Key so that the server can create messages that only this device can open.

The X.509 standard is used for all MusicStation certificates. X.509 is an ITU-T standard for public key infrastructure (PKI). X.509 specifies, amongst other things, standard formats for public key certificates and a certification path validation algorithm.

3.1.3.5 Client GUID

The Client GUID is a unique number (Globally Unique ID) which is generated every time a new MusicStation handset application is registered with the Server. The Client GUID is returned to the MusicStation handset application whereupon it is stored and returned on all subsequent communications and requests to the MusicStation Server or the MusicStation RI.

3.1.4 Service Registration Response

The Service Registration Response message is sent from the CA to the Device in response to a MusicStation Service Registration Request message. It carries the protected Client Certificate and Client Private Key over HTTPS.

Parameter	Req'd?	Description
Status	Yes	Indicates if the MusicStation Registration Request resulted in a successful registration. Values include: • "Success"- registration was successful. • "Access Denied" – the device was not authorised. • "Malformed Request" - the CA failed to parse the Device's request, for example both IMEI and Bluetooth Address were missing from the request. • "UnsupportedVersion" - indicates that the device used a protocol version not supported by the CA.
Client GUID	Yes†	Uniquely identifies the device to the CA or RI and is sent by the device in every request to the MusicStation Server or RI.
Client Certificate	Yes†	The MusicStation handset application's public key certificate signed by the CA using the MusicStation Root CA Certificate.
Client Private Key	Yes†	The RSA 1024 bit private key used by the device to • Decrypt messages that have been encrypted using its public key. • Sign messages that can be validated using its public key.

Table: Service Registration Response Parameters

† Only mandatory if Status = "Success".

3.1.5 Post Service Registration Process

After the results are returned for a successful Service Registration the MusicStation handset application performs the following tasks.

3.1.5.1 Client Certificate Storage

The Client Certificate for the device is stored in the application's record management system (RMS) memory store. RMS in J2ME provides a mechanism through which applications can persistently store data and retrieve it later. In a record-oriented approach, J2ME RMS comprises multiple record stores.

3.1.5.2 Client GUID Storage

The Client GUID is encrypted, scrambled and stored in the application's RMS. This is used in all future requests to the MusicStation Server and MusicStation RI.

3.1.5.3 Client Private Key Storage

The MusicStation handset application uses the J2ME private RMS feature. This means that only the MusicStation application which created the RMS record store has access to it.

MusicStation, however, goes further to ensure the security of the Client Private Key. The MusicStation handset application only stores the Client Private Key after encrypting it as an extra security measure in the unlikely event that RMS becomes compromised. More over the application further obfuscates the Client Private Key using certain techniques prior to and during its storage in RMS.

3.2 Rights Issuer Registration

Immediately after the Device acquires its Client Certificate it will attempt to register with the Rights Issuer (RI). A device must be registered with a MusicStation Service before it can register and obtain Rights Objects from the RI. Successful completion of the RI registration process allows the Device to acquire a Domain Key (DK). The DK is a 128-bit AES symmetric key used to protect the Rights Encryption Keys (REKs) of the Rights Objects delivered to the Device.

The RI Registration Protocol is a complete security information exchange and handshake between the Device and the RI. The RI Registration Response message is sent from the Rights Issuer to the Device in response to a RI Registration Request message. This message completes the Registration protocol, and if successful, enables the Device to establish a RI Context for this RI. The RI Context consists of information that was negotiated with the Rights Issuer, during the 2-pass RI Registration Protocol. This RI Context is necessary for a Device to successfully acquire Rights Objects.

3.2.1 DRM Domains

A Domain is a set of one to many Devices that possess a common Domain Key distributed by a Rights Issuer. Devices in the same Domain can all access the same Domain Rights Objects (RO) and potentially then the music protected by those ROs.

In MusicStation the DRM Domains are network-centric. The RI defines the Domains, manages the Domain Keys, and controls which and how many Devices are included and excluded from the Domain. Typically each MusicStation handset application has its own DK and only one MusicStation Device is in each Domain.

3.2.2 RI Registration Request

The RI Registration Request message is sent from the Device to the Rights Issuer to initiate the 2-pass RI Registration Protocol.

Parameter	Req'd?	Description
Version	Yes	A <major.minor> representation of the highest MusicStation protocol version supported by this device.
Client GUID	Yes	Uniquely identifies the device to the RI and is sent by the device in every request to the RI.
Request Time	Yes	The current time as measured by the Device as the difference, measured in milliseconds, between the current time and midnight, January 1, 1970 UTC.
Client Certificate	Yes	The MusicStation handset application's public key certificate signed by the CA using the MusicStation Root CA Certificate.

Table: MusiStation RI Registration Request Parameters

3.2.3 RI Registration Response

The RI Registration Response message is sent from the Rights Issuer to the Device in response to a RI Registration Request message. When the registration is successful it results in a Domain Key being delivered to the MusicStation handset application. This DK is encrypted using the Client Public Key found in the Client Certificate sent to the RI in the request. This way the DK can be securely transferred to the Device as only the Device has access to its Client Private Key which is needed to decrypt and access the DK.

Parameter	Req'd?	Description
Status	Yes	Status indicates if the RI Registration Request was successful. To succeed the Software Licence must be unlocked. If the RI Registration Request was unsuccessful then one of the following error codes results: • "AccessDenied" - Device is not authorised to contact the RI. • "MalformedRequest" – RI failed to parse the Device's request. • "UnsupportedVersion" – Device used a protocol version not supported by the RI. • "InvalidCertificate" - RI could not verify the signature on the Client Certificate due to the certificate being invalid in some way. • "DomainFull" - no more Devices are allowed to join the Domain. • "DomainAccessDenied" – RI does not allow the Device access to the Domain, or the Client Certificate can not be authorised without more information.
Session ID	Yes†	The session identifier set up by the RI.
RI ID	Yes†	Identifies the RI to the Device. It may be possible for the Device to obtain rights from more than one RI. In this case the RI ID uniquely identifies each RI.
RI URL	Yes†	If the RI Registration Request message was successful then the RI URL parameter indicates the URL that should be stored in the RI Context. The Device uses this URL in later interactions with the RI to send requests for ROs. The RI URL is an absolute identifier. There can be more than one RI URL so that the Device can have a fail-over RO request capability.
Domain Info	Yes†	Carries the Domain Key (encrypted using the Device's 1024 bit RSA Client Public Key) as well as information about the maximum lifetime of the Domain. See Appendix A for more detail on DK encryption.

Table: RI Registration Response Parameters

† Only mandatory if Status = "Success".

3.2.4 Post RI Registration Process

After a successful RI Registration Response is received, MusicStation encrypts and obfuscates the returned Domain Key and stores it in the application's private RMS. The DK is subsequently used by MusicStation to access DK encrypted Rights Encryption Keys (REKs) in order to access sensitive parts of Rights Objects (ROs).

4. Listening to Music

In order to listen to music a MusicStation Device needs both the music file, stored as DRM protected content in the DRM Content Format (DCF), and the RO containing the Content Encryption Key (CEK) to unlock the DRM.

It is possible that at any one time neither the DCF nor the corresponding RO is on the Device. ROs contain URLs for the DCF and DCFs contain the URLs for the RO such that if you have one you can acquire the other. If neither are on the Device then the track listing shown in the MusicStation application also contains the URLs for both the RO and the DCF, so oftentimes both files are requested simultaneously after a track has been located in a search or whilst browsing.

4.1 Rights Object Acquisition

The 2-pass RO Acquisition Protocol is the protocol by which the Device acquires Rights Objects. This protocol includes mutual authentication of the Device & RI, integrity-protected request and delivery of ROs, and the secure transfer of cryptographic keying material necessary to process the RO.

4.1.1 RO Acquisition Request

The RO Acquisition Request message is sent from the Device to the RI to request Rights Objects. This message is the first message of the 2-pass RO Acquisition Protocol.

Parameter	Req'd?	Description
Client GUID	Yes	Uniquely identifies the requesting Device to the RI, sent in every request to the RI.
Domain ID	Yes	Identifies the Domain for which the requested ROs shall be issued.
RI ID	Yes	Identifies the authorising RI.
Request Time	Yes	Measured by the Device as the difference, in milliseconds, between the current time and midnight, January 1, 1970 UTC.
RO Info	Yes	Identifies the requested Rights Object(s). The parameter consists of a (non-empty) set of Rights Object identifiers specifying which ROs are being requested.

Table: MusicStation RO Acquisition Request Parameters

4.1.2 RO Acquisition Response

The RO Acquisition Response message is sent from the RI to the Device in response to a RO Acquisition Request message. It carries the ROs containing the protected Content Encryption Key (CEK) for the music DCF in question.

Parameter	Req'd?	Description
Status	Yes	Status indicates if the RO Acquisition Request was successful. If the request was unsuccessful then one of the following error codes results: • "AccessDenied" - Device is not authorised to contact the RI, for example see Client Certificate Revocation below. • "MalformedRequest" - RI failed to parse the Device's request. • "UnsupportedVersion" - Device used a protocol version not supported by the RI. • "NotFound" - Requested object was not found. • "NotRegistered" - Device tried to contact an RI with which it was not previously registered. • "RightsExpired" - Requested rights are no longer available for this Device. This response code indicates to the device that it should not make further attempts to acquire these rights.
Client GUID	Yes†	Identifies the requesting Device. The value returned here must equal the Client GUID sent by the Device in the RO Acquisition Request message that triggered this response.
RI ID	Yes†	Identifies the RI. The value must equal the RI ID sent by the Device in the preceding RO Acquisition Request message.
Protected ROs	Yes†	The Rights Objects in which sensitive information (such as the CEK) is encrypted using the Rights Encryption Key (REK). The REK is encrypted using the customer's Domain Key.

Table: MusicStation RO Acquisition Response Parameters

† Only mandatory if Status = "Success".

4.1.3 Client Certificate Revocation

Once per Device session the RI checks with the CA that the Device's Client Certificate is still valid. The CA maintains a certificate revocation list (CRL), a list of Client Certificates that have been revoked and should not be relied upon. Whenever a certificate is used it must be checked against this list to check the revocation status. The certificate will be revoked if the CA has improperly issued a certificate, the private key has been compromised, the user has violated the CA's usage policy or the MusicStation administrator has denied access to this Device for any reason.

4.2 Content Download

This section descibes how music content is prepared, protected and downloaded to MusicStation Devices from the MusicStation Content Server.

4.2.1 Content Preparation

Before music content is made available for download from the MusicStation Content Server (CS) it is protected from unauthorised access by encryption. Encrypting a music file creates a new file known as a DRM Content Format (DCF).

In MusicStation music content encryption is performed using a 128-bit RC4 symmetric Content Encryption Key (CEK). Every DCF has a different 128-bit RC4 CEK. Thus if there are 1,000,000 tracks in the music library and each track is available in 10 distinct file formats (to cater for different phone flavours and music capabilities/codecs) there are 10,000,000 distinct CEKs, one per physical file. This means that even if the CEK to one DCF is compromised, no other DCF is compromised as a result.

4.2.2 Content Acquisition Request

Since every DCF is inherently secure, DCFs can be transported using insecure transport protocols. For this reason MusicStation Devices request music content using HTTP.

Parameter	Req'd?	Description
DCF File	Yes	Identifies the DCF (encrypted music file) requested for download.
Client GUID	Yes	Identifies the requesting Device.
Range	No	Used to request part of a file if it has already been partially downloaded.

Table: Content Acquisition Request Parameters

4.2.3 Content Acquisition Response

The response from the MusicStation Content Server is typically the binary stream of the DCF file requested over HTTP. This is predominantly the entire file but sometimes the file transfer can be interrupted by broken mobile network coverage. In these situations the MusicStation handset application makes a subsequent Content Acquisition Request but this time, using the Range parameter, it only requests the part of the DCF that it does not already have.

As the DCF byte stream arrives at the Device the MusicStation application progressively writes the file to the handset's file system. Both internal and external (removal media) memory is utilised. When MusicStation's allocation of combined internal & external memory is full, MusicStation removes the track which has not been played for the longest period of time. This is repeated until there is enough space available for the newly requested track.

All music content is stored in the original DCF protected format in which it was downloaded. In order to access the music inside any DCF, the corresponding RO is required so that the CEK may be accessed.

4.3 Playing Music Content

In order to play music through the MusicStation application the music track DCF and the corresponding RO are required to be on the phone. First the RO is examined to see if the user has the right to play the music. If so the CEK is extracted from the RO and used to decrypt the DCF to access the music track which is then played via the phone's media player.

4.3.1 Evaluating the Rights Expression Language (REL)

Once a request is made to play a track, for which the relevant RO & DCF exist on the phone, the Rights Expression Language contained in the RO is parsed by the MusicStation DRM Agent. The REL defines the ways in which the content in the DCF associated with this RO can be consumed and used by the user. The rights expressed by the REL can be very rich, examples include:

- Content is free for unlimited playback.
- Content can be played once then must be bought.
- Content can be played free for one week then must be bought.
- Content can be played free for one month but not more than 5 times.
- Content can be played an unlimited number of times if purchased.
- Content can be played an unlimited number of times if user is currently inside a valid AYCE subscription period.

4.3.2 Decrypting the Content

If the DRM Agent determines from the REL that the user is able to play the music then the 128-bit AES REK is used to gain access to the encrypted CEK for the associated DCF. The 128-bit RC4 CEK is then used to decrypt the DCF to access the original music track. This decrypted track is either stored in non-permanent handset memory for the duration of the track playback or it is progressively delivered in as a decrypted stream to the handset media player depending on the behaviour of the particular phone. Decrypted music tracks are never stored permanently on the handset.

5. All-You-Can-Eat Services

Due to MusicStation's sophisticated DRM implementation it is possible to support advanced content access models such as All-You-Can-Eat (AYCE). This allows users who are within a valid subscription period to have unlimited access to download any track and play each track as often as they like.

5.1 Subscription Periods

MusicStation supports a wide range of subscription periods such as daily, weekly, monthly or any other period required. A subscription period starts when the MusicStation Server communicates with the MNO billing system and successfully charges the user's telephone bill with the appropriate charge for the subscription period.

The MusicStation Server maintains the state of the user's subscription period by recording the date/time of the successful charge made to the user's bill (prepay or postpay) and the length of the subscription period for which the user was charged. This information is shared with the DRM Agent on the MusicStation handset application and thus the Device knows if the user has a valid subscription or not.

5.2 Subscription Expiry

The preferred method for implementing AYCE subscriptions is the rolling subscription method. In this model the MusicStation Server automatically purchases a new subscription when the current subscription period runs out. The user does have the option to cancel a subscription and this causes the automatic re-subscription to be suspended. If the user tries to access content (whether on the Device or not) after the end of the last valid subscription period the user is asked if they want to restart their subscription. If so the rolling subscription starts again.

Where the subscription model is not a rolling subscription, the user needs to confirm that they wish to subscribe for another AYCE period each time the current period lapses.

5.3 Parent Rights Object

A Rights Object may inherit permissions from another Rights Object. This mechanism is used, for example, to specify rights for content acquired as part of an AYCE subscription. The RO that inherits permissions is referred to as a Child Rights Object (C-RO). The Rights Object that contains the permissions that are inherited is referred to as a Parent Rights Object (P-RO).

The Client Devices verify that the same Rights Issuer issued the C-RO and its related P-RO and they both belong to the same Domain before the associated content is made available to the user. The P-RO does not reference any DRM Content directly.

5.4 DRM Time

The DRM Agent on the phone ideally has permanent access to an accurate date/time unchangeable by the user (a DRM time). This is not the case with mobile phones so MusicStation has to use various methods to ensure that a reliable DRM time is available to the DRM Agent so that fair access can be given to users performing legitimate changes to their phone date/time whilst resisting those users who may try to fool the system and gain illegitimate access within a subscription service.

Whilst phones may legitimately have their date/time changed at any point (e.g. first time it has been set, changing time zone or daylight saving) the MusicStation Server always maintains a reliable date/time. So whilst there is a network connection the DRM Agent can always access a reliable date/time datum.

Because the MusicStation Server date/time is potentially different from the Device's local date/time the DRM Agent uses timers relative to the local date/time rather than absolute date/times. However it also monitors the local date/time relative to where it knows the local date/time should be based on the timers it keeps. This allows subscriptions to expire without network connections and also identification of local phone date/time changes. Whenever a network connection occurs all the timers and actual date/time knowledge is reset.

Issues potentially occur when users attempt to fool the system by setting their date/time to some time in the past. These issues are successfully countered using the following logic:

- When the MusicStation handset application starts, it compares the local date/time with the last application close date/time. If the application open date/time is before the last application close date/time then the difference between these two date/times is subtracted from the relative timer.
- The net result is that the longest a determined user can use MusicStation for whilst they have no network connection is an aggregate application usage time equaling the length of time of the paid-for subscription period

(e.g. one week). In order to do this they need to note down the time every time they close MusicStation and reset their phone date/time to that time just before and every time they start MusicStation. Clearly the limited reward does not appear to be worth the effort.

- The date/time is checked once a minute whilst the application is running to protect against a user changing the date/time whilst the application is open. If the current date/time is before the last date/time then the difference between these two date/times is subtracted from the relative timer.

If a user legitimately changes their local date/time to a date/time in the future and the Device can connect to the Server expiry times are re-synced with the Server. If a connection is not available:

- If the future time is still within the valid subscription period then the tracks will play.
- If the future time is greater than the license expiry time then, outside of a system configurable threshold, tracks will not play until the Device connects to the Server. It is not possible to distinguish between the user putting their clock forward (whilst MusicStation is not running) and the user not using MusicStation for a long period of time. A reconnection to the server via the data network is required to reactivate the user or renew the subscription.

5.5 AYCE Accounting

Accounting for AYCE systems requires an absolute count of every play of every track by every end-user. Track consumption information is therefore required to be transferred back to the Server where all qualifying plays by all MusicStation Devices in a particular Service are aggregated together. These aggregated play counts are used to determine what the royalty payments are to each rights owner whose music has been played in the accounting period. Tracks which are played for less than a preview threshold period, such as 30 seconds, are considered to be free previews and are not included in the royalty payments calculations.

So that MusicStation does not cause unnecessary network traffic, play counts are buffered on the Device until a natural network connection is required by the application. This buffering extends also to 0G (flights and tunnels etc) where play counts can be buffered for extended periods of time and sent to the server when a connection is finally made.

5.6 Blended Models

MusicStation provides for a blended commercial model where users who are in AYCE may still make outright purchases of tracks. If the subscription period ends without renewal, those tracks that the user has purchased may still be accessed.

Appendix A: Key Management

This is a description of the cryptographic way in which the Rights Issuer issues a Domain Key (DK) to a Device using the public key known only to the DRM Agent in the MusicStation handset application on the Device. Also described is the way that the RI protects the CEK in the RO by using a REK which it delivers to the Device having first been encrypted using the KD previously delivered to the Device.

Distributing K_D under a Device Public Key

This section applies when provisioning a Device with a Domain Key, K_D.

K_D is the symmetric key-wrapping key used when protecting K_{REK} ("Rights Object Encryption Key") issued to a Domain D. K_D is a 128-bit long AES key generated randomly by the sender and shall be unique for each Domain D. K_{REK} is the wrapping key for the content-encryption key K_{CEK} in Rights Objects.

The asymmetric encryption scheme RSA shall be used to securely transmit K_D to a recipient Device using the Device's RSA Client Public Key.

$$C = RSA.ENCRYPT(ClientPubKey, K_D)$$

After receiving C, the Device decrypts C using its Client Private Key:

$$K_D = RSA.DECRYPT(ClientPrivKey, C)$$

Distributing K_{REK} under a Domain Key K_D

This section applies when protecting a Rights Object for a Domain.

The key-wrapping scheme AES-WRAP shall be used. The RI encrypts K_{REK} using K_D:

$$C = AES\text{-}WRAP(K_D, K_{REK})$$

After receiving C, the Device decrypts C using K_D:

$$K_{REK} = AES\text{-}UNWRAP(K_D, C)$$

MusicStation: Summary of Possible Patents
2 September 2006

User interaction patents can be especially valuable in this space because they give product differentiation and its relatively easy to assess infringement. For the US and UK, it's relatively safe to rely on provisional patent applications that describe these ideas. But for protection in other European countries, it will be important for us to file full patent applications, including claims covering all of the main innovative features, prior to any public disclosure of the interface.

Because the US market is relatively unimportant to you, our recommendation therefore is that you do not defer filing full applications based on your provisional applications for the full 12 months (i.e. 5 May 2007), but instead fill full applications prior to launch (tentatively scheduled for Q1 2007).

We've identified the following concepts:

1. **Tabs**: although tabs are a well known UI device, they may not be known in the context of controlling a music player application running on a mobile telephone. For example, there are no tabs on the iPod: menu navigation in the iPod is hierarchical – you proceed down a menu branch, with a screen associated with one level followed by a screen associated with a different level. Tabs on the other hand are always visible, allowing one to immediately go to the tabbed items; changing context is much quicker than with a conventional hierarchical menu structure. Play/Inbox/Buzz menu items are the tabbed items. They can be selected by pressing an associated number (1 for Play, 2 for Inbox, 3 for Buzz). Tabs take up space, so they are arguably not the natural design choice for a small screen device.

2. **More soft key**; this is context sensitive.

3. **Multi-tasking context sensitive joystick**. Its function is shown by an icon above it (e.g. play or pause). Left means previous track; right means next track. Up means volume up; down means volume down. These functions are replicated by numbers in the keypad – 8 is the joystick; 5 is up; 0 is down; 7 is left and 9 is right.

4. **Context appropriate buy function**: 'Get New Artist' is at same level in the menu as 'Artist'. Get New Track is on the same level in the menu as the menu listing of tracks for an artist. So it's made easier for you to buy since the offer is made to you in the optimal context – when you are most likely to be interested in and open to buying.

5. **'Artist' is on the main menu**, and not 1 level down. (But cf iPod)

6. **Community sync.:** one phone acts as the master and then other phones (e.g. connected over Bluetooth) will playback in sync.

7. **Dedicated 'play' numeric key**: key 4 always toggles back to the play screen.

8. **Variable timeout**: different screens have different timeouts (e.g. never snap back – as used on the search screen; between 2 and 7 seconds – as used on different navigation screens; 20 seconds or more if reading a news item.

9. **Targeted news**: news will be filtered according to a user's playing habits (most CRMs are cognisant only of what you've bought – not what you've actually listened to).

10. **Music data**: we store detailed listening data – how long tracks have been listened to; what tracks are skipped through and when. This is locally cached on the device and then sent back to the Omnifone servers as a piggyback over some comms that happens anyway (will pre-emptively send data back without waiting for a ride only if the user has not downloaded for more than a set time). This data can be used to enrich a music suggestion engine. Content can be targeted using this data (e.g. targeted news etc. – news items will be meta-tagged to enable fast filtering).

11. **Multi-threaded music player** might also be new.

OMNIFONE DISCLOSURES

What follows is an edited typescript of the contents of a meeting.

I had a list of things of the best we could come up with regard to classifying the inventions. At least the inventions we could group, which was user-interface-client related ones and then client-server related ones as well.

Just so I understand, you are a client peforming operations using the handset, client-server things which happen between the handset and the server, and extension, or possible extension, to a device-adaptive architecture.

Other concepts include:

Music Loader - an approach to loading music, which can be content related

Personal Computer (PC) client is our client playing music on a handset, synchronising to a PC.

Intelligent Memory Management is a client end concept. It'd be a user interface (UI) client concept probably.

Do we go beyond music at some point?

Yes, for example with the Media Station concept.

Right, so, let's at least work through each case. Starting at the top and we can... build an understood terminology base with you guys, so that you know what they're about.

So "Background Downloading"

Well let's make sure people understand what "Background Downloading" is. Click on a track, the track automatically goes to the end of the playlist. So there's a few elements in here for Background Downloading. In actual fact, I think that Background Downloading is probably a claim within a slightly wider ultimate concept. There's a more important top level patent which is designing a graphical user interface (GUI) in this specific environment which works for music and supports large degrees of latency. This is one of the particularly difficult things. If we build an interface where the user expects to be able to click on things and things happen immediately, you're going to be disappointed in a lot of cases. Like in India, where connectivity may be poor. So for example, one of the techniques we use is to add to the end of the playlist. That's one of a set of techniques. So the expectation is: "I'm playing music already, and when I click on something

that's out there on the network, it's added to the end of my playlist. So by the time I get to it, because of the Background Downloading, it's already there." So it doesn't matter whether it took 20 seconds, or 7 minutes, or 18 minutes necessarily, it will always ultimately arrive. Often by the time that it comes to be the current one in the current playlist, or what we call The Line-Up.

So this is the notion of supporting a large degree of latency and what you have to do is mask the large degree of latency?

Well sort of, I wouldn't say mask it so much. We support its understanding. Rather than not supporting, not being realistic about the latency of the "click-on-it" and having it lock up the interface, instead, showing a progress mark, it downloads the music and multithreading is part of our system. Multithreading is part of the answer here, because you can't do Background Downloading unless you've got processor capability for downloading, whilst you've also got processor capability for browsing and other searching capabilities. And processor capability for delivering and streaming the music to the media player. So the multithreading capability is part of this.

So for example the other part of the strategy is to show a current playlist or a line-up where you can see in black those which are in the playlist that are there ready to play immediately, whilst in grey you see those which are still on the network or are part-downloaded with a percentage sign to indicate the percent downloaded. And you can see that percentage increment. So you give the user a visibility as to where the download process is.

So you've got a number of different techniques for supporting high latency.

Background Downloading, I think, comes under high latency apps or a high latency UI style grouping.

Client Exception Handling, how the client decides what to do where an exception occurs based on the exception and what the client was doing when the exception occurred. So we've got particular documents to file there. I think ultimately it just gets filed under a general capabilities of the music station client, as it sort of indicates by the group.

Dynamic Playlist Management. Current playlists that show percentage downloaded, not just a list of tracks, and then skips tracks not completed. So that, I would say, is a component of the high latency UI. It might be that out of the high latency UI we want to make claims for things like percentage downloaded on a current downloading stream which isn't necessarily only in the context of a high latency UI, but just shows percentage, those sorts of things.

Native Handset Feature Zone. Solution for allowing java music station to access some of the native phone features, making use of the small native process

running on the handset and the java application communicated by sockets ... this is a solution for handsets.

Java has access to only a set of things, i.e. not necessarily to all things that a native application would have access to. Because of its "sandbox" model, it sits and rides in a sandbox and virtually can't get access to anything else other than that defined by the sandbox. So, for example, java can't access Open Mobile Alliance Version One (OMAV1) objects and there are various bits and bobs it can't do, but if you build a native level service which starts up with the phone and connects to our application through sockets, you can get the native service to do whatever you like. So effectively if a piece of C++ code on the device can access OMAV1 objects, but java can't, our little running bridge - it would be a daemon because it effectively starts up with the machine - presents itself with the communication sockets to say which music station java could communicate through to the communication socket, and therefore we've got a way of talking to something more powerful outside of the sandbox which we can have had included within it things which wouldn't naturally be available to java through the sandbox. It's our way of extending java capabilities available to java without having to re-write the whole of that application within the native language.

Now where that fits into general music station client clever applications, underneath the client exception, would the client exception handle it?

Recommendations Wrapping – get new bin within the list of tracks to allow downloading on the fly.

The description's okay but basically it's like ... to begin with you would just have a list of music that you have, having a separate section where you go to source new music, or something like that. It's basically having the two linked together. If you're looking at your track, called "Blondie" but right there is a recommendation of tracks that you might not have or otherwise or similar tracks.

Which is the point we discussed intermediate last time. I think we described it in your notes as the hierarchy at which it could come into the menu was early, getting you and showing you also what you've got on the phone at the same time. Now it might mean therefore that we might be targeting a ... because this is a specialist UI – music station special UI features – maybe high latency UI is one component of that and "clever access/see what you've currently got", is another aspect of the same clever UI and the next point tabbed GUI might be part of that as well. We definitely need to submit the new tabs GUI.

User interface – the way we create the user interface on the mobile for many different screen size handsets – this is using a standard skinning approach but it is unique in the way that the skin is automatically adapted to work on various screen resolutions. So it's just like a device adaptive application on a comparable visual piece.

So we've probably got a device adaptive architecture (DAA) part 2, is what I would say. Because that's actually back-end but it has an effect on the front-end but the main logic's on the back-end. The build of this is done on the back-end.

Concept of Sending Playlists and Recommended Tracks to Users in Unlimited Download Mode.

I think that's where he's talking about how you can share stuff between two users.

I think what we do here is we go "buzz/community" on the mobile music application. That is almost certainly ultimately its own patent. "Buzz" is what we call the inbox, the community aspect. So we've got a design document for "buzz" which includes ... you can get friends, you can get recommended friends, you can connect with friends, you can ask friends to join their friend network, and you can text people – there's optional swear word filtering. You can send people a playlist, you can send it to anybody, you can share a playlist, your playlists are rated. Playlists are automatically rated within the system based on the number of plays and the number of explicit ratings such as "I hate it" from the user base so you get cooler based on how many of your playlists are consumed. You get play maker charts and you can send ... it's not only an unlimited download mode, which I prefer calling "all you can eat" as opposed to "pay-per-track," but it's much more feature-full because it's much more likely that you will be able to consume the same music, certainly from the same territory. I can create a playlist, send it to you, annotate it and within seconds you can be listening to it.

It's quite an interesting comparison... even an album or any piece of content ... so a friend of mine I know who is into a bit of grungy/metal type stuff and I knew he would like a particular Snow Patrol album. He lives in Singapore. By my telephone or email recommendations it actually took nine months before he got the right album and agreed that he loved it, whereas if he was a user of Music Station a couple of seconds later he could be listening to what I had recommended to him. We ought to add on to that a piece as well, that with Blue Tooth synchronisation such that we can make sure via something, such as Blue Tooth, that for two or more users, they are linked to the same beat at the same time on the same playlist.

We also have a crack at the play charts. We've got the play charts ... should be an item on here.

So the play charts – given that this is an "all you can eat" the ... what that means is that we actually have to monitor who is playing what because "all you can eat" commercially is handled on the back-end based on, if you listen to a track for more than 30 seconds, it's considered a play. If you listen to a track for less than 30 seconds, it's considered a preview. All plays are ... in fact this is ... there's

another piece in here, there's intellectual property rights (IPR) all over the place in here.

There's a data hierarchy, which is another patentable piece. It might be in relation to the M-com stuff. M-com is what we are calling the communications between the client and the server which is particularly clever for a number of reasons that I think we'll get into. M-com is our soon to be patented communications strategy between the client and the server. I was just describing going more into "all you can eat". M-com data synchronisation is our clever way of doing things in this high latency network. So, one of the things for example that, like the "all you can eat" data of how many tracks you can listen to for over 30 seconds and how many tracks haven't been listened to for over 30 seconds and which tracks they are (and also there is other types of data that we collect on the clients). We want to send back these data to the server - Data Protection Act notwithstanding - and this different information has different criticality. For example, if we were to lose a load of audit track data (which was possible in the past), then so we lose CRM data, for data mining as to what areas of the system people are using and in what order, which in itself is quite an interesting capability. The ability to monitor every single key press. Because M-Com also includes the knowledge of everything that is on the client, we know that from the server so we can effectively look at the key presses from the client and thus know exactly where they are in what they are doing. The ability to look at all that stuff, aggregate it, data mine it, use it for product development, etc. is quite significant. But that CRM data stuff is sacrificable because we are talking about resources that can become full and ultimately you need to know whether you can sacrifice stuff or not.

Data that you generate on the client ultimately has a target home with the server. You want to send to the server but you don't necessarily want to create a server request just because you've got some data on the client. So we've got a different priority stack of stuff on the client so when data is created, there's two types of priority. There's "Should this never be lost, or is it sacrificable, and at what priority level should I get it up to the server?" Stuff might happen straight away so as soon as it's generated, bang, send it off to the server. It might have very important ... in which case there would be a small timer basis upon which stuff that we cache ... and then it would go on the back of the next user request. M-Com includes the ability of both ways actually, for a user to actually say "do me a search" we are able to see whether there is anything in the outbox, essentially, that could be tacked on the end of the user request so we don't break additional network round trips. On the server side, it's exactly the same. There's effectively an outbox or an inbox, (depending on which way you choose look at it) for the client. We actually can't create our brand promise in its raw form from the server to the client, we have to wait for an incoming request. Or we can send the client an SMS. There's actually a couple more in there: note down "SMS" and we'll come back to that point. But as far as HTTP because it's going through firewalls and stuff, it has to initiate with the client. Because our server processes can look

at the inboxes of that particular client, it thinks "I've got three new Chinese characters that I need to send out to that device", or "I've got some new news about Bono because that client listens to Bono", stick it in his inbox and then tack that on the end of a normal search when the client uses it.

So, for example, a couple of things about planes and trains; or tunnels. Probably tacked into the background downloading piece we want to add the fact that there's a resumption function eg. if it's a one megabyte file and you have downloaded 300 kbytes and you go into a tunnel and then you get out of the tunnel at the other end, it will pick up from where it left off. So it doesn't go back to the start. It picks up from where it left off. That's minimising the ultimate amount of traffic. The other thing about the nice UI part of that is that any of the network functions which were available outside of the tunnel go to grey, when you're inside the tunnel and effectively the thing becomes a normal digital audio player, with what happens to be your best music on it at the time. That's the reason for that as well as ... probably ... Intelligent Memory Management. This is part of the clever application and may be part of background downloading or a related point, or maybe even intelligent power bar.

Intelligent Memory Management. What this means is effectively we are using the phone as a cache, intelligently. So we are intelligently utilizing its memory. We see what memory is also available on the phone, we see what memory is also available on the removable memory device and we take a view as to what's a sensible proportion of that memory to have and then we go ahead and start using that memory. Any target phone that we find ourselves operating on will be anywhere between 30 songs capability and 5,000 songs capability. So because we run an "all you can eat" download subscription service for whatever our price point is per week, like £1.99 in the UK, you can download as many songs as you like, as often as you like, play as often as you like and there are no further financial implications for you. It's all within the £1.99. So there's no network costs, there are no re-download costs. That only works if you've got intelligent parallel downloading with background downloading and an application that's designed for latency, and the multi-threaded architecture so that you can always play music and there's always something going on whilst that's going on. Then effectively what we've got is the ability, if you've got a handset handling 30 tracks, you click on your 31st one and what the application will do is it keeps an audit trail of which songs that are on your handset are playing when and how often and it will delete enough songs in the least played/longest ago category to free up memory space.

Will this leave enough space for you to download one or two tracks or will it just delete tem, say, when you actually download them and there isn't space for them?

We include both permutations. Effectively that way, it just becomes a buffer for playing and if you've something that can handle 300 songs, theoretically on that

basis, by the time you step into the aeroplane the 300 songs that you have on your phone will be those you want to play most frequently so those which will give you the most enjoyment.

We get to planes and tunnels. The point is when you're on a plane or in a tunnel, (under "Roaming internet protocol (IP) gateways") you generate a data base which is audit trail data and actually music usage data. The music usage data – it will never be deleted. We'll delete a track before we delete our music usage data because it's critical that we report the right numbers to the music labels - so that's never sacrificable. Effectively, you can go into a plane and even though your subscription will know you're subscribed for a week, or for a subscription period, with also a period of grace that's agreed with the music labels, so you can be using the handset in an off-line mode non-connected when it just seems like an iPod effectively and all this usage data will be actually collected and then when you get out of the plane, switch your phone back on and it gets reconnected it can ... (planes make a bad example because the underground would be better) when you get out of the underground then next time when you press something to do a search it will lump together all of the usage data which will eventually be used for the subscription service payments out to the labels, it will lump that onto the end of the search request. So there's intelligent high security caching of usage information buffered until there's a network capability available.

IP Roaming. We'll put this in the client server section under "roaming". What we do on the server: it's important for end users not to wander into a place where the service is zero rated with data, or is low rated. Where we know the server is at, we know the user is either going to be billed nothing in addition for their data with all this user consumption and downloading of tracks or perhaps it's a very, very low rated country and we know that the data rates are so small that they can use it as much as they like and they're not going to get a shocking bill. Eg. if they go out from South Africa and step into Botswana, the roaming agreement for data between one mobile network operator (MNO) and another might be awful, so suddenly they might be from a zero rated situation into running £1,000.00 over a week because they just didn't have the awareness that where they were was consuming a huge amount of data which was no longer zero rated, and was expensively rated. Mobile data is, as you know, a notoriously high cost. We implement a strategy to resolve that situation. In particular there are three ways in which that's done. The first way is looking at the incoming traffic requests from the client and analysing the routing and header information in the communications and looking at the gateways where this communication first popped out onto the network. When we're rolling out with a mobile network operator, we ask them for all their IP gateway addresses which are friendly zero-rated IP's. If we see incoming communications from any of those, we just ignore it, but if the flags are set properly, i.e. watch out for roaming, and we see incoming requests from music station clients from gateways that we don't yet know, what we do is flag the user and say "you may be in an area that is not

zero-rated or not low-rated. Be careful, do you want to switch Music Station to off-line mode?"

Then there's a couple more suffixes on that: it might be as granular as off line or on line, i.e. Music Station works as normal whatever the cost, with just wandering if Music Station goes completely off line. There might be a more granular one than that which is in the event that the subscription period runs out whilst you are abroad and the grace period has run out as well, then we might do a tiny little prompt, which hopefully won't cost anything, which is just to update and confirm one week's more subscription at low cost.

Another way of doing it is the MNO might be able to tell us or there's something else that emanates from the MNO at the time that when the communication comes in, in the header of the comms, (this is a third way of doing it) is that the MNO will know in its internal network that a handset is roaming and provides us with an API that we can on our request make a query into the MNO's home network and say "Is this guy local or roaming?"

Here's another one which is Intelligent Network Selection, which is: let's make sure that M-Com has got two ways of knowing what the quality of signal is. For example there are three types of network it might be faced with: 2.5G, 3G or Wifi. To me there are ... okay there's latency but there are two important aspects of those networks. One is the cost and the other is "What is the bandwidth?" Unless it's zero rated, actually, you've got ... even then there is a cost to the MNO as well - even if it is marginal - so 2.5G and 3G are low bandwidth, and high bandwidth respectively, both high cost, whereas Wifi is high bandwidth, low cost. Although Wifi is a pig to actually get deployed and currently have access to all the access points, and maybe WiMax is a solution, but the point is what we're talking about here is an active application that knows from the meta data that it has available to it whether it's on a high cost or low cost network of low or high bandwidth. For example, there's a particular problem with virtual mobile network operators who don't have such freedom as the real mobile network operators, zero rate data - they've got to wake up and buy into data reckoning, so an application like this might work. Interestingly enough all the intelligent work you do with an application like Music Station, like select that, select that, select that and build a playlist, it's low bandwidth. All the high bandwidth stuff is low intelligence: big lump of data gets communicated. If you are implementing the network intelligence into the client, you can do all the high intelligence, high feature user functions on a high-cost, low bandwidth or a high-cost high bandwidth network, and as soon as you walk past eg. Starbucks a big fat content goes "wallop" into your phone, or when you walk into the home. That will without doubt be a feature for many, many products of this type. Taking advantage of waiting to get the high bandwidth content for when you're ready for it, but having that scheduled and organised via a high cost lower bandwidth connection.

Starbucks have a Wifi network, and you will get your music while you are buying your drink. It will draw customers into Starbucks or similar shops.

The notable thing about that is that most people have very similar habits to what they did the day before. You could pick up the trend in terms of when they expect to be there... for a lot of people who go the same route to work every day ...

DL Roaming. We've got a very detailed DL Roaming implementation for the client and it's modelled after OMAV2. OMAV2 is a standard, yet to be formalized: Open Mobile Alliance Version Two, for Digital Rights Management.

Right, client server. Client data synchronisation. How the server sends data uploads to the clients. I'm not sure whether this comes under M-Com, I'm inclined that it does. Client data synchronisation – to facilitate this type of application, you need to know what data the client has got. The way we implement is that the server ... (I think this includes both the inbox/outbox concepts for tacking on the back of communications which is naturally going both ways) ... plus also the knowledge of at the server end exactly what the client has in terms of resources.

Client login – how will you record what happened on the client and how we send it to the server. What do you know about that?

I think that's exactly what you were talking about in terms of ... you know, what the client has done.

Yes, so there's ... because that might actually be extended in that client login document to cover exception-handling, to cover faults which have occurred on the client side. If we've got faults which occur on the client side, we capture the exception, the audit trail of what led up to that exception, package it up, stick it in the outbox and eventually it gets sent up to the server. We've got an interesting piece of kit on the server called the Incident Monitor. It's a useful utility, I don't think we need to go too far into it. What the Incident Monitor does is it's basically like an inbox where you can throw any incident at it whatsoever and incidents have types. Different types of incident have different people who are interested, and different rules for escalation. In actual fact our incident monitor ought to be part integrated into the framework of our third level of customer support escalation. It just means one can say "I deal with java client bugs," so one can register interest in any incident that occurs anywhere in the network in relation to these types of bugs or another incident: it could be someone listening to Pavarotti and Steve could say "I am interested in people who listened to Pavarotti 10 times, as well," so it's quite flexible. That's how client login works. Then it allows us to track down bugs and reliability issues on the client remote.

The other thing, and I'm not sure where it's explicitly stated, is the remote controllability of the client. The client is architectured in such a way, you can

imagine that a button press on a screen, it makes the screen do an action associated with a button press. Actually, logically what we have is an abstraction layer between the button press and the action doing it so we have effectively an action layer that a button can be associated with, so what that gives us is the ability to switch off buttons and instead send from the server a stream of commands that goes through the action layer that works as if it were button presses. Also if you think about the return so that the priority of the data that we are being sent, that's being sent back, so we have a kind of audit trail which is losable information, very low priority and is sacrificable. Over the air we can tag a message to the client which says next time you are in communication with a client, deliver a message which says the audit trail is now high priority real time. Any action on the client would then immediately cause a communication up to the server because now audit trails are now high priority real time.

This means that given an audit trail out of a client in real time, you could ... and the other part I just described ... you could effectively have the following situation. Customer has a problem with Music Station and say for ease picks up their landline and phones the call centre. We could then do a screen pop based on their calling number, their MSISDN, and pop up in our interface at the call centre an interface which already as the call centre person pickx up the phone displays the rest of the client's record so they immediately see the client record which relates to this client. Then we've got a simulator which we are currently building which is effectively a little music station implementation (we're doing it in FLASH) and we can send a stream of things like key presses and screen decorations which allow them to completely replicate and interpret and behave exactly like any handset. If we go real time on the client, what we can actually see is the telephone communication and the call centre person says "click on inbox" and instead they click on "play", the call centre person will see the error on her screen because we get the audit trail come up to something that can render the specifics of this user when it goes to play and she says "no, I said click on inbox" and then the user can click correctly.

Or to bring about a "click on inbox," the call centre person can click an icon which says "take control". We then send a message down to the client which disables all key presses so there's no confusion and then the call centre person can click on the screen in front of her, click on the inbox and that will change the inbox which will send a message down to the client which effectively now simulates the key press on that inbox and effectively remote controls the client.

Device for Separate Media Delivery. That's just like delivering media audio or video link, specific to the device, that will play back that media. With audio, encoding may be a different audio encoder. This is a new angle. This would be dependent on what codex the device could play. We should probably go for the one which is based on meta data that we store about the phone, we know what media types each phone can take and we know information about the codex, etc. and we know whether it can handle subtleties within the media type like sampling

rate. This would be dependent on the bandwidth, but perhaps our more intelligent take on this is based on two factors. One is what quality of network we've got and what storage capability have I got on my device as well as the bog-standard stuff, what can I play anyway? Should I get a nice fat file down or should I have a nice thin slender one? It's a quality choice, real time being made by the application so it won't take 32 kilobytes per second EAC plus file down if it's only got 50 song capability in it's RAM and it's on a 2.5G network, but the same phone on a 3G network with 2 MEG, memory, might go for a 48 kilobytes per second. So it's a service level decision, not merely physical capabilities. I think that's probably the "clever music station" concept. With video there is the consideration of the screen size of then device. I think it's going to be highly desirable, that type of bog-standard basic physical attribute differentiated content.

How we prioritise what the client should download next. This is one of the most important aspects of the client side. I think this comes under client M-Com communications. So, for example, we are looking at devices with limited capabilities. It's not like a multi-processing PC with bags of memory, big fat broadband connection, connects to the internet and you go download that and download that, download that and watch a load of things happening and then expect to be able to use Outlook. We are in a situation where we've got maybe only one communications capability, maybe two. If you start using two heavily then what happens in the GUI? And also what happens in the GUI if I've just gone into a news article and clicked on it, which I've only got the headline and image for, I now am requesting it because I'm clicking on it, I'm requesting the news article, so how does it know that the bandwidth is all blocked up with dowloading and putting something else on. You need intelligence on the client to know that certain actions suddenly take priority over something else so one has "progressive downloads" - next time that the download responds and it's packet size - stop what it's doing there and then give the communications chance to request that piece of news that they are requesting. Prefer that, and then back onto downloading.

Music Station Transfer Protocol (MSTP) is equivalent to M-Com, I guess.

Chinese characters is an interesting one. When you're working with limited devices, you don't want to bloat those devices. When you're working on limited devices you don't want to bloat those devices so when you get to a character set that isn't able to be represented normally ... because we represent things in images, so each one's got a few bits of device relating to it ... we have to generally when we're opening roaming character sets there's a file of 200 to 300 characters so maybe we've got two font sizes, so we've got at least one set, perhaps two or more sets of those in different font sizes, in certain images, so when you move to a traditional character set that's got 3,000 or another Taiwanese, Chinese or Japanese that's got 7,000 characters your approach can't be to put 7,000 times 1, 2 or 3 sets of images on the client because it just bloats it out i.e. it uses up far too much memory or bandwidth. The way you have to

deal with this, or one of the ways to deal with this in this limited device thing, is particularly when you know what's on the client, which we do, you pre-load a set of high use characters, and certainly those which you are using in the context of the initial application, and then as you deliver the new content down to the client, on the fly you check what character set is currently on the client and if the content which you are delivering to the client includes new characters which the client doesn't have, then you package those characters which you need for rendering that object on the same data packet down to the client.

Device Adaptive Architecture part two. Basically this is the delivery of high complex client applications, it's got GUI's, multiple threading, background downloading, digital rights management, encryption, decryption, security, Public Key encryption (PKI), coms, multimedia ... it's the way of authenticating clients. It's a massively complex file compared to Java, and all the platforms are different. With Java, you build one application it will run on one handset: you can imagine you've got one handset guaranteed, it might run two handsets if you're lucky. Every device is a new platform to all intents and purposes. The difference between one platform and another is variable, but the number of concerns over which differences can occur is plethoric, so it's the first time in computing ever that such significant applications need to be built for so many platforms and behave in the same way by having, effectively, a unique built software that fits itself around a platform.

Basically we get a handset in and we've got a model of the world of java on the server, it sends a test application onto the handset, pummels it from the inside out, tries to see the shape of the platform, the java platform, and reports all that structure back to the server whereupon the server offloads a test application specific for that structure, it goes back onto the handset and then calls that structure and tests the behaviour of every node inside that structure. So the first one defines the structure, the second one defines the behaviour of the structure and then all that goes back into the back end and is stored against the handset and then the device adaptive architecture engine reads the metadata then turns to a software component bank and cherry picks, based upon the metadata, all the components it needs to build an application specific to this handset. The net result is that in 50-60% of cases we can have no engineering input.

The first time we see a handset, this is done. This has got nothing to do with end-user use. The end-user will never use our application unless we commission it specifically for that handset, so they will only ever get something that works. The previous description was a commissioning step.

Client bill system. Our entire client bill system which is wrapped around the device adaptive architecture could be patented. You see, to me, it might already be patented because we did deal previously with building applications.

Cross-selling: pressing eg. the right hand button where a track is playing, allowing the user to purchase a related item. So that cross-selling I would say is heavily related to UI, it goes to the start and then is related to somewhere like tabbed, you know, tabbed GUI. It's one of our interface features of the music station and what it means is the ability to have context-sensitive selling of other media items, so you can be listening to "Burning" from The Wailers and there might be the album cover or another Bob Marley and The Wailers different type of media type there available there to buy, or the ability to cross-sell and link, and link other multimedia asset types consumable on the mobile phone. I'm not necessarily concerned that there might be the physical CD as well. Other consumable items cross-related by artist, album or track or even the genre. So what we haven't got here is collaborative filtering and our suggestions and also searching.

So, searching: we've got a big fat searching document. The searching document will explain itself but in essence searching for music is interesting. So there's like seven layers of stripping things down, like stripping out vowels since they don't mean anything. They can only add to errors. The search term and the action entered by the user and the actual term ... the actual character set that is used for in the search are vastly different in some cases, like for example "TOO, TO, a numerical 2, TWO" would go down to perhaps a "TO" or would transform down to a "2". S's and Z's go to S's and these sorts of ... lots and lots of steps like this, phonetical and all sorts of things such that the end search term is hugely more likely to deal with people who don't know how to spell.

The recommendation stuff that you're going to see is ... it shows a two stage recommendation process. One is a network surface without momentum, i.e. no active set of users yet at the point at which you launch it, and the one is with momentum. So for the one without momentum we use static meta data from "people like me" and other sources on the web where we've got a list of eg. twelve artists that are like the other artist, and for each artist we might have meta data which says these are their top albums. For each album we might have meta data of which are the most important tracks on an album. Based on somebody playing one thing, we can suggest another eg. twelve artists that they might like as an artist, or another eg. twelve albums that they might like. So this is the static list of followers and "these artists are related to these artists" is the way we deal with a zero momentum network. When you've got users on there who are using it, we use a nearest neighbour technology which we've built into our recommendations platform and what that does is ... one of it's main features ... because it can recommend most object types to most people, so it can recommend you artists, albums, tracks or play lists, or even news as well, so the news is targeted based on what groups you listen to. So we have a content management system both at headquarters and locally as well where you can publish news content, which is meta data up by the genre and the artist and then only the people who are interested in that artist get that news content. The nearest neighbour ... the way the nearest neighbour stuff works is we ... based

upon what people are actually listening to – the records back on the server - we find a set of people, the nearest neighbours who are the people who listen to the stuff most similar to the stuff you listen to, and these are the nearest neighbours and then we aggregate all of their listening habits and look at the top stuff that they have listened to - that your nearest neighbours have listened to - that you have not yet listened to, and then promote that back to you in terms of eg. artists, albums or tracks that you might like. So that is basically our recommendation structure.

Concept of sending play list and recommended tracks to users in unlimited download mode. Offering users freebie subscription on the handset which automatically goes through to weekly. Now this is interesting. A method of generating an annuity revenue for a mobile subscriber. Basically, what we do is at the back-end we've got billing capability for each MNO. Also at the back end, we've got subscription management, so we know what type of subscriber this is, eg. a weekly mobile-only "all you can eat" subscriber, and the billing information is related to that subscriber so we know when we attempted to take a week's worth of bill and when that transaction came back positive and it went onto the user's bill, therefore we know what the week is they've got an active subscription for. What we do is on the backend (this is configurable for the server as well) but on the backend we have on the server, right at the point where the subscription is due for renewal, we automatically fire off another message on the server to give the end user a seamless and rolling subscription. All this is flexible by territory. Nearly every MNO that we're rolling out with, just does a rolling subscription, so the user starts up and the first time they ever start up the music station, they have to actively subscribe. It says "Would you like to get some free tracks on the phone?" and perhaps they have a week's free subscription to get them into the "all you can eat". At the start of that, they can play the tracks that are on the phone by default, but then when they click on something and it says "would you like to have a week's free access to the world's music?" and "click yes and this will be downloaded in approximately three minutes", so they click "yes," go to a week's free subscription and then actually, unless they de-subscribe, then we've got a rolling subscription which goes on out the backend as it were, implemented by the server, basically knowing, looking at the subscription duration on the backend and generating itself an audit queue of which subscriptions are up for renewal.

In the context of mobile music consumer on the phone using a mobile network operator's billing system: on the handset where the backend is an MNO's billing system ... it's quite valuable.

One touch download and play. With one button press the user can currently select a track to put at the end of the current play list and it will be automatically downloaded in the background without further action by the user. That's quite a good sentence to put in the latent GUI part where we discuss all that because it is a nice one touch feature.

How we record everything the client ever did and send it to the server for storage. This is a method to enable a seamless dynamic for automatic reporting occurring on a mobile device to a server data store.

Cross Selling. Well, you can write a soft key so you can buy a CD. There's a nice idea: buying CD's off the back of the music you're listening to is quite a funky idea. What you need is a physical CD fulfilment pass with a ... we've got a payment mechanism. We may or may not have an APIM to the MNO to get the current user's address to ease the address entry, but we've got a keyboard in any case, so we've got billing 8 kinds of CD fulfilment, we've got the ISRC international numbers for identifying a physical product, which we know they're listening to. So essentially at least after the first-time use where you confirm your address, you're listening to something, do you still look at the publication "buy a house"? Press "Yes." It will be there in two days.

We can include also the ring tones, the wallpapers and other things that you can link by meta data into either the artist, the track or the album or whatever object you're listening to.

Methods to enable an always-responsive user interface on a device which is performing several tasks and power management. So this is about a multi-threading architecture on the client, UI download, decryption, buffer usage, play, report data to server, confirm received data from the server: it relies on a particular threaded architecture on the client side, to enable processing autonomy. So you can create separate threads which do different things. But you need to have them responsive to each other eg. "I'll back off because you need a bit of space. No, you back off, I need some space." It's not just multi-threading, it's multi-threading and having intelligently shared resources.

PC Tools including data driven objects. So now with "PC Tools" we're now talking about our internal tools here: GUI, content management, where we've got a window on our data on our system because our systems are heavily meta data driven. So, for example, our systems are so meta data driven that the majority of a completely new service can be defined in the data, thus massively decreasing the time it takes to create a new incidence of the service. So, for example, if there are a million pieces of data which describe to a service how to behave and how to render itself, but the only thing different between Vodafone Malta and Vodafone UK were the support centre telephone number, then we then have to add one new bit of data and duplicate perhaps the other million bits representing the UK, and bang, we've got a server set up for Malta.

Right, so PC Tools which we have built comprise data driven objects which enable the quick build of dynamic tabs of hierarchically designed objects. The concept relies on the fact that the tabs register an object. Hence an object can be placed on any tab there, an object is selected: dynamic tabs shows both the

things it contains and those which reference it or created it. This allows deep data mining to occur without having to repeatedly re-open other objects or perform other searches. Call this a Method to Enable the Deep Analysis of Hierarchical Data Structures with the Dynamic Creation of Object Tabs.

Relating to the above, a method to enable the automatic local caching of hierarchical data retrieved from the server data store, thus reducing to a minimum the required server hits. So this is a critical part of the M-Com. The PC tools include objects which automatically handle the local caching. This method reduces the programming required to a minimum, particularly suited to over-the-web style database-centric PC style applications.

I mean, now those two things together, there is competitive advantage in this in the sense that ... these tools are deployable out to partners and with this type of architecture it allows us to deliver what looked like client/server local stuff that built out of things that look like VB rather than web pages, over the web.

New CD. The design is to allow the user to decide to purchase a physical CD version of any play list. Now this is a variation on what I've said before, which is rather than us actually recognising that it's that particular partner of that whatsit list, Mark wants to allow the ability to allow the user to build a play list and have that physically blown for you on a CD by a dynamic CD fulfilment company and have that sent to your house. The originator of the play list will receive a commission for every CD on his play list which is pressed. There are issues to resolve: just in time, one CD only, pressing costs, licence agreements. It's an interesting aside to stuff alongside the physical products.

Advertising on the client side. This uses user-based real-time pop up adverts, targeted based on CRM data and meta data, providing offerings. So for example the user has listened to U2 music label, inform them that a U2 new album is available - target all the people who have listened to U2 more than eg. twice and send the message down to the client which results in what appears to be a real-time pop-up of an advert which they can click on, which will take them to a news story which promotes and covers that new U2 release with a link in to the album where they can go and click and play it.

This is a method of deploying software in such a sense that if the software were to be finished and completed and had gone through at least a second step then it would be subject to a licence fee due to the fact that it was being covered by a patent. So for example if the patent says "run object a, if successful run object b, if successful run object c," (but one didn't have the ability to deploy software over the air i.e. so the software has to be in there from day one) but you want to avoid the patent fee or the licence fee and you want to have a non-infringing product, then this is a method by which you abstract the actual software from the sequencing of software components together such that you can deploy components a, b and c but because they have no logical relationship between

them, you are not forming a "if successful run b, if successful run c." Only at the time when you get a connected client which for example you might then confirm there is a commercial relationship, like being able to bill them i.e. you know you are now making money, and therefore you don't mind paying licence fees, to send down at that time the meta data to the little engine which can link a to b, if a was successful, to c if b was successful. Is that enough of a picture?

What is the specific fact situation you're covering?

Trying to avoid paying licence fees on it if the client's not being billed. We want to deploy the software but not link the sequencing, not have the sequencing logic down there.

OMNIfONE™

Proposed Patents

7 February 2007

Contents:

1. UI/Client

	Group		Case	Abstract	Specific Documents	Source
A	High latency UI	1	Multi-threading	A method to enable an always responsive user interface on a device which is performing several tasks in parallel* (ui, download, decrypt, buffer/play, report data to server, receive data from server). This relies upon the particular threaded architecture	Previous submission?	Phil Sant
		2	Background downloading	i.e. click on a track, auto goes to end of playlist and downloads in background... "one-touch-download-and-play"		Rob Lewis
		3	Dynamic playlist management	Current Playlist that show % downloaded (not just a list of tracks) and then skips tracks not completed	Not sure?	Rob Lewis
B	General client capabilities	1	Client exception handling	How the client decides what to do when an exception occurs based on the exception and what the client was doing when the exception occurred	G:\Projects\MusicStation\Architecture & Design\Client\Client Exception Handling.v2-0.doc	Mark Sullivan
			Inbound SMS			
		2	Native handset features app	Potential solution for allowing Java MusicStation to access some of the native phone features making use of a small native process running on the handset and the Java app communicating via sockets with that app.	Native Phone API Bridge See attached email ~ Sandy's summary and MK's description+waiting for MK response	Steve Pocock
		3	Recommedations wrapping	Get New> being within the list of tracks, artists, etc to allow downloading on-the-fly (same playlists etc)	Technical: G:\Projects\MusicStation\Architecture &Design\Recommendation\MusicStation_Recommendations_Ratings_v1-1.doc whitepaper: G:\Projects\MusicStation\Marketing\Whitepapers\MusicStation_Search_Whitepaper_v1-2.pdf	Rob Lewis
		4	Search function	The ability to Search for tracks, artists,calbums, etc from the client	Technical:G:\Projects\MusicStation\Technical Investigation\Categorisation & Search\MusicStation_Search_Definition_v1-4.doc whitepaper:G:\Projects\MusicStation\Marketing\Whitepapers\MusicStation_Recommendations_Whitepaper_v1-2.pdf	
		5	Chinese Characters	Preload high use characters only, package with data as necessary	Not sure?	
		6	Cross-selling	Cross-selling (we don't do this yet but the right-hand-button when a track is playing allowing a user to purchase a related item. E,g. A truetones, wallpaper, video. Ring-back-tone. "one-touch-cross-product-buy". We allow the user to decide to purchase a physical CD version of any playlist. He can select or provide through the modile phone artwork which is printed on the CD cover, with his declared information	Mark has designed how to achieve it - need to get from him	
C	Buzz/ community	1	Send to friend	Concept of sending playlists and recommended tracks to users in Unlimited Download mode (not sure?)	Not sure?	Rob Lewis
		2	Bluetooth synch	?btn 2 users	Not sure?	
		3	Play charts	?Playmaker and playlist	Not sure?	
D	Previous?	1	Tabbed GUI	Play, Inbox, Buzz		Richard Monday

General documents:

1. G:\Projects\MusicStation\UI\musicstation.client.overview Jan07 new tabs.ppt

2. mCom/Client-Server

	Group		Case	Abstract	Specific Documents	Source
A	mCom	1	Client data synchronisation	How the server we sends data updates to the client. The essential point is "a method which allows the seemless dynamic automatic updating of a mobile device from a changing server data store".	G:\Projects\MusicStation\Architecture & Design\Client-Server Communications\Client Data Synchronization.v2-0.doc This document needs to be updated with the latest design for how we send data from the client to the server (e.g. user created playlist changes).	Mark Sullivan/ Mark Knight
		2	Intelligent memory management	Deleting of tracks which haven't been played for a long time to maximise use of memory	Have a paragraph from Mark Sull to attach	Rob Lewis
		3	Incomplete	pick up again?		
		4	Client logging	How we record everything the client ever did and send it to the server for storage. This is "a method to enable the seemless dynamic automatic reporting of events occurring on a mobile device to a server data store	G:\Projects\MusicStation\Architecture & Design\Client-Server Communications\Client Logging.doc	Mark Sullivan
		5	Data Objects	the basic unit of object that is passed between the server and client and client and server. They encapsulate the representation of some entity which is displayed within the client interface (such as an artist, album, etc) or data which needs to be sent	G:\Projects\MusicStation\Architecture & Design\Client-Server Communications\Client Data Synchronization.v2-0.doc Have paragraph from Backward Compatibility doc to attach.	Steve Pocock
		6	Intelligent Parallel Downloader	This concerns the heuristics used to determine (a) which track to download next and (b) which track to play next in a playlist which has not been fully download. These rules are a critically important part of the design which enable the user to have a seemless always-playing music experience. We might call this "a method to ensure a continuous music listening experience of downloaded content on a device with limited bandwidth and memory	G:\Projects\MusicStation\Architecture & Design\Client-Server Communications\Intelligent Parallel Downloader.doc	Mark Sullivan/ Mark Knight
B	General client capabilities?	1	Device specific media delivery	A system for delivering media (audio or video) encoded in a way specific for the abilities of the device that will playback that media. With audio; The encoding may be a different audio codec (MP3, AAC, eAAC+, etc). This would be dependant upon what codecs the device can play. The encoding may be at a different bit rate to produce a smaller file. This would be dependant upon the bandwidth. With video (not that we do this yet), there is also the consideration of the screen size of the device. i.e. The video may be sized 128x128 for one device, whereas the same video can be delivered in 640x480 for another device.	Have description to attach from CK + SP Need an image of the db design for music format association with devices as well. SP may have to provide.	Mike Lamb
C	Previous	1	MSTP (MusicStation Transport Protocol)	but pretty sure that was in previous submissions	G:\Projects\MusicStation\Architecture & Design\Client-Server Communications\Connected MusicStation Protocol DesignV3.doc	Steve Pocock

3. Client Build System

	Group		Case	Abstract	Specific Documents	Source
A	Overall	1	Client build system	Our entire client build system which is wrapped around the DA architecture could be patented		Steve Pocock
B	DAA	1	Device Adaptive Architecture - review	Device Adaptive Architecture which we have previously defined covers a lot of items. I would suggest that it were reviewed to determine whether anything has been come up since then.	Previous?	Steve Pocock
	DAA?	2	Detective		Ask CK if he has any documents describing how Detective works, db design for detective etc If not then Detective is generally covered in the Build/DAA docs listed below- possibly all outdated as CK doesn't have new stuff	Steve Pocock
	DAA?	3	User Interface - Skinning	The way we create the user-interface on the mobile for many different sized handsets. This is using a standard "skinning" approach, but it unique in the way that the "skin" is automatically adapted to work on various screen resolutions.	Have a description from Mike to attach.	Mike Lamb
	DAA?	4	Multiple language support	How we support multiple languages in a client build:	G:\Projects\MusicStation\Architecture & Design\Build and Release\Client Builds and Multi-Language Support.v1-1.doc	Mark Sullivan

General Documents:

1. G:\Projects\MusicStation\Architecture & Design\Build and Release (?Which doc: Building and Provisioning MS Clients 1-1
2. G:\Projects\MusicStation\Handset Commissioning\Commissioning Process\MusicStation Handset Compatibility Rules v4.doc
3. G:\Projects\MusicStation\Handset Commissioning\Commissioning Process\MusicStation Handset Commissioning v3.doc

4. Content Ingestion

	Group		Case	Abstract	Specific Documents
A	MusicLoader	1	?	MusicLoader's approach to loading music and metadata	\\Werejamming\group\Projects\MusicStation\MusicLoader\Design Ask Alex West and get an overview design diagram and notes from him describing the MusicLoader approach. MK will edit them to a suitable format

Source

Steve Pocock

5. PC Client

	Group		Case	Abstract	Specific Documents	Source
A	Sychronisation	1	Playlist synchronisation	Syncing of playlists, favourite tracks, news etc over mobile network to server and then over broadband to a desktop companion product	Have email from CE describing this to attach	Rob Lewis

6. PC Tools

	Group	Case		Abstract	Specific Documents
A		1	?	The PC Tools which we have built comprise data-driven objects which enable the quick build of dynamic tabs of hierarhically organised objects. The concept relies on the fact that the tabs register an object and callbacks about how to expand it. Hence an object can be placed on any tab. When an object is selected, dynamic tabs of shows both the things it contains and those which reference it are created. This allows deep data-mining to occur without having to repeatedly re-open other objects or perform other searches. We can call this "a method to enable the deep analysis of hierarchical data structures through the dynamic creation of a object tabs	Talk to AJ which can provide more design information. I will edit the notes he provides you if required.
		2	?	Related to the above, "a method to enable the automatic local cacheing of hierarchal data retrieved from a server data store thus reducing to a minmum the required server hits". The PC Tools include objects which automatically handle the local caching where the data so cached may be in a 1-1, 1-M, 1-1-M, 1-M-1, 1-M-M etc. relationships. To implement such cacheing from scratch is a very complex task indeed and this method reduces the programmatic work required to achieve this to a minimum. Particularly suited to over-the-web style database-centric PC applications.	

Source	Mark Knight	Mark Knight

7. mDRM

	Group	Case	Abstract	Specific Documents	Source
A		1		DRM Whitepaper?	

8. Other

	Group		Case	Abstract	Specific Documents
		1	**Intelligent Network Selection**	When roaming the client determines whether the IP gateway corresponds to a list of acceptable gateways to avoid usage while roaming without permission	
		1	**Remote control**	Customer service has the ability to directly control the navigation of the app remotely	
		1	**Service offering**	offering users a free week's sub on the handset which then automatically goes through to weekly?	

Source
Phil Sant
Phil Sant
Phil Sant

Suggested patents

Group	Case	Abstract	Notes	
UI/Client - high latency UI	Background downloading	i.e. click on a track, auto goes to end of playlist and downloads in background... "one-touch-download-and-play"	High latency UI	\\Werejamming\group\Projects\MusicStation\UI\musicstation.client.overview Jan07 new tabs.ppt
	Client exception handling	How the client decides what to do when an exception occurs based on the exception and what the client was doing when the exception occurred	General client capabilities	
	Inbound SMS			
	Dynamic playlist management	Current Playlist that show % downloaded (not just a list of tracks) and then skips tracks not completed	High latency UI	
	Native handset features app	Potential solution for allowing Java MusicStation to access some of the native phone features making use of a small native process running on the handset and the Java app communicating via sockets with that app. This is potentially a solution for Moto b	General client capabilities	
	Recommedations wrapping	Get New> being within the list of tracks to allow downloading on-the-fly (same playlists etc)	General client capabilities	
	Tabbed GUI	Play, Inbox, Buzz	Previous	
	Chinese Characters	Preload high use characters only, package with data as necessary	General client capabilities	
	Send to friend	Concept of sending playlists and recommended tracks to users in Unlimited Download mode (not sure?)	Buzz/community	
	Multi-threading	A method to enable an always responsive user interface on a device which is performing several tasks in parallel* (ui, download, decrypt, buffer/play, report data to server, receive data from server). This relies upon the particular threaded architecture	High latency UI	
	Bluetooth synch	?btn 2 users	Buzz/community	
	Play charts	?Playmaker and playlist	Buzz/community	

Client-Server	Client data synchronisation	How the server we sends data updates to the client. This document needs to be updated with the latest design for how we send data from the client to the server (e.g. user created playlist changes)	mCom	
		How the server... The essential point is "a method which allows the seemless dynamic automatic updating of a mobile device from a changing server data store"	mCom	
	Intelligent memory management	Deleting of tracks which haven't been played for a long time to maximise use of memory	mCom	
	Incomplete	pick up again?	mCom	
	Client logging	How we record what happened on the client and how we send it to the server:	mCom	
		How we record.. We record everything the client ever did and send it to the server for storage. This is "a method to enable the seemless dynamic automatic reporting of events occurring on a mobile device to a server data store	mCom	
	Data Objects	the basic unit of object that is passed between the server and client and client and server. They encapsulate the representation of some entity which is displayed within the client interface (such as an artist, album, etc) or data which needs to be sent	mCom	
	Device specific media delivery	A system for delivering media (audio or video) encoded in a way specific for the abilities of the device that will playback that media.	General client capabilities?	
		With audio; The encoding may be a different audio codec (MP3, AAC, eAAC+, etc). This would be dependant upon what codecs the device can play. The encoding may be at a different bit rate to produce a smaller file. This would be dependant upon the bandwidth		
		With video (not that we do this yet), there is also the consideration of the screen size of the device. i.e. The video may be sized 128x128 for one device, whereas the same video can be delivered in 640x480 for another device		
	Intelligent Parallel Downloader	How we prioritise what the client should download next - this is one of the most important aspects of the client design:	mCom	
		How we prioritise... This concerns the heuristics used to determine (a) which track to download next and (b) which track to play next in a playlist which has not been fully download. These rules are a critically important part of the design which enable t		
	MSTP (MusicStation Transport Protocol)	but pretty sure that was in previous submissions		

Build / DAA	Client build system	Our entire client build system which is wrapped around the DA architecture could be patented	\\Werejamming\group\Projects\MusicStation\Architecture & Design\Build and Release \\Werejamming\group\Projects\MusicStation\Handset Commissioning\Commissioning Process\MusicStation Handset Compatibility Rules v4.doc
	Detective		
	Device Adaptive Architecture - review	Device Adaptive Architecture which we have previously defined covers a lot of items. I would suggest that it were reviewed to determine whether anything has been come up since then	
	User Interface - Skinning	The way we create the user-interface on the mobile for many different sized handsets. This is using a standard "skinning" approach, but it unique in the way that the "skin" is automatically adapted to work on various screen resolutions.	
	Multiple language support	How we support multiple languages in a client build:	

PC Client	Playlist synchronisation	Syncing of playlists, favourite tracks, news etc over mobile network to server and then over broadband to a desktop companion product	Have email describing to attach from CE
Other	Cross-selling	Cross-selling (we don't do this yet but the right-hand-button when a track is playing allowing a user to purchase a related item. E,g. A truetones, wallpaper, video, Ring-back-tone)	
		Cross-selling... The point here is that with one button press the user can purchase a related (cross-sold) product to the one currently selected. "one-touch-cross-product-buy". If we can patent our design for the one-click cross-selling feature for mobile We don't do this yet but I have designed how to achieve it. We allow the user to decide to purchase a physical CD version of any playlist. He can select or provide through the modile phone artwork which is printed on the CD cover, with his declared inform	
	?	Concept of sending playlists and recommended tracks to users in Unlimited Download mode (not sure?)	
	?	offering users a free week's sub on the handset which then automatically goes through to weekly?	

Documentation	Source
	Rob Lewis
G:\Projects\MusicStation\Architecture & Design\Client\Client Exception Handling.v2-0.doc	Mark Sullivan
	Rob Lewis
Native Phone API Bridge See attached email – Sandy's summary and MK's description	Steve Pocock
	Rob Lewis
	Richard Monday
	Rob Lewis

G:\Projects\MusicStation\Architecture & Design\Client-Server Communications\Client Data Synchronization.v2-0.doc	Mark Sullivan

Need a paragraph from Mark Sull or he may have it covered in some doc already	Rob Lewis
G:\Projects\MusicStation\Architecture & Design\Client-Server Communications\Client Logging.doc	Mark Sullivan

\\Werejamming\Group\Projects\MusicStation\Architecture & Design\Client-Server Communications\Client Data Synchronization.v2-0.doc	Steve Pocock
Ask CK if he has some words written on how we decide which music format to select for a device You need an image of the db design for music format association with devices as well. I may have to provide	Mike Lamb
G:\Projects\MusicStation\Architecture & Design\Client-Server Communications\Intelligent Parallel Downloader.doc	Mark Sullivan

\\Werejamming\group\Projects\MusicStation\Architecture & Design\Client-Server Communications\Connected MusicStation Protocol DesignV3.doc	Steve Pocock

	Steve Pocock
Ask CK if he has any documents describing how Detective works, db design for detective etc If not then Detective is generally covered in the Build/DAA docs listed above	Steve Pocock
	Steve Pocock
need a paragraph or two from Mike	Mike Lamb
G:\Projects\MusicStation\Architecture & Design\Build and Release\Client Builds and Multi-Language Support.v1-1.doc	Mark Sullivan

	Rob Lewis
	Rob Lewis

	Rob Lewis
This is UI but it has not yet got into the UI spec.	Rob Lewis

Document made available under the Patent Cooperation Treaty (PCT)

International application number: PCT/GB2007/001675

International filing date: 08 May 2007 (08.05.2007)

Document type: Certified copy of priority document

Document details: Country/Office: GB
Number: 0608934.6
Filing date: 05 May 2006 (05.05.2006)

Date of receipt at the International Bureau: 09 July 2007 (09.07.2007)

Remark: Priority document submitted or transmitted to the International Bureau in compliance with Rule 17.1(a) or (b)

World Intellectual Property Organization (WIPO) - Geneva, Switzerland
Organisation Mondiale de la Propriété Intellectuelle (OMPI) - Genève, Suisse

UK Intellectual Property Office

For Creativity and Innovation

Concept House
Cardiff Road
Newport
South Wales
NP10 8QQ

I, the undersigned, being an officer duly authorised in accordance with Section 74(1) and (4) of the Deregulation & Contracting Out Act 1994, to sign and issue certificates on behalf of the Comptroller-General, hereby certify that annexed hereto is a true copy of the documents as originally filed in connection with patent application GB0608934.6 filed on 5 May 2006.

In accordance with the Patents (Companies Re-registration) Rules 1982, if a company named in this certificate and any accompanying documents has re-registered under the Companies Act 1980 with the same name as that with which it was registered immediately before re-registration save for the substitution as, or inclusion as, the last part of the name of the words "public limited company" or their equivalents in Welsh, references to the name of the company in this certificate and any accompanying documents shall be treated as references to the name with which it is so re-registered.

In accordance with the rules, the words "public limited company" may be replaced by p.l.c., plc, P.L.C. or PLC.

Re-registration under the Companies Act does not constitute a new legal entity but merely subjects the company to certain additional company law rules.

Signed

Dated 25 June 2007

J193660/ 006 D10092 P01/77NOFEE....
08MAY06 0.00 NONE 0608934.6

For official use only

THE PATENT OFFICE
MC
- 5 MAY 2006
LONDON

- 5 MAY 2006

Your reference **Introduction (UK)**

The **Patent Office**

Request for grant of a Patent

Form 1/77

Patents Act 1977

1 Title of invention

Introducing MusicStation

0608934.6

2. Applicant's details

[X] First or only applicant

2a If applying as a corporate body: Corporate Name

Omnifone Limited

Country

GB

2b If applying as an individual or partnership

Surname

Forenames

2c Address **The Foundry**
7 Glenthorne Mews
London

UK Postcode **W6 0LJ**

Country **GB**

ADP Number 8997785001

4 Reference Number

Introduction (UK)

5 Claiming an earlier application date

An earlier filing date is claimed:

Yes ☐ No ☒

Number of earlier application or patent number

Filing date

15 (4) (Divisional)	8(3)	12(6)	37(4)
☐	☐	☐	☐

6 Declaration of priority

Country of filing	Priority Application Number	Filing Date

7 Inventorship

The applicant(s) are the sole inventors/joint inventors

Yes ☐ No ☒

Introducing MusicStation

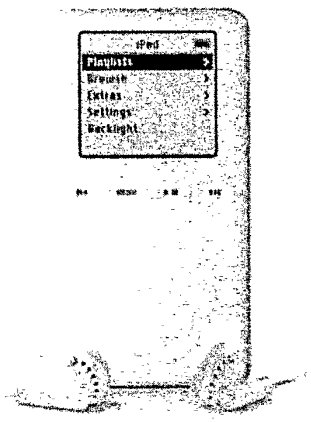
OMNIFONE™

Introducing Omnifone

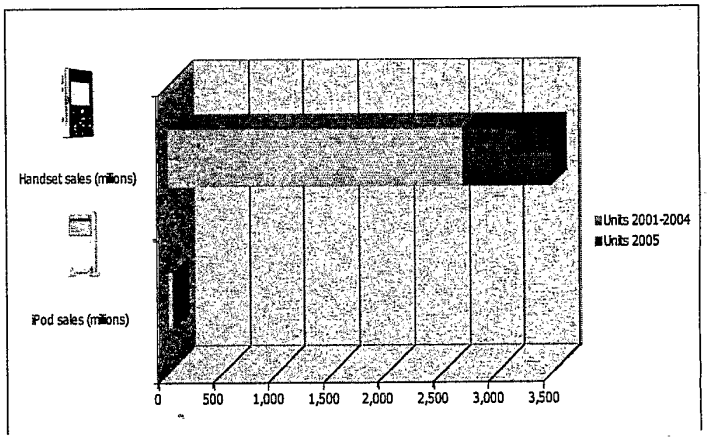
- UK private company founded in 2003 by successful entrepreneurs Rob Lewis, Mark Knight and Phil Sant.
- Previously founded/ran European Internet developer Cromwell Media (sold *March 2000, £850m*).
- Previously founded & ran European tech news service Silicon.com (*sold to NASDAQ-listed publisher CNET*).
- Omnifone is a leader in the development of mobile handset-based user-centric network applications.
- Founders are shareholders in Opera Telecom and Mikoishi.

OMNIFONE™

The mobile music market

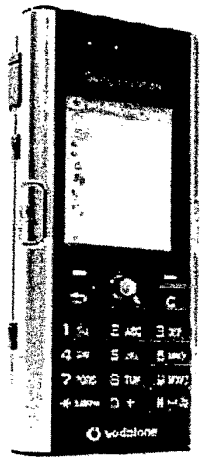


- Apple's iPod (with iTunes) has been a revolutionary device, bringing on-the-move digital music to millions of consumers.
- However, the iPod's market reach is tiny compared with the mobile phone market.
- iPod vs Mobile:
  - *c50m iPods sold since 2001.*
  - *c810m mobile handsets sold in 2005 alone.*
- Analysts expect circa 950 million devices to be sold this year globally.



OMNIfONE™

Why the handset?



The key reasons why Omnifone believes the mobile phone will become the primary Digital Music Player of the future…

1. Almost-Always Connected
   - … *enabling access to music directly from the device on-the-move*
   - … *no need for a connected PC, wires, credit card or broadband access*
   - … *networked community features on-the-move*

2. One touch billing
   - … *No credit cards required*

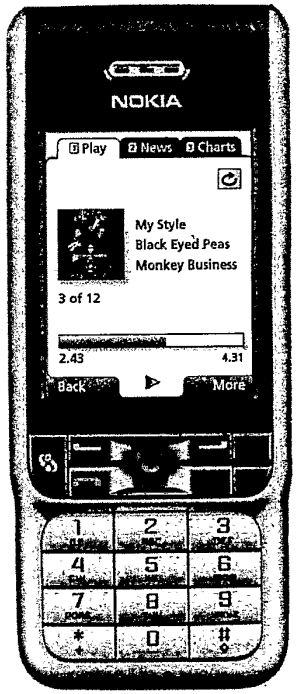
3. The one device that's with the user 24x7

4. It's the Future
   - … *Majority of handsets will sport digital music capability by Q1 2007*

OMNIfONE™

Introducing MusicStation

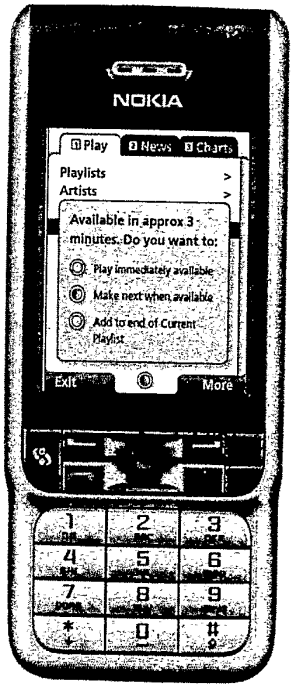
- Omnifone's handset-based dedicated digital music application optimised for pre/post install by Mobile Network Operators (MNOs).
- Will turn vast majority of handsets sold in second half of 2006 and beyond into Super-iPods, with far more functionality than Apple currently delivers.
- For the first time, all the playing functionality and user experience quality of an iPod... *PLUS all the browse, search, celeb playlist features of iTunes (on a PC connected to the internet)... PLUS killer tribal community features... PLUS news and views...*
- All above features available on the move.



OMNIFONE™

MusicStation features

- Super-cool intuitive interface, offering a brilliant user-experience.
- All features available during music play.
- Unique Intelligent Parallel Downloading technology - *with intelligent caching of favourite/owned content.*
- Unique "wrap" of all music play/playlist/content acquisition/news functionality into one clean user interface.
- One touch purchase/subscription uptake with Direct Billing integration with operator.
- User experience personalised both by the provision of explicit user preferences and automatically through a user's implicit application usage.
- Advanced, standards-based, Digital Rights Management (DRM).



OMNIFONE™

MusicStation features

- Three versions of MusicStation:

  1. Monthly subscription plan
  … all the music in the world for a fixed monthly fee

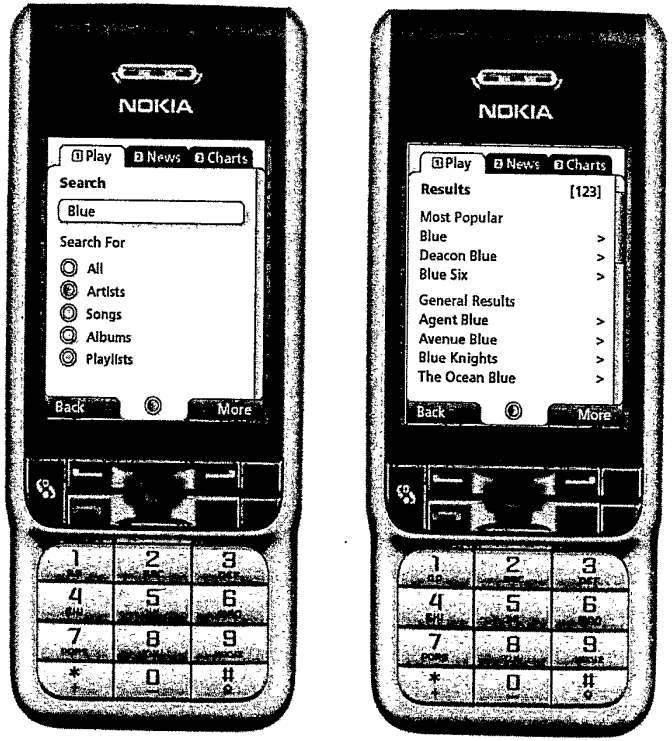
  2. Pay Per Track
  … traditional purchase of album/tracks

  3. Blended version
  … starting with Pay Per Track but allowing consumer to upgrade to subscription

- Operators plan to bundle free or low rate data network usage and will pre-install on millions of devices prior to shipping.

OMNIfONE™

The Key: The Widest handset reach



- Mobile applications generally suffer limited handset reach because of manufacturer variations in handset design/memory capabilities/OS bugs/screen sizes/media file support/Java implementation differentials – *features vary wildly!*
- Omnifone's unique and revolutionary patent-pending Device Adaptive Architecture resolves these problems and enables MusicStation to be available on all music-capable handsets.
- MusicStation will be available on all 2.5G and 3G music-capable handsets. Supports pre-pay and contract.
- Java and Symbian versions at launch, enabling maximum features *and* maximum reach.
- Unlike competitor products, Java is the reference platform which is critical to future handset reach. BREW, Windows Mobile & Linux versions to follow.

OMNIFONE™

MNO Development Partnerships

- Seeing is believing!
- Twelve MNO Development Partners are now working in partnership with Omnifone to create operator-specific versions of MusicStation with localised content, interface and licensing for roll-out on a pre-install basis.
- MNO Development Partners have >500 million subscribers worldwide.
- Partners must commit to provide time to review prototype/betas and supply feedback which will feed into Omnifone's Development Programme.
- Partners get preferential access to market.

OMNIfONE™

Flexible business model

- Flexible business model designed to minimise all obstructions to growth:-
  - *No up-front software license, support costs;*
  - *No bespoke development costs;*
  - *Omnifone can source/manage music licensing or use any existing MNO rights/content.*
- Operator agrees to:
  - *pre-install on all music-capable handsets in the territory with associated application launch key;*
  - *mass-market promotion to consumers.*
- Omnifone shares in the success of MusicStation with its MNO partners through every transaction
  - *be this a Subscription payment or Pay Per Track purchase.*
- Omnifone seeks 9% of transaction value.

Party	Typical Revenue Split	£1.50 per track acquisition	£9 per month subscription
Operator	33%	£0.50	£2.97
Music Industry	50%	£0.75	£4.50
Collection Agencies (*PRS/MCPS*)	8%	£0.12	£0.72
Omnifone	9%	£0.14	£0.81

OMNIFONE™

Management team & Advisors

- Board:
  - *Jim Feeney (Chairman);*
  - *Rob Lewis*
  - *Phil Sant*
  - *Mark Knight*
  - *Steve Pocock (COO);*
- Advisors:
  - *Guy Fletcher OBE (Music Advisory Board Chairman);*
  - *Seth Jackson (Music Advisory Board)*
  - *Peter Langley (Patent/IPR protection)*

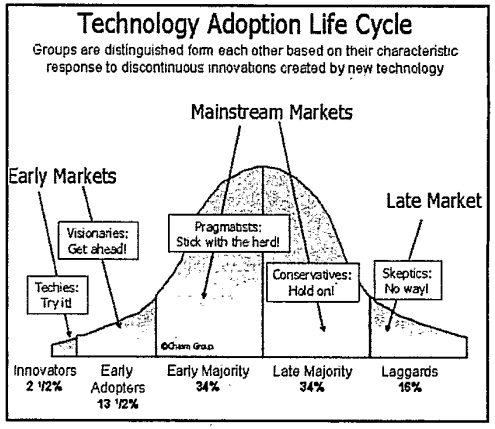
OMNIfONE™

Timetable of key events

May 06	> *Delivery of Demo to MNO Development Partners* > *MNO network integration work commences*
June 06	> *Provision of partner-specific Beta version of MusicStation to Development partners* > *Roll out of user trials in first MNO country areas*
July 06	> *Finalisation of commercial roll out arrangements, localised music licensing arrangements with music partners for early launch MNO partner roll-outs*
Q3 06	> *Roll out with first tranche MNO partners*

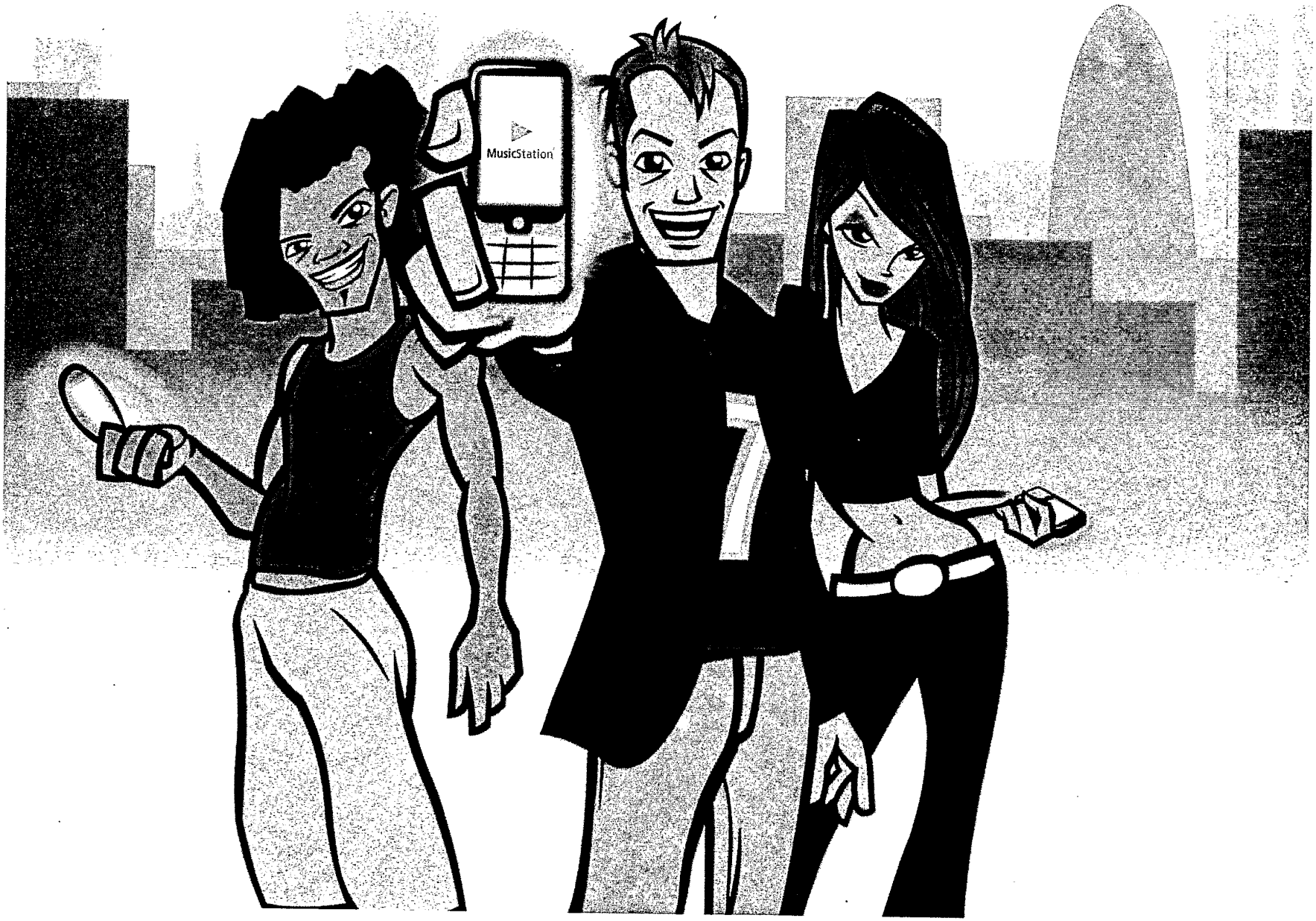
The future for digital music

- What is the future for iPod?
- Significant proportion of consumers want an *always-connected* device for playing/choosing/managing mobile digital music on the move.
- MusicStation offers this across all music capable capable 2.5G and 3G handsets.
- MusicStation delivers a better, richer, more convenient experience than that available on the unconnected iPod or MP3 player.
- The intuitive nature of one-touch billing will take the mobile digital music market from the "Early Adopter" to "Early Majority" and "Late Majority" segments of the tech adoption cycle.
- MusicStation offers key to real network differentiation, increased ARPU and reduction in churn.



OMNIFONE™

"Thank-you"



Document made available under the Patent Cooperation Treaty (PCT)

International application number: PCT/GB2007/001675

International filing date: 08 May 2007 (08.05.2007)

Document type: Certified copy of priority document

Document details: Country/Office: GB
Number: 0608936.1
Filing date: 05 May 2006 (05.05.2006)

Date of receipt at the International Bureau: 12 July 2007 (12.07.2007)

Remark: Priority document submitted or transmitted to the International Bureau in compliance with Rule 17.1(a) or (b)

World Intellectual Property Organization (WIPO) - Geneva, Switzerland
Organisation Mondiale de la Propriété Intellectuelle (OMPI) - Genève, Suisse

PCT/GB 2007/001675

UK Intellectual Property Office
For Creativity and Innovation

Concept House
Cardiff Road
Newport
South Wales
NP10 8QQ

I, the undersigned, being an officer duly authorised in accordance with Section 74(1) and (4) of the Deregulation & Contracting Out Act 1994, to sign and issue certificates on behalf of the Comptroller-General, hereby certify that annexed hereto is a true copy of the documents as originally filed in connection with patent application GB0608936.1 filed on 5 May 2006.

In accordance with the Patents (Companies Re-registration) Rules 1982, if a company named in this certificate and any accompanying documents has re-registered under the Companies Act 1980 with the same name as that with which it was registered immediately before re-registration save for the substitution as, or inclusion as, the last part of the name of the words "public limited company" or their equivalents in Welsh, references to the name of the company in this certificate and any accompanying documents shall be treated as references to the name with which it is so re-registered.

In accordance with the rules, the words "public limited company" may be replaced by p.l.c., plc, P.L.C. or PLC.

Re-registration under the Companies Act does not constitute a new legal entity but merely subjects the company to certain additional company law rules.

Signed

Dated 3 July 2007

For official use only

Your reference **Architect & Design (UK)**

0608936.1

- 5 MAY 2006

The **Patent Office**

Request for grant of a Patent

Form 1/77 Patents Act 1977

1 Title of invention

MusicStation client architecture highlights

2. Applicant's details

[X] First or only applicant

2a If applying as a corporate body: Corporate Name

Omnifone Limited

Country

GB

2b If applying as an individual or partnership

Surname

Forenames

2c Address

The Foundry
7 Glenthorne Mews
London

UK Postcode **W6 0LJ**

Country **GB**

ADP Number 8997785001

4 Reference Number

Architect & Design (UK)

5 Claiming an earlier application date

An earlier filing date is claimed:

Yes ☐ No ☒

Number of earlier application or patent number

Filing date

15 (4) (Divisional)	8(3)	12(6)	37(4)
☐	☐	☐	☐

6 Declaration of priority

Country of filing	Priority Application Number	Filing Date

7 Inventorship

The applicant(s) are the sole inventors/joint inventors

Yes ☐ No ☒

MusicStation Client Architecture Highlights

Mike, Mark (and a little from Steve) – 5 Apr 2006

Table of Contents

1. Introduction

The MusicStation client follows the classic Model-View-Controller model. See the associated high level architecture diagram for an overview.

- Model - `DataObjects` and `DataObjectGroups`
- View - `DataObjectViews`, `DataObjectGroupViews`, and a `CanvasRenderer`.
- Controller - `RunnableCommands`

Additionally the project is split into distinct libraries.

- Canvas User Interface Library
  This is the "Canvas Renderer". It knows nothing about other MusicStation objects and could be used by any MIDlet that wanted to display itself in the MusicStation look and feel.
- MIDP Commons
  This is our implementation of objects and methods that are not provided in J2ME that are provided in J2SE. For example we have our own implementation of the J2SE objects / methods, Comparator, Timer, TimerTask and Collections.sort.
- Shared Objects
  These are our Data Objects that are shared between the Client and the Server. They are in their own library, and know how to hold data, but (for example) have no knowledge about how they would be display.
- Encryption
  This is shared between the Client and the Server, and handles the encryption required for DRM. Again, this is a stand alone library that has no knowledge of any other part of the MusicStation Client project.

The MusicStation MIDlet project builds upon these libraries to provide the required functionality.

2. Model

2.1 Data Objects

The data objects are shared between the client and the server. That is, the server knows how to populate these data objects, and the client knows how to provide views on these objects.

A data object is an abstract object that provides methods that are common to any type of data object. For example, every data object needs to know how to write itself from a stream, and it needs to know how to reconstruct itself from a stream.

Data Objects in the MusicStation prototype system are

- `Article`
- `Genre`
- `Media`
- `Song`

For example, the Song object will extend the abstract Data Object and add functionality for getting the Image associated with this song.

2.2 Data Object Groups

Data Object Groups are simply a collection of `DataObjects`. They are similar to Data Objects, in that they provide functionality that is common to all Groups of `DataObjects`.

Data Object Groups in the MusicStation prototype system are

- `Album` – A Collection of Songs
- `SongGroup` – A Collection of Songs
- `ArticleGroup` – A Collection of Articles
- `AlbumGroup` – A Collection of Albums
- `ArtistGroup` – A Collection of Artists
- `PlaylistGroup` – A collection of Playlists

Common methods that may be available for a Group, include knowing how to read from and write to a stream. And knowing how to sort by name.

3. View

3.1 Data Object Views

`DataObjectViews` allow us to have many views on the same underlying data, without requiring duplicate copies of that `DataObject`.

For example, each `Song` only exists in the System once, but it can be displayed on a User's Playlist, in the Charts, on an Album. Depending upon the View of this Song, different functionality needs to be made available to the user. A Song appearing on a User created playlist needs to have an option to remove it from the playlist, but the same Song appearing in the Chart listing should not have a remove option.

`DataObjectGroupsViews` register to receive events when the `DataObjectGroup` is added to, or an item is deleted from the Group, or when any item is changed within the Group. The View can act upon this event if required.

3.2 Screen Types

The MusicStation Client provides 3 different Screen Types.

The following types of screen exist:

- Form – Used for screens such as Search, About
- List – Used for screens such as main menu, Album screen, etc
- Alert – The popups. Three types currently:
  - Radio button for choice selection (e.g. the purchase this track question)
  - Text input (e.g. used when enter new playlist name)
  - Prompt (e.g. the keyboard lock popups)

DataObjectGroup views will generally render themselves on a List.
ArticleView is an example of a DataObjectView that renders itself on a form.

3.3 View Renderer

The View Render is the component that actually draws the View on the Data to the screen. For example, given an ArticleView, the renderer controls the font that is used, the size of the font, line wrapping, image and text layout and scrolling.

When Rendering a list the View Render controls whether to draw the item in it's selected state, and whether the item needs to be animated to display itself properly.

3.4 Screen Painting

There are several events that cause a screen to be "painted".

- On initial display of the screen
- When user input has changed the current state
- When the DataObjectView fires an event to say the underlying Data has changed
- When animating

The first 3 of these are straightforward, as we simply repaint the screen as required.
All animation is controller by a central animation controller. This is a thread that's sole responsibility is animation.
In general this thread aims to repaint the screen 15 times a seconds (15fps). The "Now Playing" screen only repaints at 4fps because of the track prefetching requiring a greater slice of the processor time.
The Repaint Controller simply asks the current screen whether it wants to be redrawn. And if it does need to be redrawn, then the screen display is refreshed.
On list screens, the only animation we have is the currently selected item scrolling, so in this case, the screen will ask the currently selected item if it requires a repaint. If the item is currently moving, it will ask for a repaint. If it is not animating, then the screen will not be redrawn.

In MusicStation prototype 0.4.1, the Repaint Controller simply asks whether a repaint is required. And if it is, then the whole screen is repainted. In a future release the Repaint Controller will ask for the areas of the screen that need to be repainted, and only update the areas that have changed. This in turn will free up processor cycles to be used by other application functionality.

4. Controller

4.1 Commands

A command is an object that can be run in it's own thread. Every user action corresponds to a Command. For example, adding a track to a playlist, or showing the about screen. The commands create the screens, and if required create the Views of the data to be rendered on those screens.

The views of the Data will be a Task that is added to the Command Queue. This allows the screen to be displayed immediately, and the Data Objects Views to be added to the screen when this data is available.

Commands can optionally add other Commands to the screen that they are about to show.

For Example, the Album screen shows the list of tracks you own on that album. Below the list of tracks we add a BrowseCommand that when selected will change the view to show all Songs on that Album.

4.2 Association of Commands

Commands ultimately appear on the screen (usually on a softkey), but the Commands can come from any place on the flow of Data to that screen.

At the lowest level, we have the **Data Objects** and **Data Object Groups**. These can create Commands that are relevant to all types of Data Object or all Groups. For example, every group may provide a "A-Z sort" and a "Z-A sort".

The next layer up is the **instance of a Data Object**. For example an Album DataObjectGroup may add a command to "Play" the current album. Or to go to "View Artist" associated with this Album. Adding the commands at this level, means that wherever an Album is used in the system, it will always have the Commands "A-Z sort", "Z-A sort", "Play" and "View Artist".

At the next level up is the **View** on the data. This View may add more commands, or remove commands added by a lower level. For example, and A-Z sorted lists of Album tracks may want to remove the "A-Z sort" command, as the list is already sorted.

Screens can also have their own Commands that are associated with the screen, rather than with the items on the screen. Commands like this are commands that do not operate on data, such as "Main Menu".

4.3 Key Presses and the Command Queue

All key presses generate commands, but only if commands are currently present on the screen. Until the commands exist they will be ignored.

All commands are instances of RunnableCommand and must implement the run() method. The Command must ensure that the run method completes quickly. Typically this will just involve putting the next Screen on the Display. Any lengthy operation such as loading data from the file system must be wrapped in a Task. This Task is then put on the CommandQueue. This frees the event thread to redraw the screen and start responding to user input.

Similar to the PlayQueue, there's a single background thread that is used to execute all of the CommandQueue's tasks. A task is typically a method call that loads a data object from storage and is likely to block. We put this operation on a Task and put it on the CommandQueue. We can then show the screen immediately and the screen is populated as the data is loaded. This has the effect of the list growing as the object data is read from the stream. We will be able to use a similar method when loading objects via HTTP.

Once a LoadTask has been started it cannot be cancelled. This is to avoid DataObjects being left in a semi loaded state. At anytime there can only be one LoadTask running and one LoadTask queued to run. If another LoadTask is put on the queue it is assumed the user has decided to navigate to a different screen and the one previously on the queue is cancelled and discarded.

StoreTasks are used to store data objects and are placed on the same queue. This means that load and store operations always run in the same thread avoiding conflicts. StoreTasks are never cancelled.

4.4 Play Queue

There's a single background thread that is used to execute all of the PlayQueue's tasks. A Task is typically a method call on the Player that is likely to block or change the state of the player.

Example Tasks are:
* PrefetchTask - fills the player's buffer
* StartTask - starts the player
* CloseTask - closes the player and releases resources

Each Task can be scheduled to run immediately or after a specified period. If the Task is cancelled before it is executed then it will never run. If it is already being executed it will complete. If it has finished executing then cancelling the Task has no effect

By using a single thread to execute the tasks we avoid locking and synchronization problems between competing threads trying to perform operations on the player.

The current design for parallel playback and prefetching of the next track works like this.

1. The user selects to start a playlist:

CommandThread: A StartTask is placed on the TaskQueue for execution in 1 second. The playback screen is displayed immediatedly.
PlayQueueThread: The StartTask is taken from the TaskQueue and after 1 second the StartTask is executed. After 4 seconds the player has started. A PrefetchTask for the next track is put on the PlayQueue. The PrefetchTask is executed immediately and the next Player's buffer is filled in 4 seonds.

2. The first track finishes:

PlayerThread: On END_OF_MEDIA event a CloseTask for the current song is put on the TaskQueue followed by a StartTask for the next song.
PlayQueueThread: The CloseTask is picked up from the TaskQueue and executed followed by the StartTask for the next song. The song starts immediately becuase it is prefetched. A PrefetchTask for the next track is put on the PlayQueue. The PrefetchTask is executed immediately and the next Player's buffer begins to fill.

3. Within 2 seconds the user skips to the next track:
CommandThread: The current track is stopped. A CloseTask for the current song is put on the TaskQueue followed by a StartTask for the next Song.
PlayQueueThread: The PrefetchTask is currently being executed, after 2 seconds it finishes and the CloseTask is picked up from the TaskQueue and executed followed by the StartTask for the next song. The song starts immediately becuase it is prefetched. A PrefetchTask for the next track is put on the PlayQueue. The PrefetchTask is executed immediately and the next Player's buffer is filled in 4 seconds.

4. Half way through the song the user selects next Song 3 times in a row:

CommandThread: The current track is stopped. A CloseTask for the current song is put on the TaskQueue followed by a StartTask for the next Song scheduled to execute in 1 second. The next track is dsiplayed immediately on the screen.

PlayQueueThread: Picks up the StartTask and schedules it for execution in 1 second.

CommandThread: Second key press occurs within 1 second so the first StartTask is cancelled. StartTask for the next Song scheduled to execute in 1 second. The next track is dsiplayed immediately on the screen.

PlayQueueThread: Cancels first StartTask. Picks up the second StartTask and schedules it for execution in 1 second.

CommandThread: Third key press occurs within 1 second so the second StartTask is cancelled. StartTask for the next Song scheduled to execute in 1 second. The next track is dsiplayed immediately on the screen.

PlayQueueThread: Cancels second StartTask. Picks up the second StartTask and schedules it for execution in 1 second. After 1 second the 3rd Song starts.

5. The user selects the previous track:

a) if the next track has already been prefetched then the current and next track are closed and the previous track is prefetched then started
b) if the next track is half way through being prefetched then it will finish prefetching the next track (because prefetch blocks) before closing the current and next track and prefetching the previous.

Assuming the user will skip to previous again seems like a sensible assumption although at some point into the song we can probably assume they are going to listen to the whole track and then prefetch the next track.

5. On Phone Storage

The data used by the application is currently stored on the phone's memory card. This includes:

- Raw image files in PNG format
- Music files in a format appropriate to the handset
- Data Objects and Data Object Groups.

Full details are provided in **MusicStation Prototype File Mgmt Spec.doc**.

The logic for managing access to this data (managing locks, partial downloads when the user goes into a tunnel, etc) needs further work and any suggestions for an approach from MK gratefully received.

6. Defining Menus and Screens in Data

Currently all menu lists and commands are defined in the client code. There's some opportunity for defining the list of commands in a menu using an XML definition, but these would always need to refer to commands which are already implemented in the client, though of course these commands should be parameterised.

We need to understand in which ways we may want to update the client over the air. As we experienced with MyFone, introducing over the air updates has management knock-on effects for the server, and a clear scoping is important to preventing this from running out of control.

7. Server - Client Connection

The demonstration implementation does not include a server connection. However, the approach will be that the server will populate Data Objects and Data Object Groups on the handset. Suggestions from MK happily received, as this will need to handle connection failure during send etc.

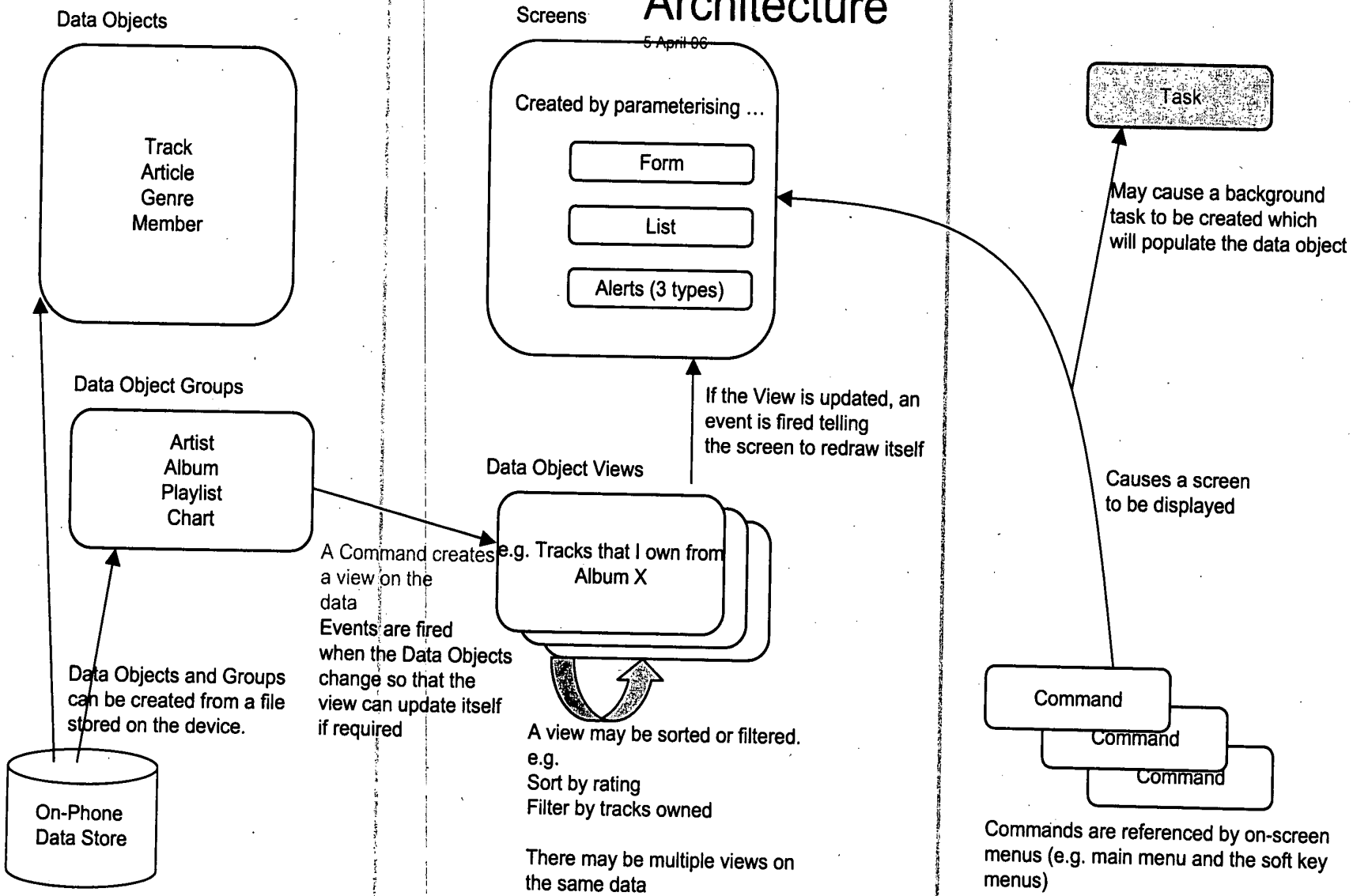
8. Miscellany

The client implementation also includes:

- **Exception handling** - hierarchical mechanism which displays the error message most appropriate to the context
- **Exception logging** – currently logged to RMS but will log to server
- **Device properties** – Separate from the implementation and used to tune the application build to the specific device.
- **Midlet Lifecycle controller** – Controls general lifecycle of the midlet including start-up, exit, pausing when interrupted by phone function, etc
- **Screen resizing** – Rendering the screen to fit the handset's screen size by scaling based on screen height and width
- **Font rendering** – Rendering fonts for all textual content in the interface. The font libraries are directly converted from their vector representations and loaded into the application that renders them as bitmaps.

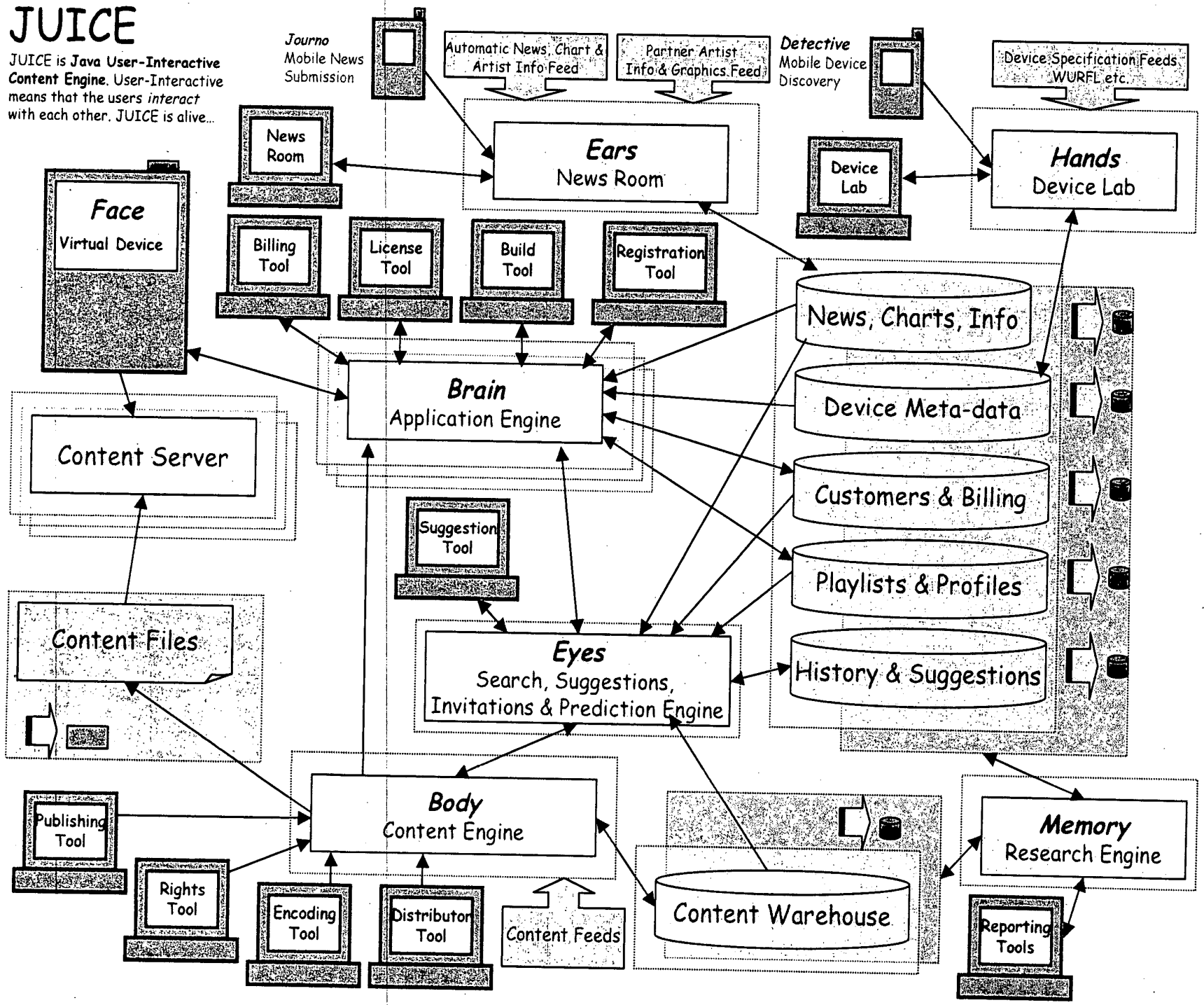
MusicStation Client M-V-C Architecture

5 April 06



JUICE

JUICE is **Java User-Interactive Content Engine**. User-Interactive means that the users *interact* with each other. JUICE is alive...



Main points on the M-DRM:

* There is a Software License embedded in the pre-installed application which is the same for each installation of a particular pre-install deal

* The Software License is checked against the server the first time the application is used in a Registration Request. The IMEI, MAC, BLUETOOTH ID are also then supplied. This information is stored at the server and used to create a Registration-ID which is returned to the client. A Private 1024 RSA key is also returned to the client on successful Registration. The Server only stores the Public part of the key.

* Thereafter the client always supplies only the Registration-ID with each request.

* A Registratiion may be expired by the client or the server.

* A Registratiion may be locked, unlocked or unexpired by the server.

* There are denial of service attack defenses.

* A Software License and all its associared Registrations may be locked or unlocked

* Content can be distributed to any other phone but the receiver must Register to received the Rights Object (not drawn yet).

* Content can be global RAID - i.e. distributed in an intelligent redundant manner (not drawn yet)

* All content is encrypted to it's own AES 128 bit symmetric key - this never changes unless we think the security has been breached

* Content is accessed by requesting Content from the Brain which delivers a Content Rights Object which contains the content rights for one or more pieces of Content. This Rights Object is encrypted with the Subscriber's Public RSA 1024 bit Key.

* The client decrypts the Rights Object with its Private RSA 1024 bit key and downloads the content from the content delivery servers noted in the Rights Object

* Hence no software at all by us is required at the content delivery centre - we only need FTP access

It may look complex but the above is (I believe) OMA 2.0 compliant and quite lightweight to implement at both server and client. No need for complex session handling or on-the-fly encryption is required.

The only question is can we use RSA 1024 bit keys for the assymetric Subscriber Privat/Public pair? If not we just reduce of the key (it's size) but the method remains the same.

We do assume that the Operator partner can append the MSISDN to every HTTP request for our application going through their dateway to our server(s): this isn't critical but would be the nice way to do it. Is this a likely possibility?

Still on drawing board:

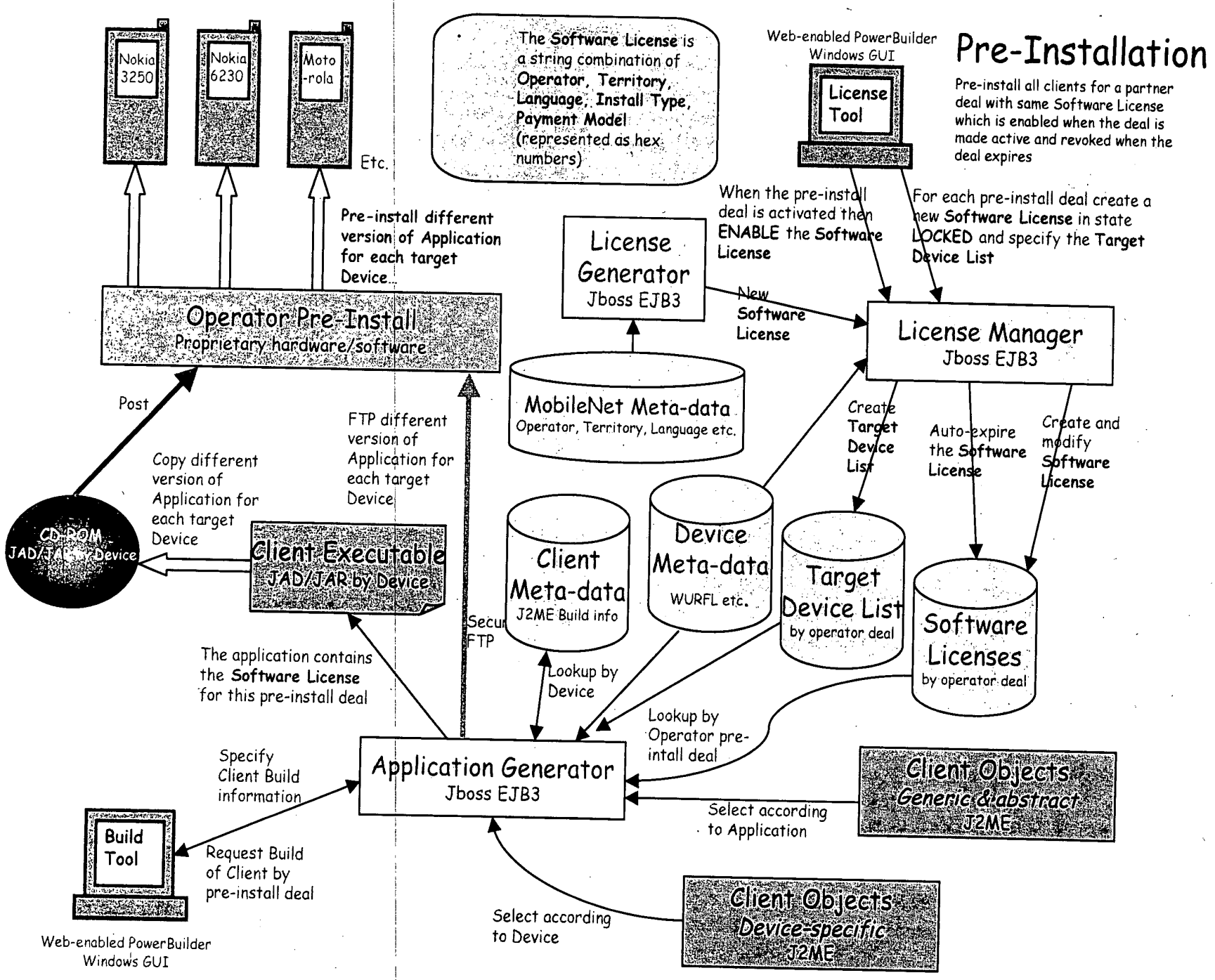
* Client frame design

* Search & Suggestions Server design

* Billing payment mode

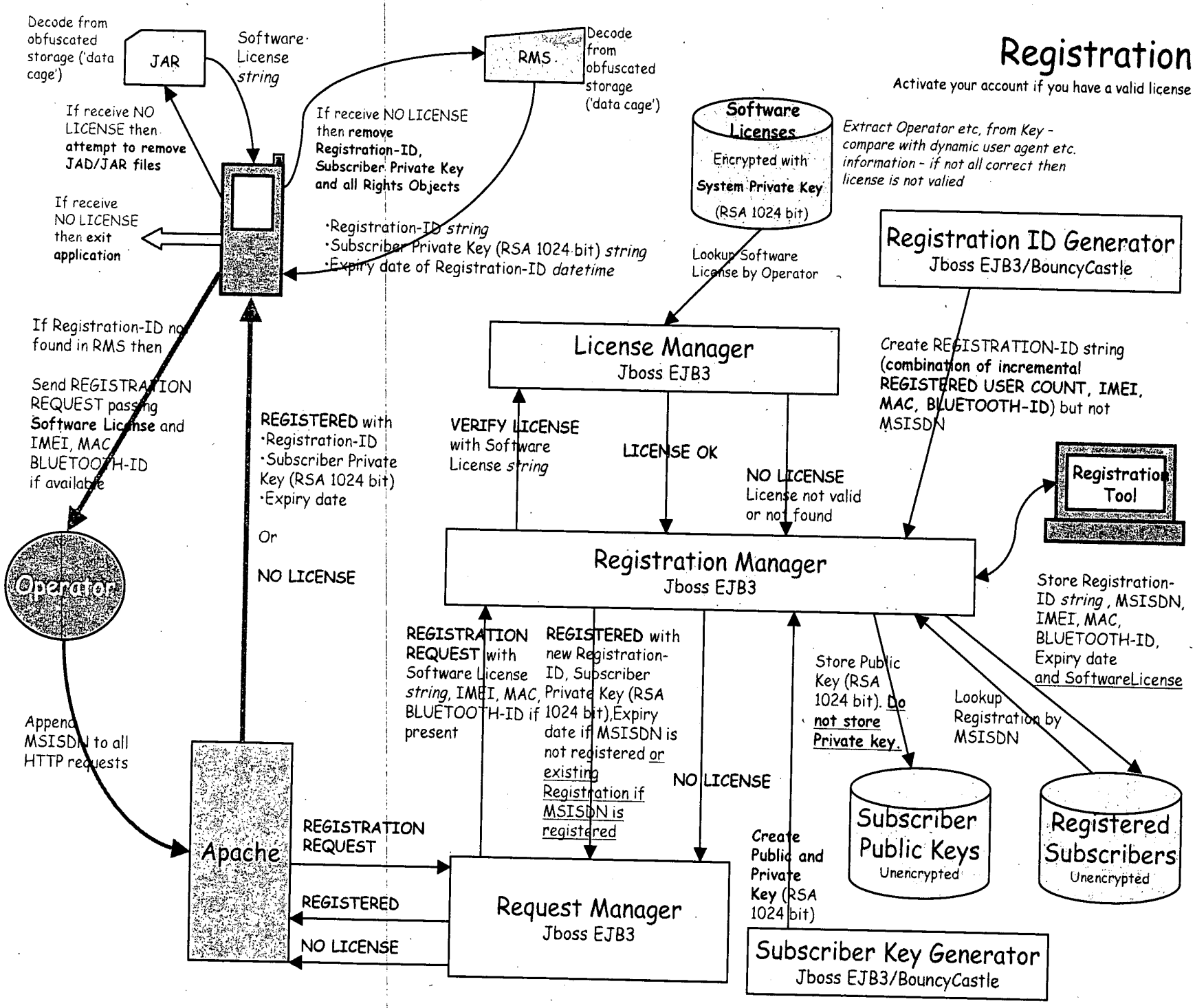
* SIM changing model

The point of all the above is to draw out a workable OMA 2.0 compliant moble architecture. We can use the diagrams as a basis for our shared understanding and the start point to design our perfect M-DRM system which the music industry will also accept. We also will gain a perspective from which to view the integration of any third-party components we may want to use.



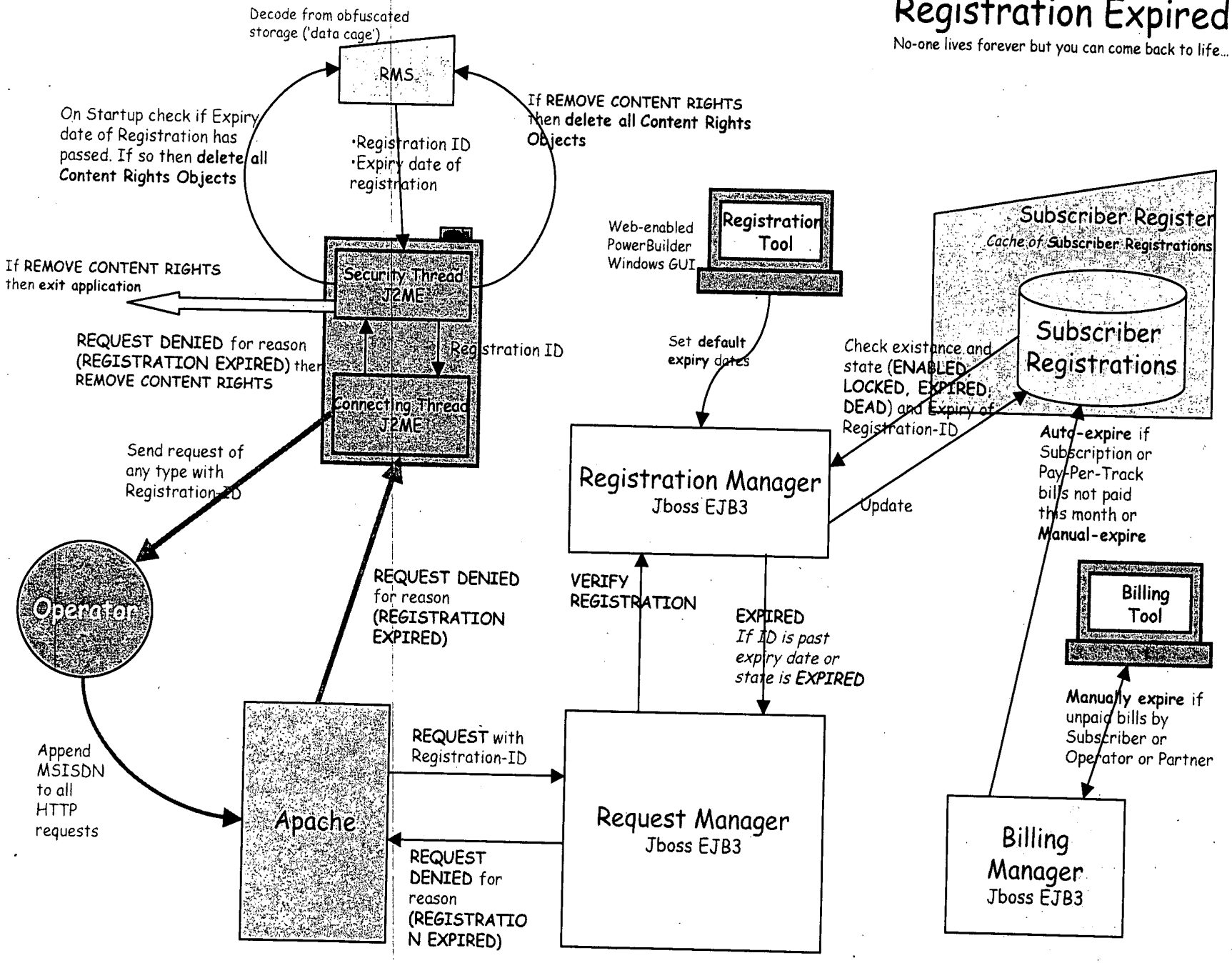
Registration

Activate your account if you have a valid license



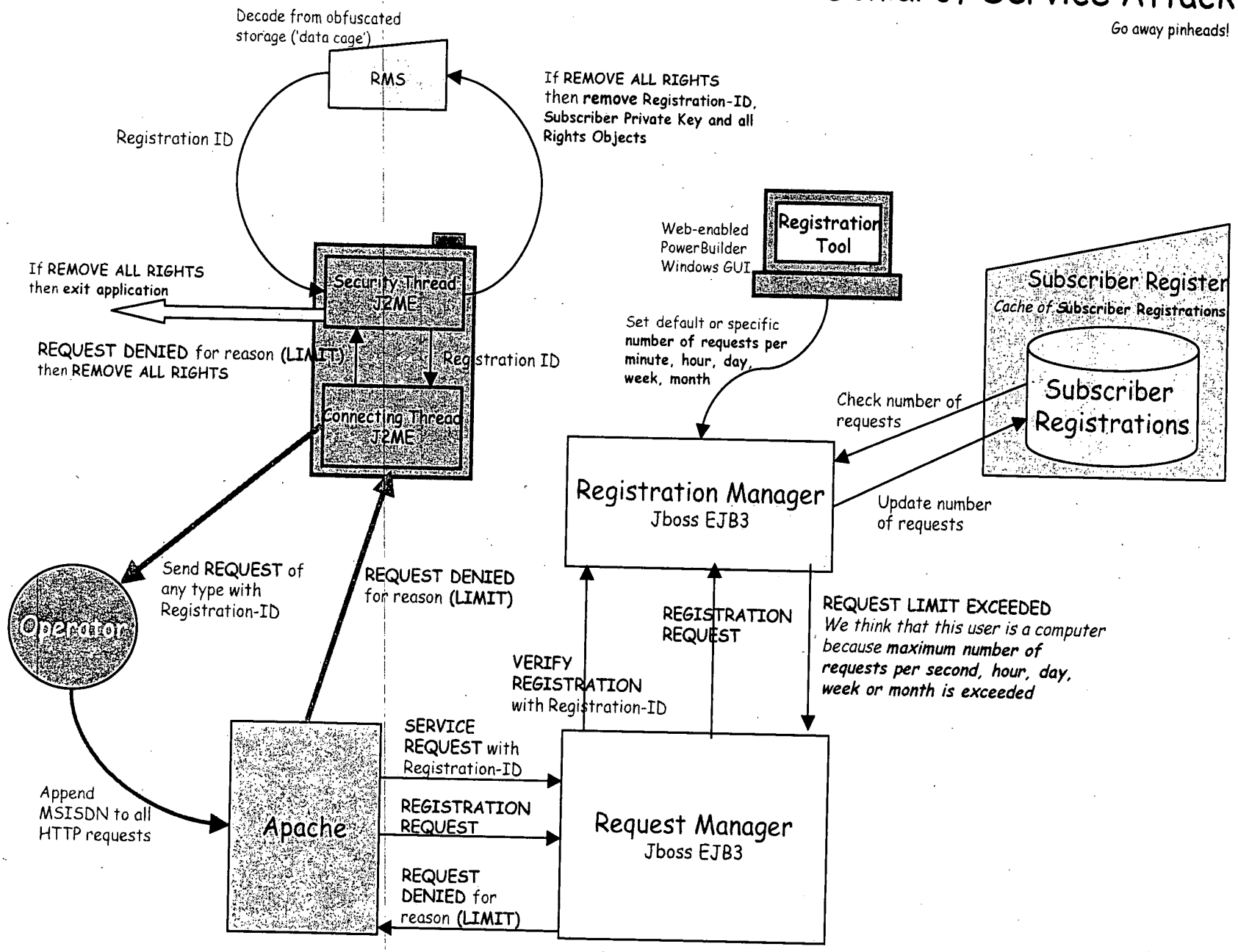
Registration Expired

No-one lives forever but you can come back to life...



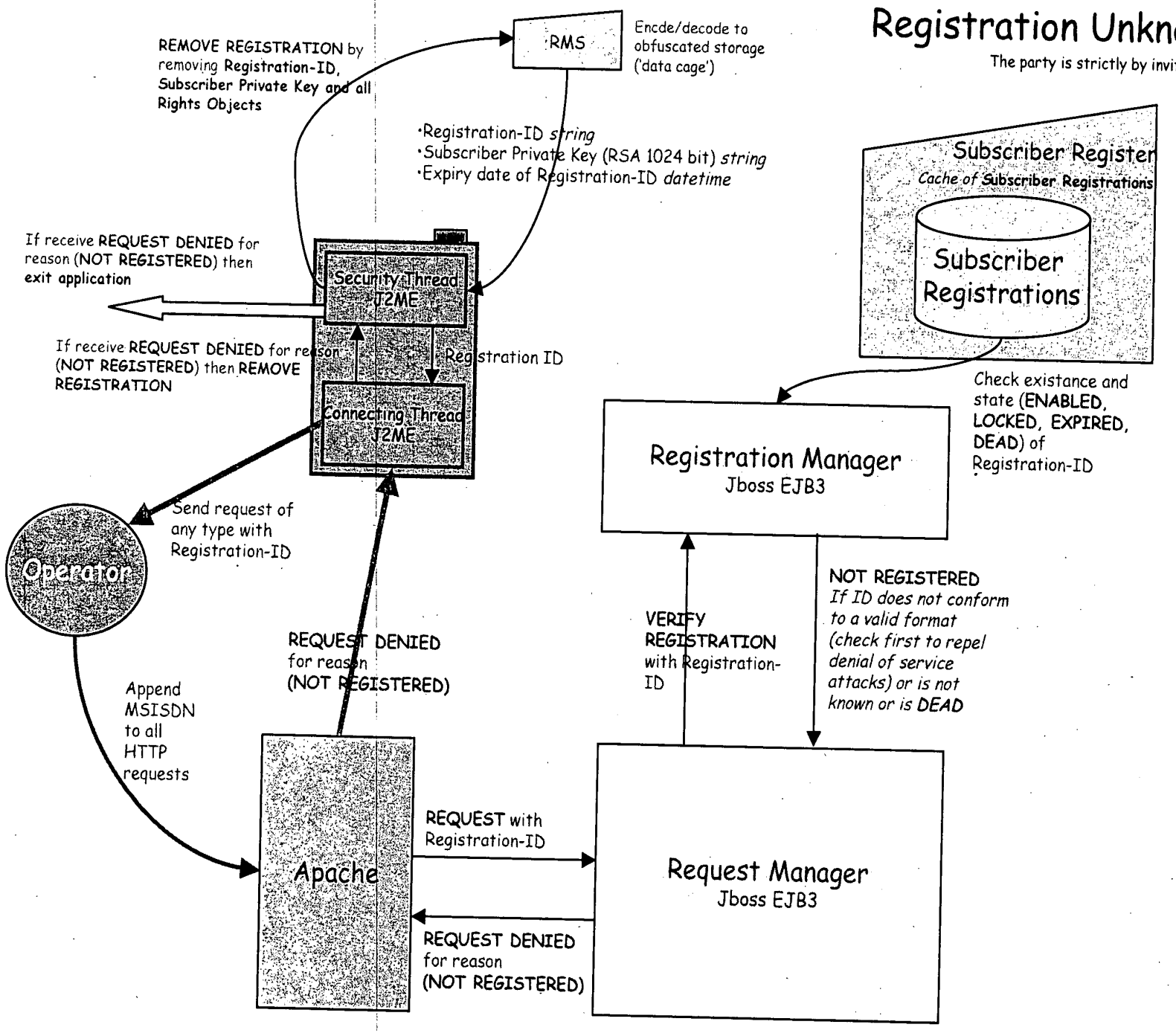
Denial of Service Attack

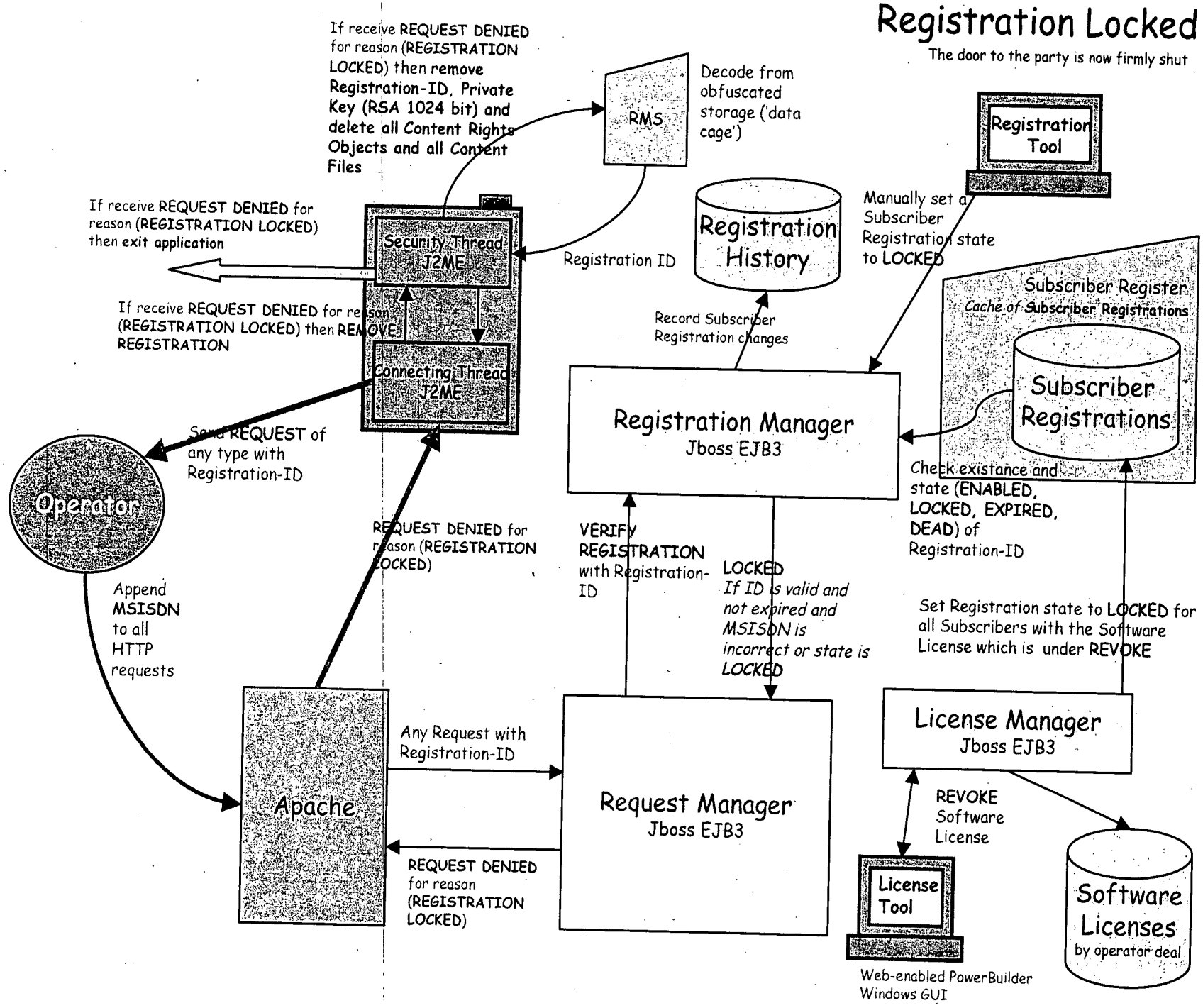
Go away pinheads!



Registration Unknown

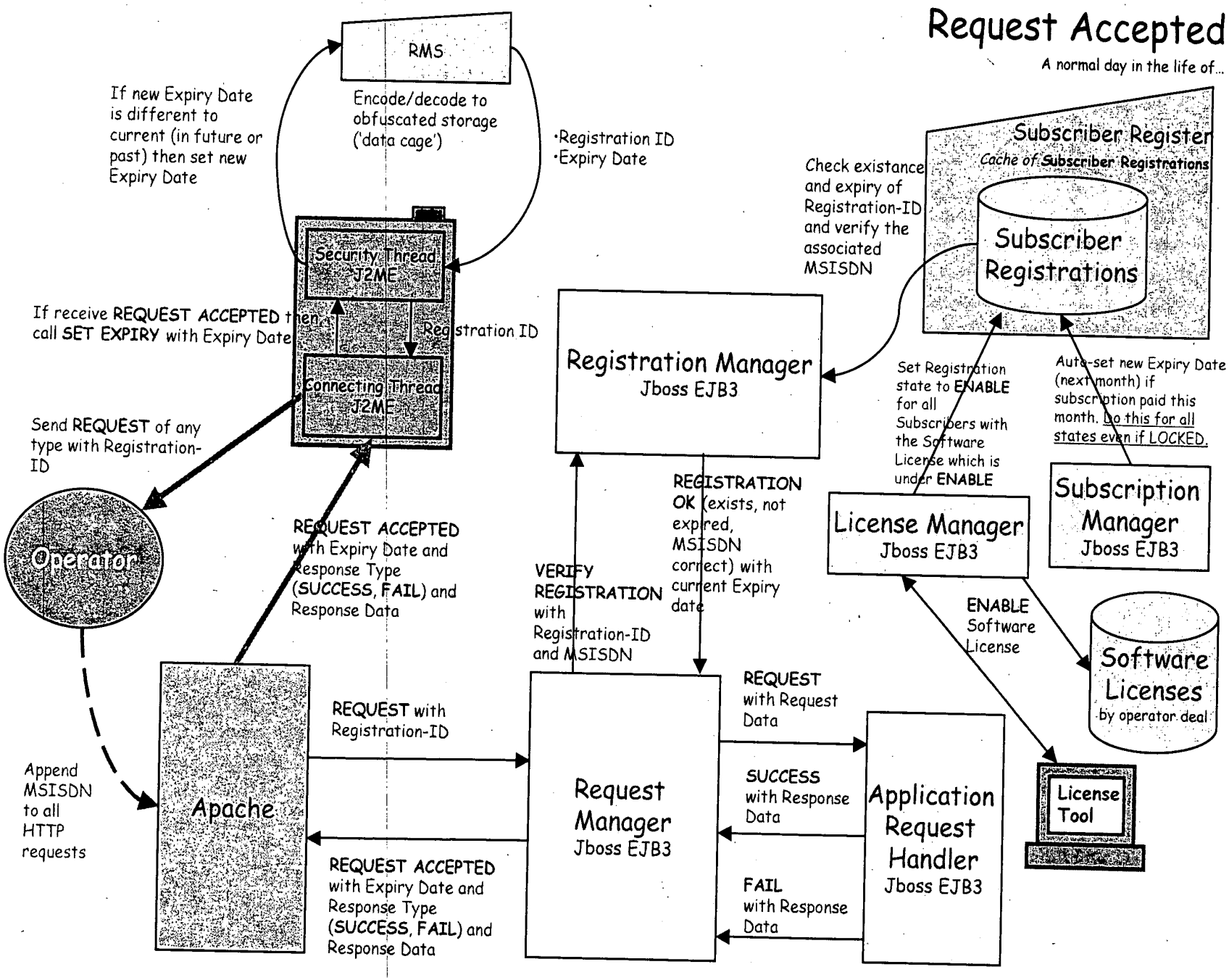
The party is strictly by invitation only





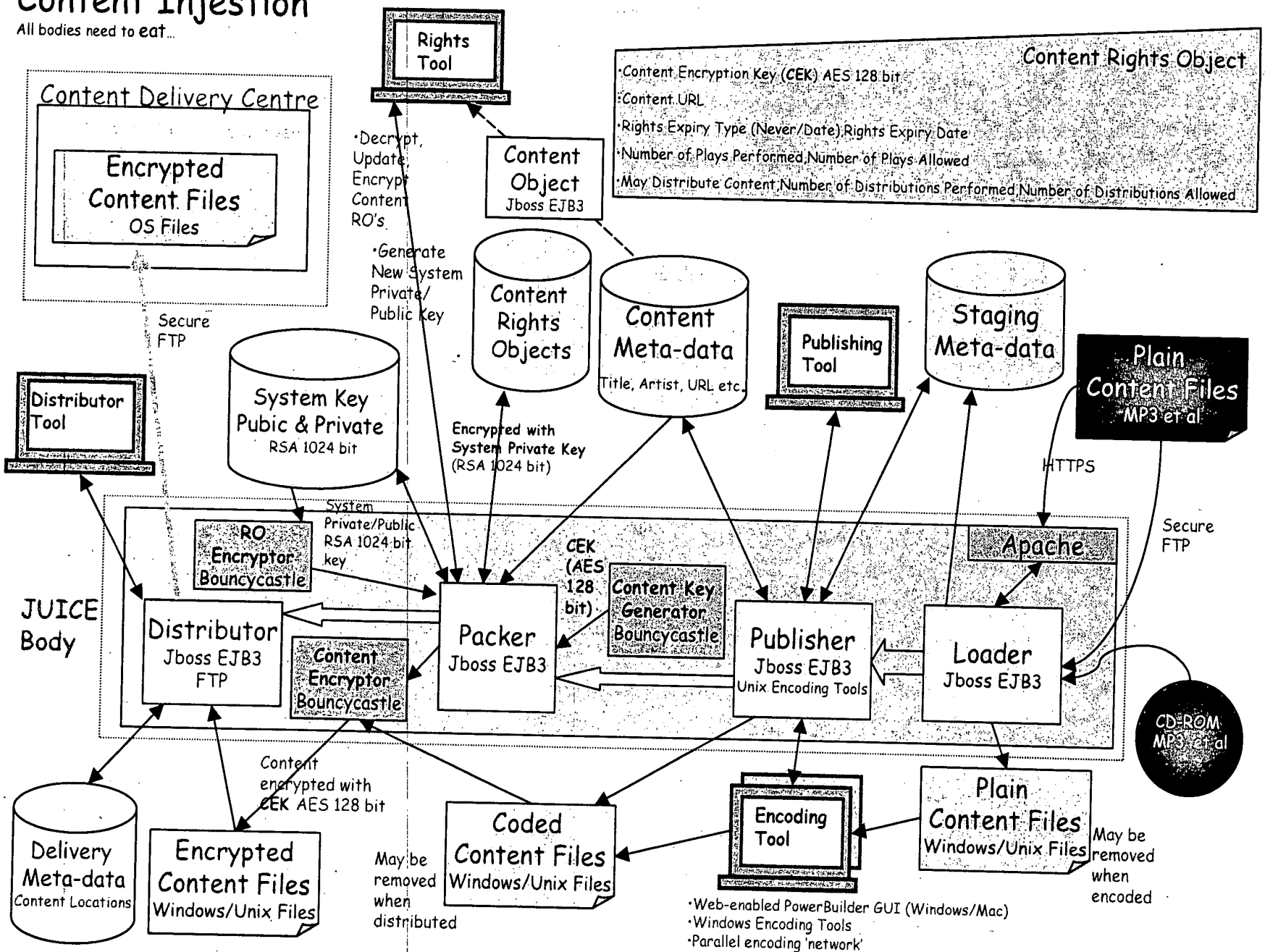
Request Accepted

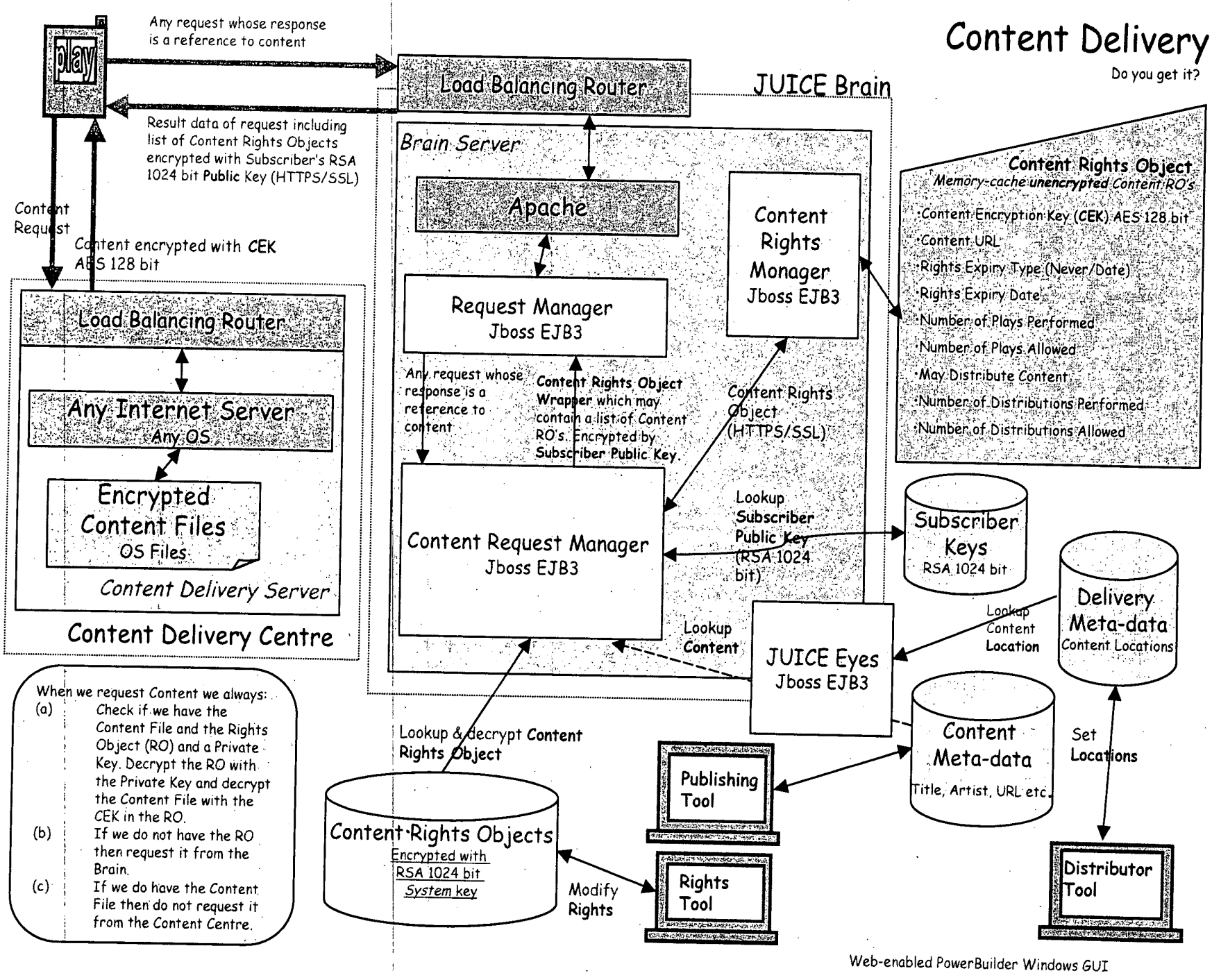
A normal day in the life of...



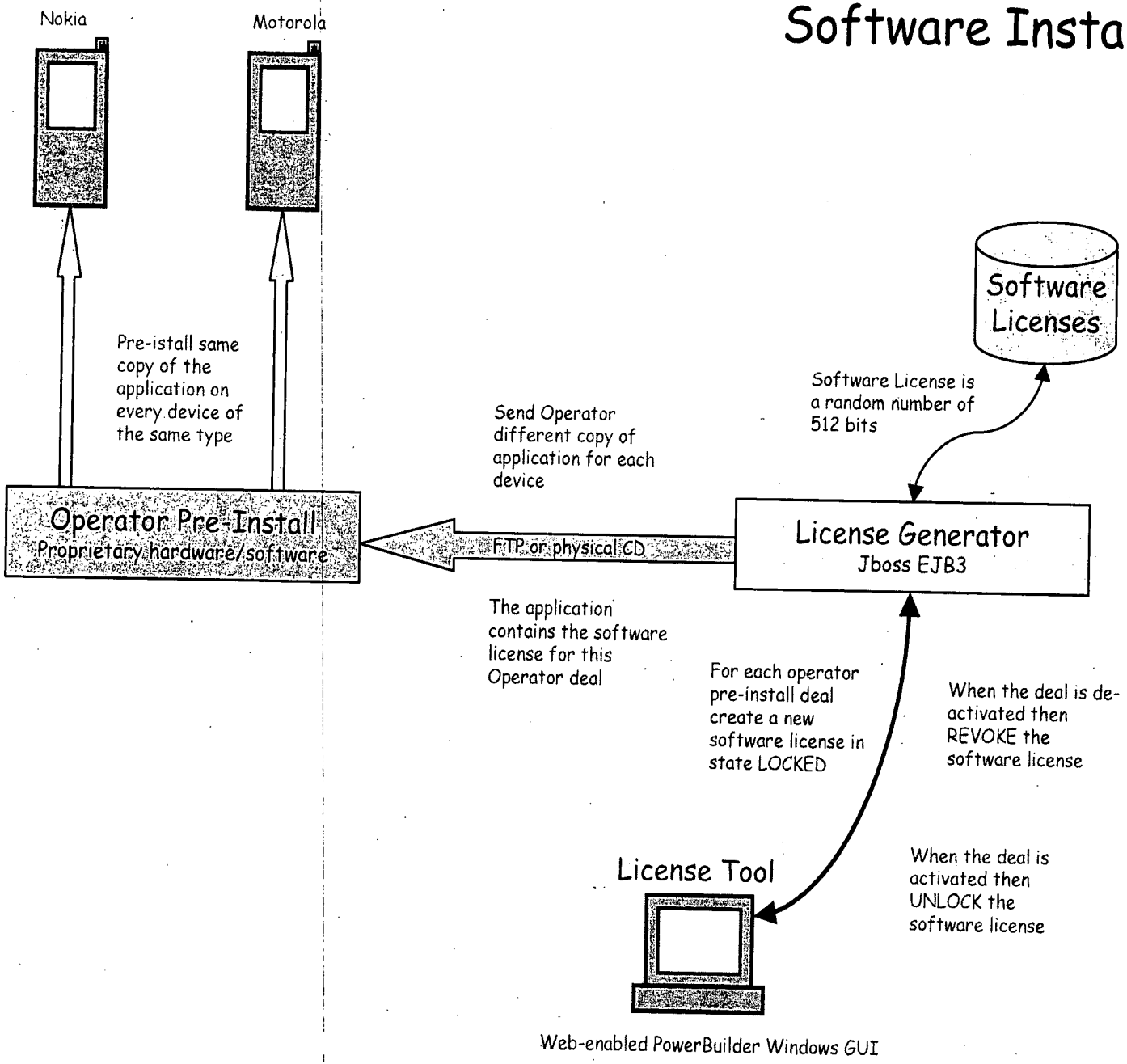
Content Injestion

All bodies need to eat...

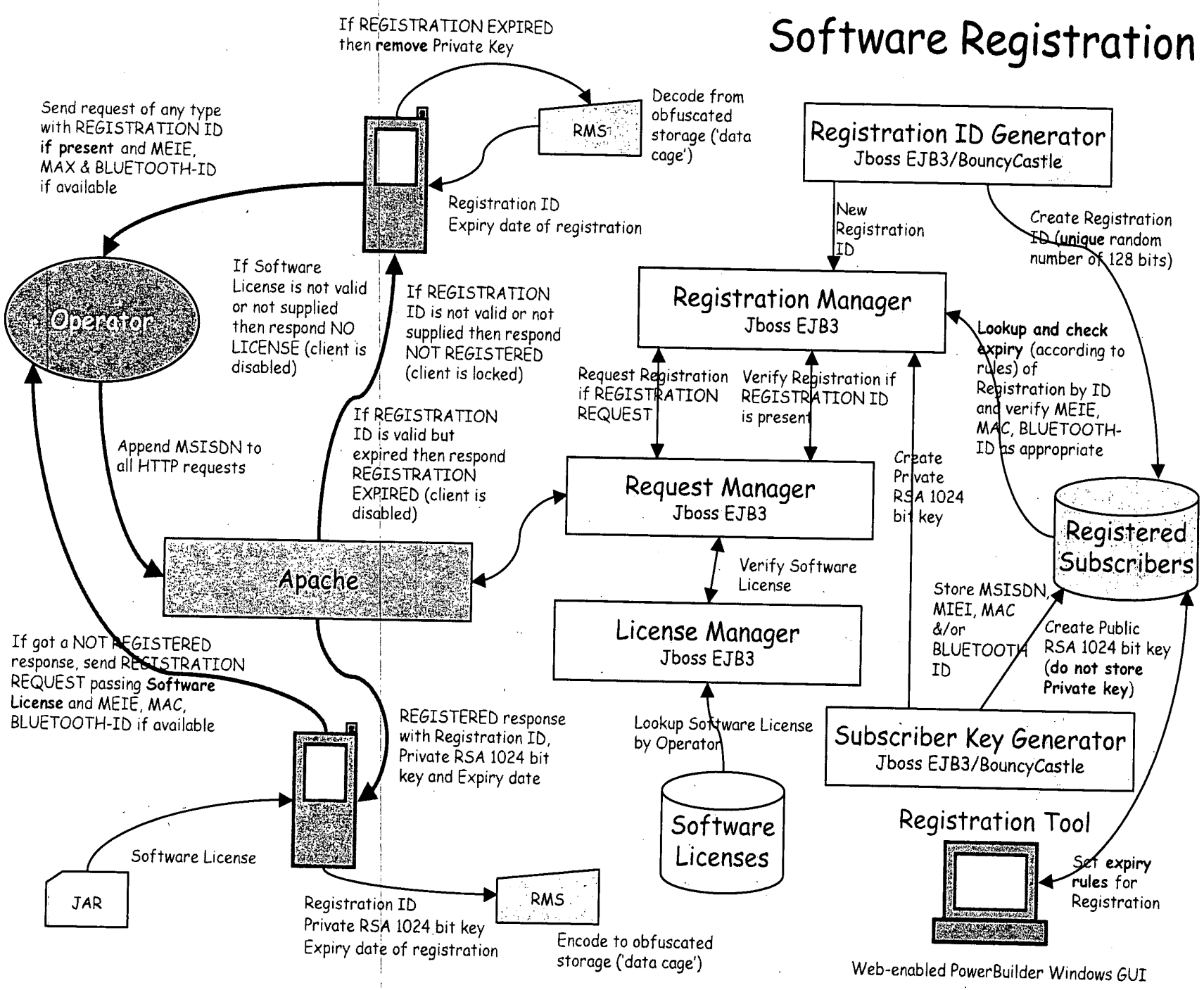


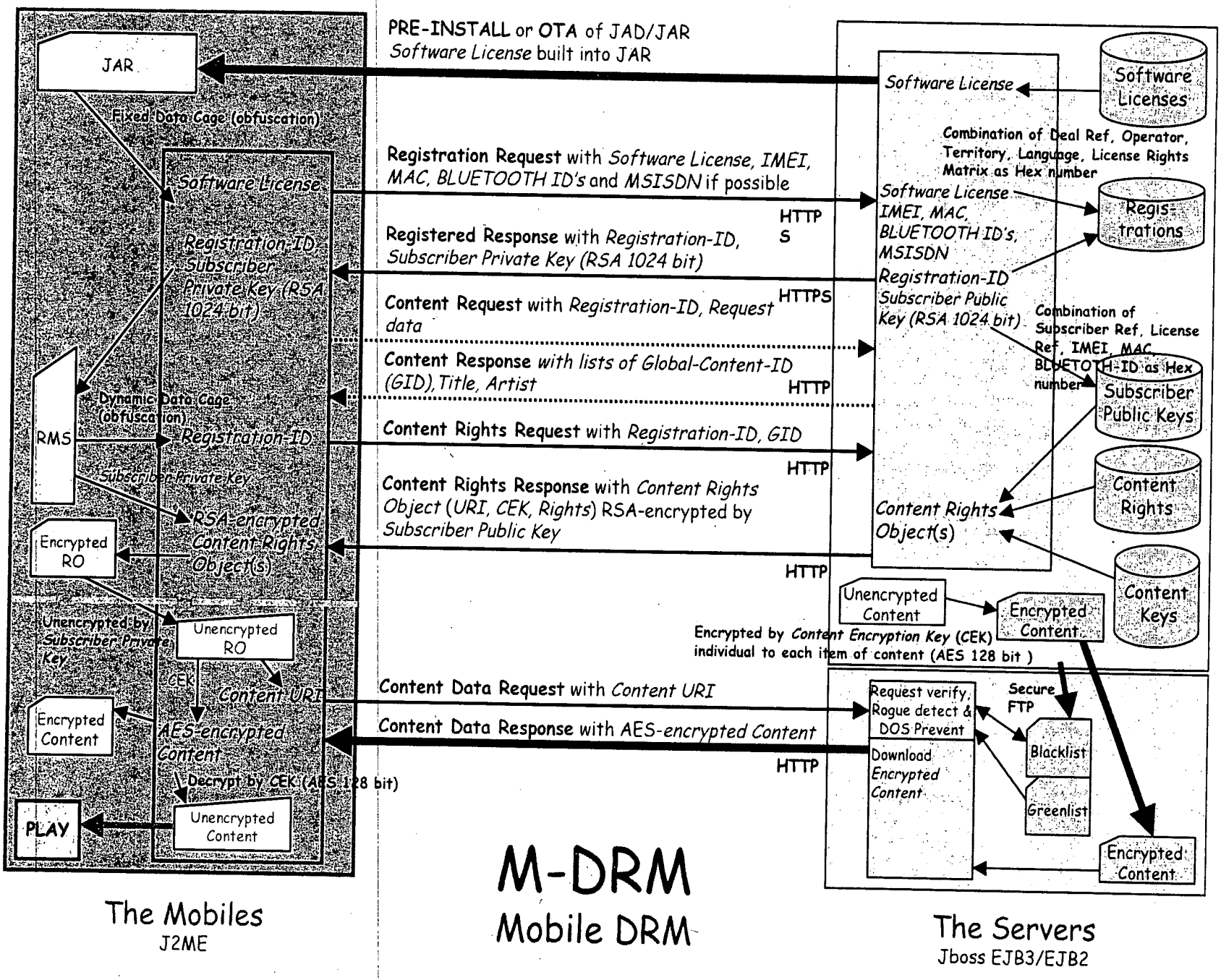


Software Installation

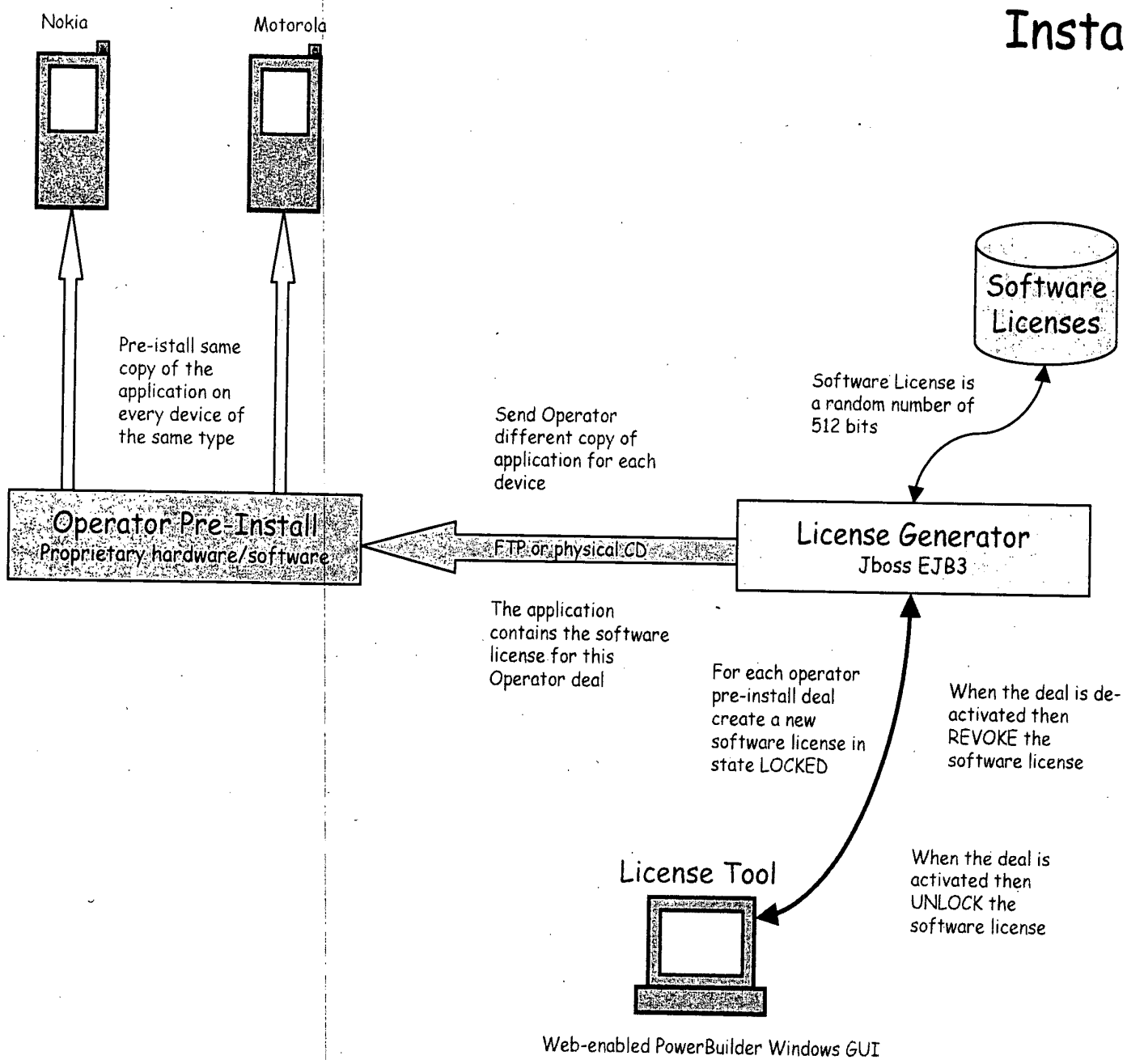


Software Registration

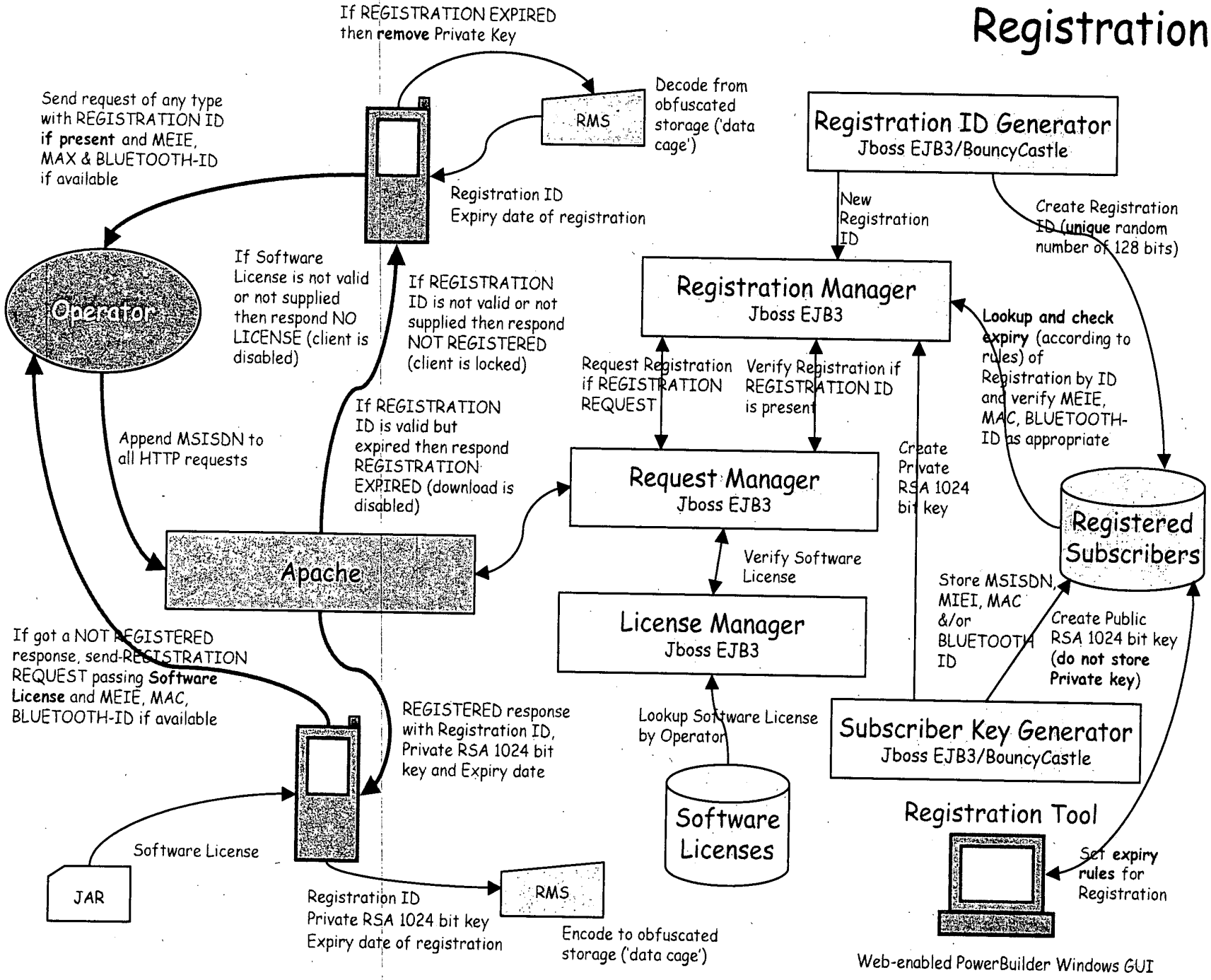




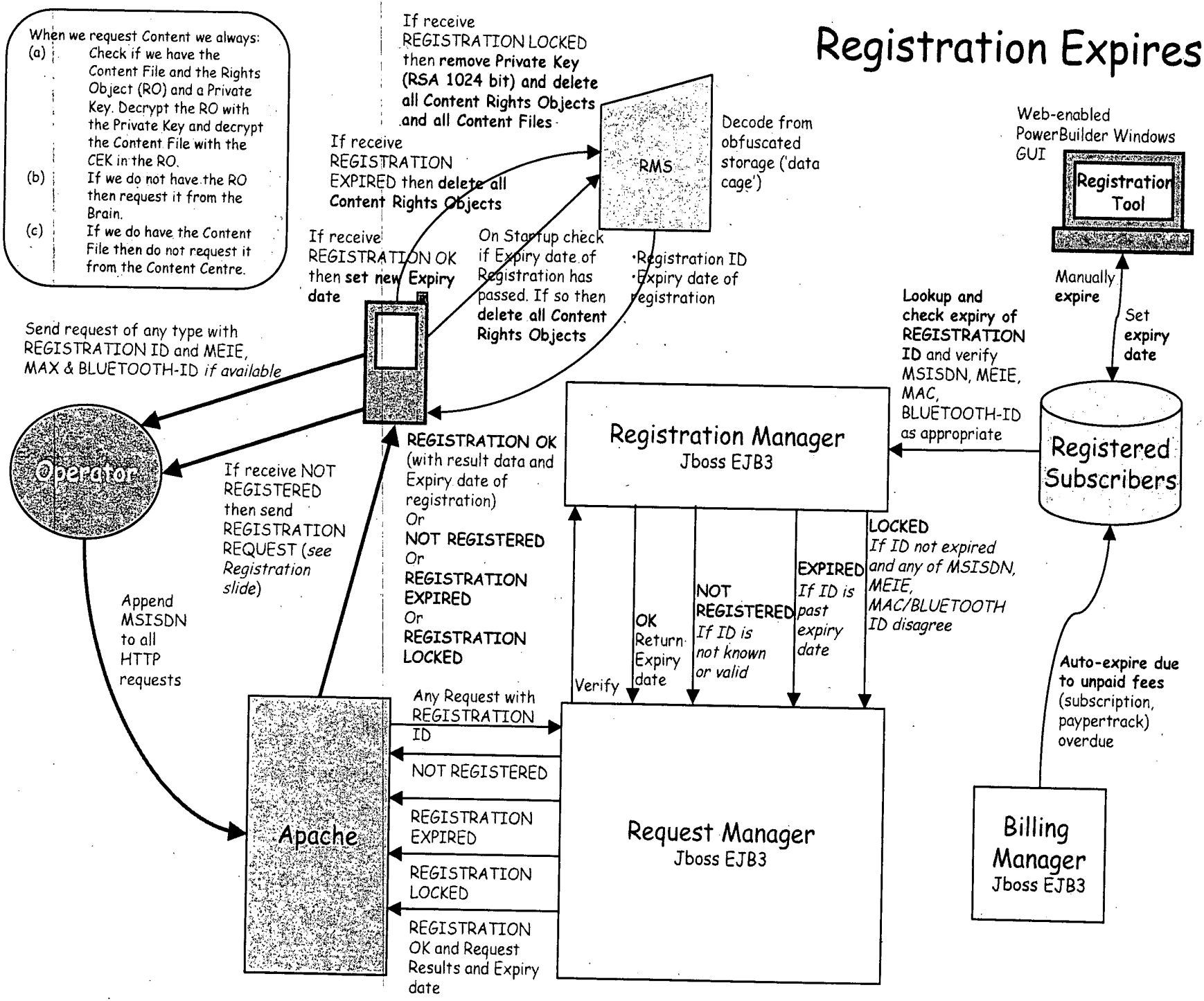
Installation

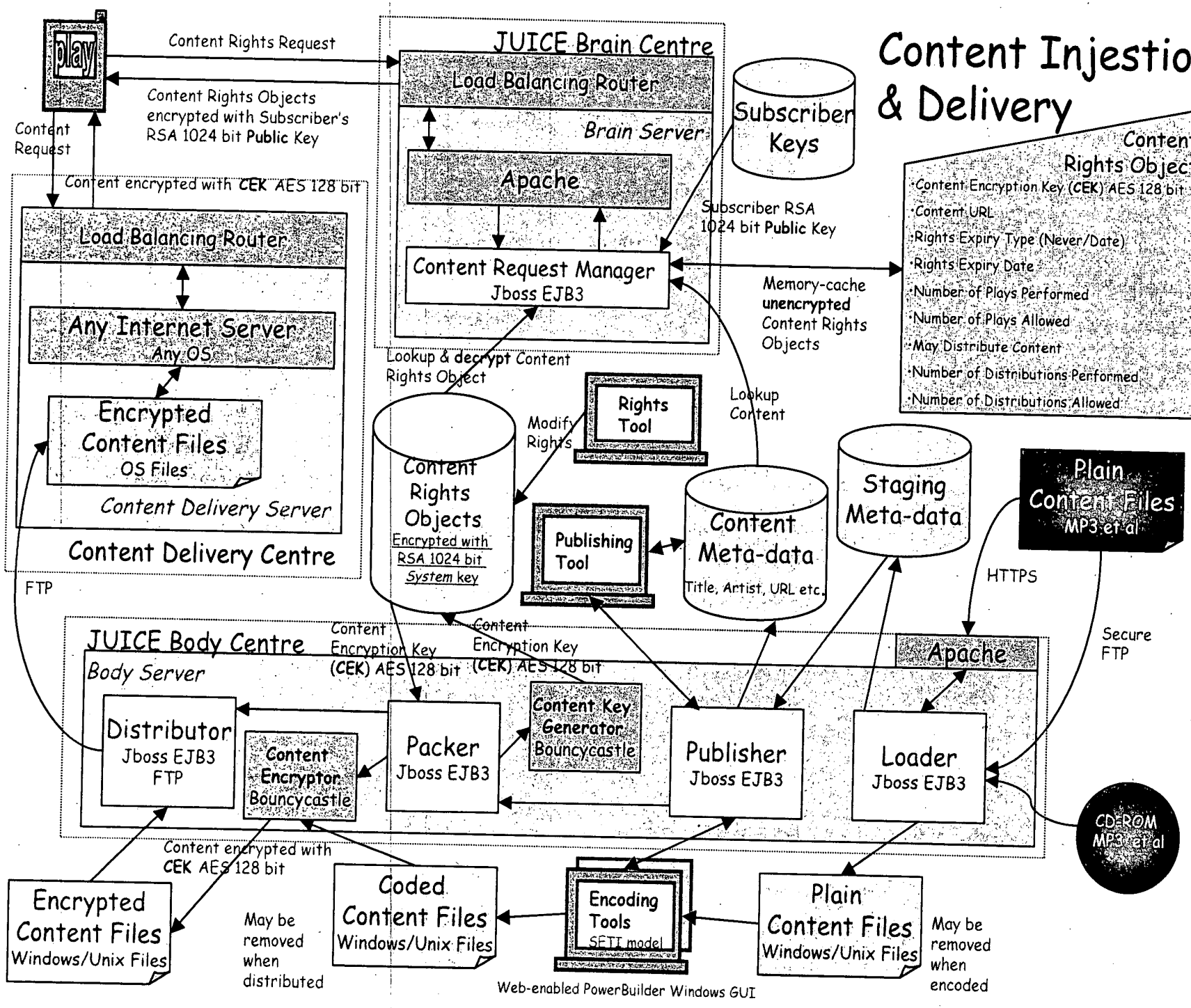


Registration



Registration Expires





MyFone Components & suitability for use in MusicStation

This table identifies the functional areas which have been identified in MyFone which are likely of use in MusicStation. The intention is to port these to Oracle within a EJB2 environment initially.

Area/Component	What it Does	Use in MusicStation	Beans	Depends on
Messages & Resources & I18N	The system for delivering text and images tailored to the properties of the user. This enabled white-labelling and internationalization (i18n)	Very useful to have. The user properties are most likely going to be different for MusicStation so will require some recoding. May be worth coming up with a slightly cleverer property matching system for this rather than placing all the fields in the DB (device_type_id, operator_id etc).	Message & Resource plus all the potential keys on them. Certainly need Country, Operator, Language, DeviceType	Cache Manager required for caching and searching for messages & resources at runtime.
Help page definition	An extension of the message mechanism for defining generic help messages in XML which can be rendered using a stylesheet suitable for the channel (WAP, web, etc). As messages, supports the substitution of values . Help messages can be included in other help messages to allow larger help messages to be constructed from smaller building blocks. Help messages can be organized into pages and sections.	We have considered implementing help for the MusicStation prototype using WAP pages and this functionality would be very useful for that. Could also be used to provide web-based help.	HelpPage, HelpSection, HelpSectionMessage, Message	Cache Manager & Messages & Resources & I18N
Devices & Device Build	Allows us to build jars for devices matching a set of device properties. Use custom built ant tasks which use SQL.	The MidletSuite bean provides this in a runtime system. Sits nicely independent of other system areas so can pull out as a component ok. Will be built from a combination of existing MyFone device type, select data and property representation taken from ChrisK's Detective work, and the new Netbeans based property mechanisms developed for the MusicStation demonstration.	DeviceType DeviceTypeProperty Property MidletSuite MidletSuiteData MidletSuiteDataProperty	Nothing
SMPP (premium and zero rated)	Allows us to send SMS and receive delivery reports.	Useful in case we ever plan to send or receive SMS within MusicStation. Would need integration with any new SMPP server that we use. Parts of the code rely on the use of MidletInstance and Service beans which we may wish to remove to use it in music station.	SMSMessage, PremiumSMSMessage SMSC ServiceSMSC SMSCTrigger, SMPPConnectionControllerMBean	(Currently) MidletInstance, Service
Subscriber/Operator	Bit loose this area. Manages 'users' as SIM cards with associated operators. Allows us to deliver specifics content and help to users on a given operator and deliver settings to them.	We are going to need to keep track of "users as SIMs" in MusicStation as in MyFone and our whole white-labelling is going to run off operator most likely.	Subscriber Operator Operator Address Operator Gateway OperatorPolicyList VirtualOperator	Not sure
Cache Manager	Generic caching built on top of Jboss TreeCache (a transactional, clustering cache)	We are going to need some caching ability. The messages and resources require this to be in place. We have some problems in this area, particularly with the time the cache takes to refresh (caused by the length of time it takes to recreate it before populating with data and then switching out the old	CacheManagerMBean	Needs TreeCacheAccess classes. No porting required for these either.

		cache and bringing in the new). In this case no porting is required. It does not use any specific entity beans so no EJB3 porting questions. Tom suggested looking at perhaps using the existing mechanism in conjunction with an alternative like JCS		
Product/Content model	Models products as a collection of different items of different content types.	Potentially relevant as the basis for representing music tracks in multiple file formats.	Product, Preview, Content, ContentType, PreviewType, ProductType	
Incident Monitor	General system for logging incidents based on log4j, and identifying situations which require alerts being raised, and raising those alerts. Supports assignment and closure of incidents, and converting abstract incidents into named incident types. Intended to replace the overwhelming number of warning/error log emails the dev team received with MyFone.	The IncidentMonitor is completely generic and is already plugged into some MusicStation components.		
Strategies & Splits	Allows users to be herded into groups and treated differently. Keeps record of how they were grouped for later analysis.	It does some user preference based work but this may fall under the capabilities of our suggestion engine. Was used in MyFone for controlling installation of the midlet, governing whether a user received the Java or WAP version.	Action Execution Strategy Split Segment MidletInstanceSplit	Relies slightly on MidletInstance but this can be removed easily I suspect
Scheduler	Allows us to run purpose built tasks and certain prescribed times on the system. Works very much like cron but can use the beans in the system. Has come in useful for adding functionality after releases.	Sits independent of the other areas of the system and can be easily extracted as a component. May be required to kick of self analysis tasks like calling the suggestion engine to change recommended content. Already ported as our initial test case of the porting process.	ServiceAction ScheduledAction	Nothing.
Tracking	Provide an API for asynchronously creating tracking records in a tracker table.	We shall certainly need tracking in MusicStation. Not sure if our currently solution is what we want entirely. Being only one bean we can quickly rewrite it in EJB3 with minimal work if we want. EJB3 has some better mapping capabilities which may give us a better tracking solution. Has some reliance on midlet instance and server etc structural we may wish to remove. Might be possible to use the IncidentMonitor as the basis for this, but Tom has concerns about performance which we should verify first. Tracker items need to be pre-defined so that they	Tracker TrackingManagerMBean	MidletInstance Server DeviceType

		are better structured than in MyFone.		
User Monitor	Suggests action to users who may be stuck during installation. Done by managing time delayed actions which pop-up later on and check if a use has progressed. If not it perform some action.	Would be required for OTA installations if we do that (e.g. following up with reminders for the user to make the manual changes to make their settings work), and possibly for followup to other user actions.	Midletinstance MidletInstallStatus MidletInstallHistory	MidletInstance Service DeviceType
Pricing Model	Associate items with price bands and services with price models. Price models use rules to calculate a particu	For MusicStation different attributes will control price, including territory, whether being purchased as album or single, purchase channel. Existing model could likely extend to cover this, though it does seem slightly too complex and hard to understand at the moment and could do with a cleanup.	PricingModel, PriceBand, PricingModelPrice	Language, Currency, ProductType
Application Group/Categories	Defines a menu set consisting of a hierarchy of categories. A category can contain further categories, or be associated with either a classification of products or an explicit list of products.	MusicStation menus are generally fairly static with tracks, albums, playlist etc lists appearing in the interface in set locations, but theoretically they could be modeled in data, with tabs mapping to the top level categories. In the browse screens (e.g. Browse Albums) there's more of a publishing approach where someone may need to manage and order the selection of items more dynamically.	ApplicationGroup, Category, ApplicationGroupCategory, CategoryChildCategory, CategoryProduct, CategoryArticle	
Network Settings	Sends network settings appropriate for the handset/network/contract type and for their intended use (whether midlet or browser)	Would be required for those operators who want to do OTA install of MusicStation. But we would hope that the operator would take some responsibility for getting the settings correct. This is not a perfect solution since it still sometimes requires the user to make some manual change or follow specific instructions.		
OTA Install / WAP Install / Signing	Delivers signed clients to handsets. Includes WAP/XHTML-based user flows to gather required information from the user and provide device/operator/contract type specific help.	Would be required for those operators who want to do OTA install of MusicStation.		

PTO/SB/06 (07-06)
Approved for use through 1/31/2007. OMB 0651-0032
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE

PATENT APPLICATION FEE DETERMINATION RECORD

Substitute for Form PTO-875

Application or Docket Number	Filing Date	☐ To be Mailed
12/299,505	05/20/2009	

APPLICATION AS FILED – PART I

	(Column 1)	(Column 2)	SMALL ENTITY ☐		OR	OTHER THAN SMALL ENTITY	
FOR	NUMBER FILED	NUMBER EXTRA	RATE ($)	FEE ($)		RATE ($)	FEE ($)
☐ BASIC FEE (37 CFR 1.16(a), (b), or (c))	N/A	N/A	N/A			N/A	
☐ SEARCH FEE (37 CFR 1.16(k), (i), or (m))	N/A	N/A	N/A			N/A	
☐ EXAMINATION FEE (37 CFR 1.16(o), (p), or (q))	N/A	N/A	N/A			N/A	
TOTAL CLAIMS (37 CFR 1.16(i))	minus 20 =	*	X $ =		OR	X $ =	
INDEPENDENT CLAIMS (37 CFR 1.16(h))	minus 3 =	*	X $ =			X $ =	
☐ APPLICATION SIZE FEE (37 CFR 1.16(s))	If the specification and drawings exceed 100 sheets of paper, the application size fee due is $250 ($125 for small entity) for each additional 50 sheets or fraction thereof. See 35 U.S.C. 41(a)(1)(G) and 37 CFR 1.16(s).						
☐ MULTIPLE DEPENDENT CLAIM PRESENT (37 CFR 1.16(j))							
* If the difference in column 1 is less than zero, enter "0" in column 2.			TOTAL			TOTAL	

APPLICATION AS AMENDED – PART II

AMENDMENT 11/04/2008	(Column 1) CLAIMS REMAINING AFTER AMENDMENT		(Column 2) HIGHEST NUMBER PREVIOUSLY PAID FOR	(Column 3) PRESENT EXTRA	SMALL ENTITY RATE ($)	SMALL ENTITY ADDITIONAL FEE ($)	OR	OTHER THAN SMALL ENTITY RATE ($)	OTHER THAN SMALL ENTITY ADDITIONAL FEE ($)
Total (37 CFR 1.16(i))	* 24	Minus	** 24	= 0	X $ =		OR	X $52=	0
Independent (37 CFR 1.16(h))	* 2	Minus	***3	= 0	X $ =		OR	X $220=	0
☐ Application Size Fee (37 CFR 1.16(s))									
☐ FIRST PRESENTATION OF MULTIPLE DEPENDENT CLAIM (37 CFR 1.16(j))							OR		
					TOTAL ADD'L FEE		OR	TOTAL ADD'L FEE	0

AMENDMENT	(Column 1) CLAIMS REMAINING AFTER AMENDMENT		(Column 2) HIGHEST NUMBER PREVIOUSLY PAID FOR	(Column 3) PRESENT EXTRA	RATE ($)	ADDITIONAL FEE ($)	OR	RATE ($)	ADDITIONAL FEE ($)
Total (37 CFR 1.16(i))	*	Minus	**	=	X $ =		OR	X $ =	
Independent (37 CFR 1.16(h))	*	Minus	***	=	X $ =		OR	X $ =	
☐ Application Size Fee (37 CFR 1.16(s))									
☐ FIRST PRESENTATION OF MULTIPLE DEPENDENT CLAIM (37 CFR 1.16(j))							OR		
					TOTAL ADD'L FEE		OR	TOTAL ADD'L FEE	

* If the entry in column 1 is less than the entry in column 2, write "0" in column 3.

** If the "Highest Number Previously Paid For" IN THIS SPACE is less than 20, enter "20".

*** If the "Highest Number Previously Paid For" IN THIS SPACE is less than 3, enter "3".

The "Highest Number Previously Paid For" (Total or Independent) is the highest number found in the appropriate box in column 1.

Legal Instrument Examiner:
/BRENDA L. TURNER/