



Media Center Edition Control Server

MCECS Service Installation Instructions
Control Protocol Functional Specification

Table of Contents

Overview.....	4
Getting Ready	5
Installation.....	5
Autonomic Home MCE Control Server Setup Wizard.....	5
Configuration Utility	6
Control Server - Port.....	7
Media Center Instances.....	7
Web Server Port.....	7
Web Server Missing Album Art.....	8
Server Status	8
License Status	8
Opening Your First Client Session	9
Local Connection	9
LAN Connection.....	11
Internet Connection.....	11
MCE Control Specification	12
Protocol Conventions.....	12
Initiating control sessions.....	12
Starting the MCE Session	12
Selecting an MCE Instance	13
List Processing.....	14
Asynchronous processing	15
Displaying Album Art	16
Command Reference	17
General Commands.....	17
Banner	17
CLS	17
Exit.....	17
Help.....	17
GetVersions.....	18
GetLicenseMessage.....	19
Time	20
BrowseEncodings.....	21
SetEncoding	22
MCE Instance Commands	23
BrowseInstances.....	23
SetInstance	24
Interfacing with the Media Center Shell	25
MsgBox.....	25
Media Center Feedback	26
GetMCEStatus.....	26
SubscribeEvents	27
StateChanged Message.....	28
ReportState Message.....	30
Media Center Interface Navigation.....	31
Navigate	31

Transport Control	32
Transport Commands	32
Browse Media Commands	33
BrowseAlbums	33
BrowseArtists	34
BrowseGenres	35
BrowseNowPlaying.....	36
BrowsePlaylists	37
BrowseTitles	38
Play Media Commands	39
PlayAlbum	39
PlayArtist	40
PlayGenre.....	41
PlayPlaylist.....	42
PlayTitle	43
JumpToNowPlayingItem	44
RemoveNowPlayingItem	45
Filter Media Library Commands.....	46
SetMediaFilter	46
PushMediaFilter	47
PopMediaFilter.....	48
GetMediaFilter	49
IR Commands	50
SendKeys	50

Overview

This document describes the Media Center Edition Control Server, and its control protocol (MCCP). MCCP is implemented via a Microsoft Windows Service application that resides on a host Media Center Edition server.

This protocol provides two way communications and control of the Media Center Edition Interface (ehshell.exe) and associated media libraries over a TCP-IP session on a port specified by the configuration utility.

Typical applications include proprietary control systems, remote controls, web-based and rich client applications, and touch-panel interfaces.

Commands are included for media transport control, media library browsing, MCE shell navigation, and posting dialog boxes within the MCE Interface. Feedback is provided for browsing, currently playing media, and navigation.

MCCP offers a ready-to-use solution for supporting network media playback in your custom consumer electronics application. The control protocol allows a client device to interactively access any of Media Center's functionality and to retrieve feedback for those actions in an asynchronous manner. The client device or application can use the content enumeration and selection primitives in the MCCP control protocol to create their own custom UI.

Getting Ready

MCE Control Server requires Windows XP Media Center Edition, rollup 2, or Windows Vista in order to run.

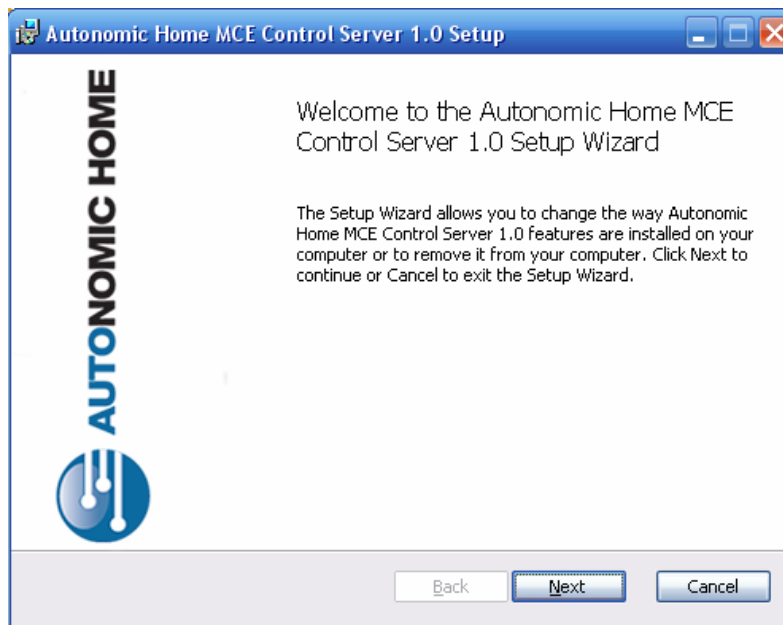
Since the host Media Center Edition computer will be accessed by external devices (your client), we strongly recommend that the host computer use a static IP address. If your computer is using DHCP to obtain an IP address, or if you're not sure, consult your Microsoft Windows documentation for instructions on how to configure your PC to use a static address. You should do this before running the configuration utility.

If the Media Center Interface is running, shut it down before beginning the installation process.

Installation

Autonomic Home MCE Control Server Setup Wizard

Download the MCE Control Server from the [Autonomic Home Website](#). The files in the .zip file should be extracted to a folder on your hard-drive. Run the installation .msi file and the installer will take you through the initial installation process.



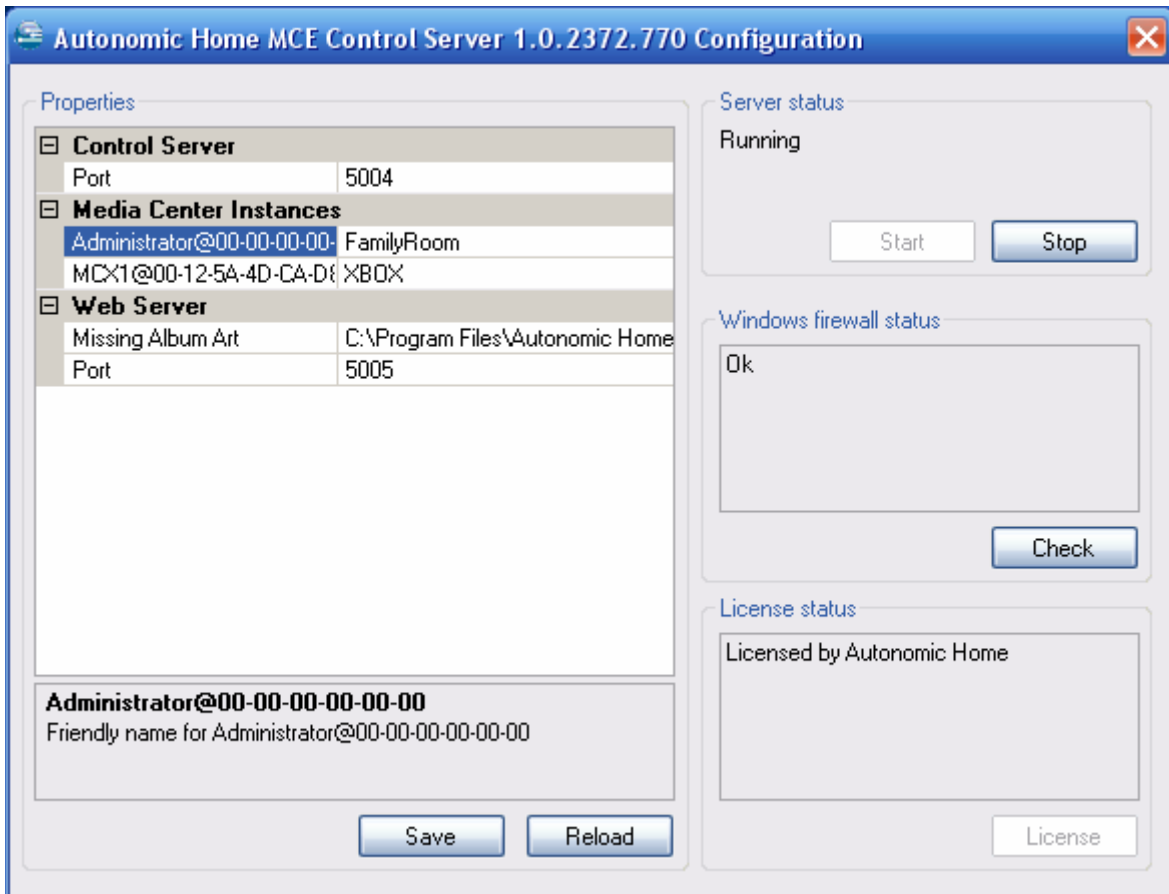
When the installation has completed, the installation wizard will ask you to reboot your machine. This is an important step to insure the protocol service is able to start properly.

Configuration Utility

After rebooting, the next step is to configure the server's ports and register your license with Autonomic, Inc.

First, make sure that the Microsoft Media Center Edition shell interface is running. You'll find it most convenient later if you resize the MCE interface to about half your screen height and width for testing.

From the Start Menu, select the Autonomic Home folder, and run the Autonomic Home MCE Control Server Configuration utility.



There are several settings you should observe on this screen.

Properties

Control Server - Port. This setting establishes the port number on which your client application will communicate with the server. This should only be changed if you have a known conflict, or you are directed by Autonomic Home technical support to do so. If your client will be accessing the MCE Control Server through a hardware firewall (through the internet), then you must configure the firewall to forward requests on this port to the IP address of the host PC. The installation utility will open the default port (5004) on the Windows Firewall during installation. If you change this setting, you must manually re-configure the Windows Firewall to allow communication on this port. Remember, the Windows Firewall is effective even when communicating within your LAN.

Media Center Instances – MCE Control Server allows your client application to control multiple instances on the host MCE computer, as well as those belonging to Media Center Extender devices. By default, these sessions are identified by their Network Identification Code (NIC). This section allows you to establish more friendly names that your client can refer to in the control protocol. (ie. “Mikes Profile”, “Family Room”, or “XBOX”)

In this example, two instances are running. The first one belongs to the Administrator account on the MCE computer, and the other belongs to an XBOX extender session. You can change the value after the “=” to a name that’s easier to remember and use. The MCE client can then refer to this specific session with that name.

Example: Administrator@00-00-00-00-00-00=Family Room

The easiest way to identify the session IDs is to start with one known session and give it a friendly name. Then you can start another session by logging into another user account on the computer (don’t log off of this session, however), or by starting up a Media Center Extender device. If using an XBOX 360 extender, the Media Center interface on the XBOX must be started. INTEGRATION TIP: Pressing the green start button on the XBOX 360 remote control instead of the power button will start the XBOX in Media Center Mode.

When the second device or session is started, press the “Reload” button on the configuration utilities main screen. The session for the newly activated device or login should appear. Now you can give it a friendly name. If you have more extenders or sessions, continue this process until you have given them all friendly names.

Web Server Port. The MCE Control Server provides URLs to .jpg image files so that your client application can display album art. (See **Displaying Album Art** section). The URL will point to a HTTP server application (installed with MCE Control Server) on the Media Center computer. This setting establishes the port number on which your client can access the images. Once again, this setting should only be changed if absolutely necessary.

The installation utility will open the default port (5005) on the Windows Firewall during installation. If you change this setting, you must manually re-configure the Windows Firewall to allow communication on this port. Remember, the Windows Firewall is effective even when communicating within your LAN.

Web Server Missing Album Art.

Use this setting to point to a .jpg file to be displayed when album art cannot be located for a particular album. Picture files should be sized at 200x200 pixels. A default file is provided.

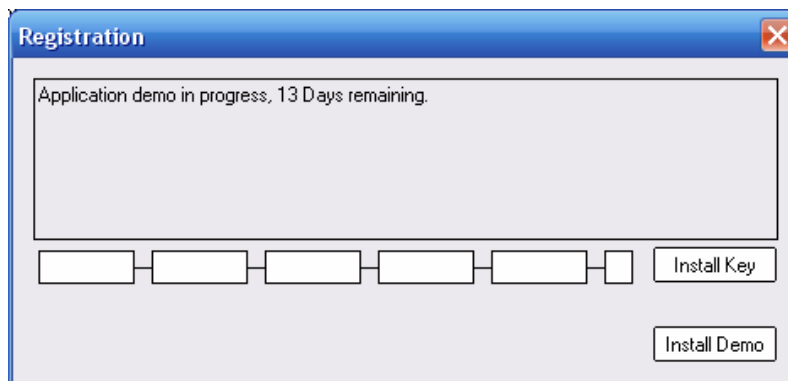
Server Status

The Server Status area of the utility tells you if the MCE Control Service is running, and allows you to start or stop the service. If it is not running, you should click the Start button now.

License Status

MCE Control Server is licensed per host computer. This area tells you if you are running a trial license, permanent license, or if you are unlicensed.

In order to use the MCE Control Server, you must register a valid license. Clicking on the “License” button will open the following dialog box:



If you have a valid license key, enter it in the boxes provided. If you have been provided a license file by Autonomic Controls, you will have the option of browsing for the license file. Otherwise, click the Install Demo button to register a 14 day demo license. In order to register a license, your computer must have an active connection to the internet. **Please note that while in demo mode, MCECS requires a full time connection to the internet in order to operate. This restriction is eliminated once you have entered a valid license key.**

Opening Your First Client Session

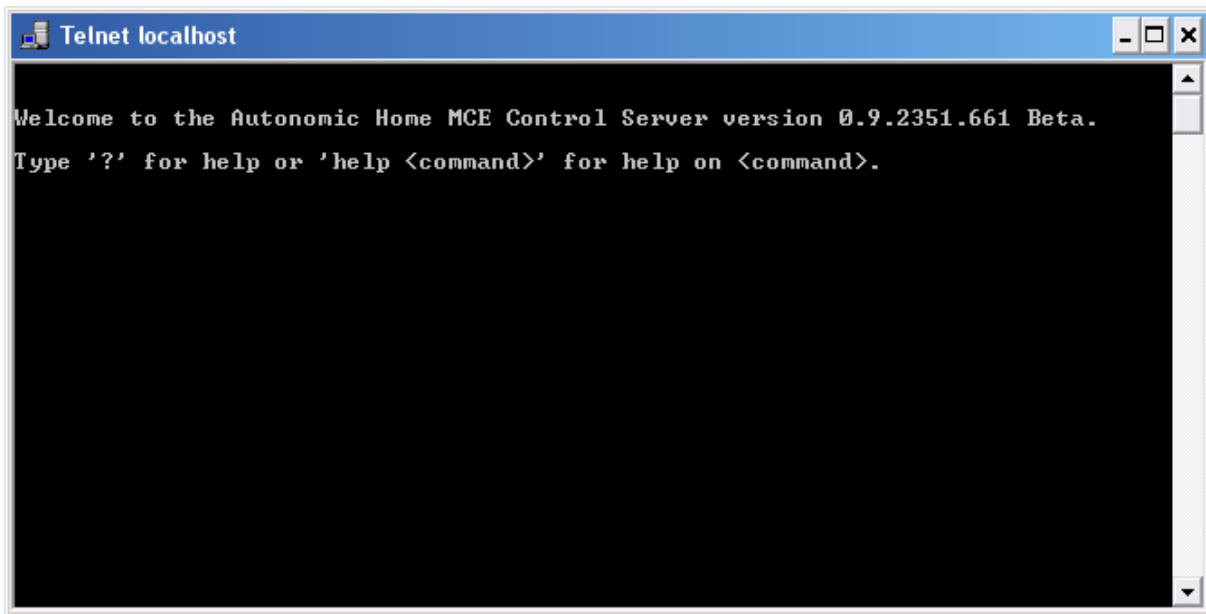
Once the MCE Control Server is installed and configured, it's time to open a test client session to and test it out. We'll do this using a telnet session.

It's best to first test using a telnet session on the MCE computer, then using another computer on the LAN, and finally over the internet. Testing in this order will help you localize any connection issues you may encounter due to firewall limitations.

Local Connection

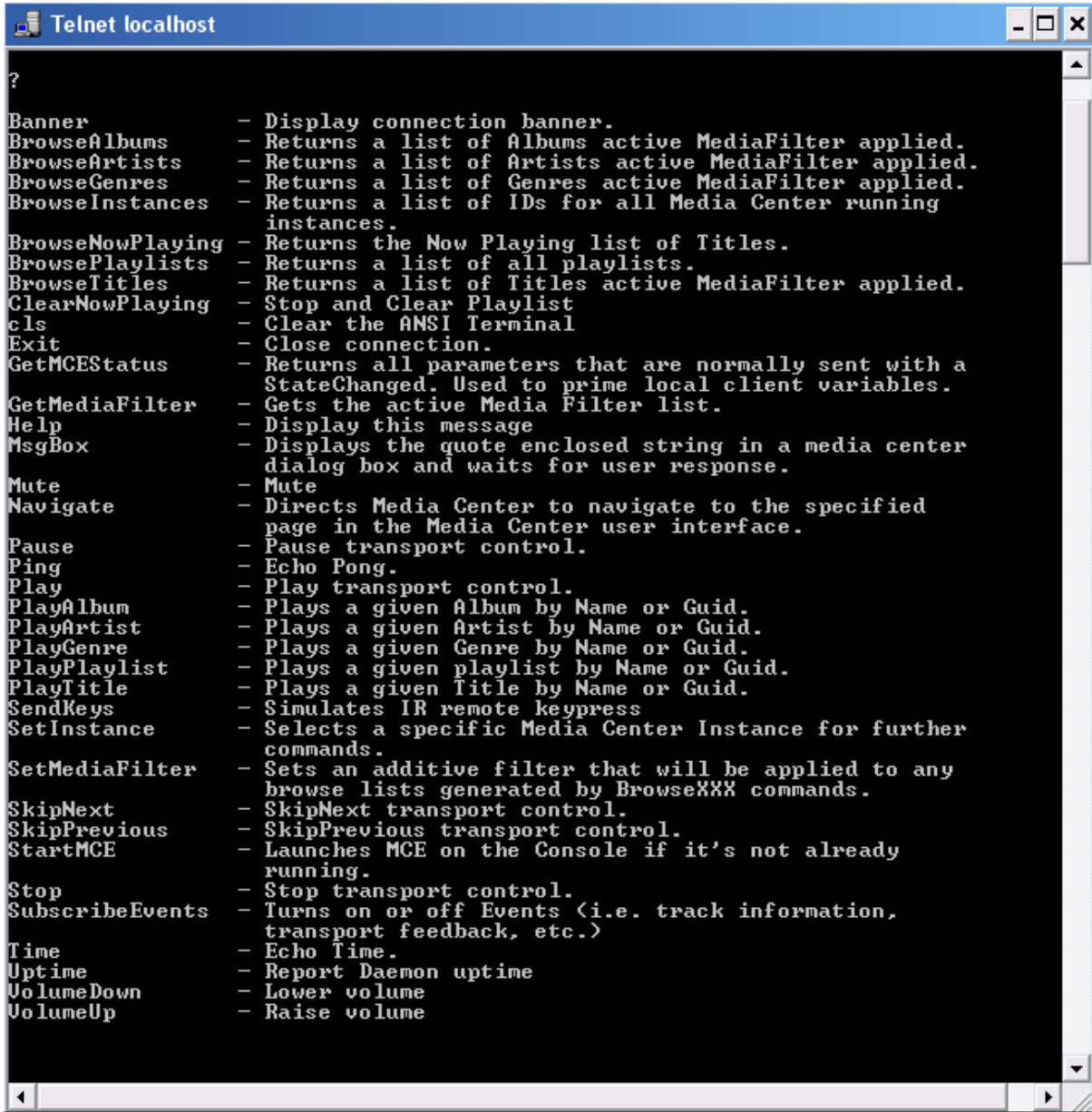
From the Start Menu, click Run, and type "telnet localhost 5004". If you have changed the server port number in the configuration utility, remember to substitute the number 5004 with the port you have selected.

If everything is working ok, you should see a window that looks something like this:



If you don't get a connection to the service, double check the Server Port setting in the configuration utility.

Once you have a connection to the server, type "?" and hit enter. The MCE Control Server will send a list of valid commands.



```
?
Banner          - Display connection banner.
BrowseAlbums    - Returns a list of Albums active MediaFilter applied.
BrowseArtists   - Returns a list of Artists active MediaFilter applied.
BrowseGenres    - Returns a list of Genres active MediaFilter applied.
BrowseInstances - Returns a list of IDs for all Media Center running
                 instances.
BrowseNowPlaying - Returns the Now Playing list of Titles.
BrowsePlaylists - Returns a list of all playlists.
BrowseTitles   - Returns a list of Titles active MediaFilter applied.
ClearNowPlaying - Stop and Clear Playlist
cls             - Clear the ANSI Terminal
Exit           - Close connection.
GetMCEStatus    - Returns all parameters that are normally sent with a
                 StateChanged. Used to prime local client variables.
GetMediaFilter  - Gets the active Media Filter list.
Help           - Display this message
MsgBox         - Displays the quote enclosed string in a media center
                 dialog box and waits for user response.
Mute           - Mute
Navigate        - Directs Media Center to navigate to the specified
                 page in the Media Center user interface.
Pause          - Pause transport control.
Ping           - Echo Pong.
Play           - Play transport control.
PlayAlbum      - Plays a given Album by Name or Guid.
PlayArtist     - Plays a given Artist by Name or Guid.
PlayGenre      - Plays a given Genre by Name or Guid.
PlayPlaylist   - Plays a given playlist by Name or Guid.
PlayTitle      - Plays a given Title by Name or Guid.
SendKeys       - Simulates IR remote keypress
SetInstance    - Selects a specific Media Center Instance for further
                 commands.
SetMediaFilter  - Sets an additive filter that will be applied to any
                 browse lists generated by BrowseXXX commands.
SkipNext       - SkipNext transport control.
SkipPrevious   - SkipPrevious transport control.
StartMCE       - Launches MCE on the Console if it's not already
                 running.
Stop           - Stop transport control.
SubscribeEvents - Turns on or off Events (i.e. track information,
                 transport feedback, etc.)
Time           - Echo Time.
Uptime         - Report Daemon uptime
VolumeDown     - Lower volume
VolumeUp       - Raise volume
```

Try out a few commands, such as “BrowseAlbums”. You should get a listing of Albums from your media library.

If you get an error message stating that the Media Center Server is not licensed, use the configuration utility to register a valid license key, or obtain a trial license.

If you get an error message stating that no instance is started, you can issue the command “startMCE” to start the Media Center Interface. (You’ll have to resize it in order to continue using the telnet window.)

LAN Connection

Once you've tried a few commands and are satisfied that the server is running correctly, try running a telnet session from another computer on the same local area network.

You can do this by clicking the start button on the remote computer, hit run, and type "telnet xxx.xxx.xxx.xxx 5004", where xxx.xxx.xxx.xxx is the static IP address of the MCE computer to be controlled.

If you cannot connect, check that you are using the correct IP address and that the MCE computer is configured to use a static IP. If the IP address is correct, check the Windows Firewall on the MCE computer to make sure that the port (default 5004) that the MCE Control Server is configured to use is opened on the firewall. See your Microsoft Windows documentation for more information.

Internet Connection

Once you have successfully established a connection from a remote computer through the LAN, if you have a need to, you can try connecting from a computer outside of your local area network through the internet. In this scenario, it is likely that you will have to configure your hardware firewall or router to forward incoming requests for the Server and Web Image ports to the static IP address of the host Media Center Computer.

Make sure that both the host MCE computer and the client have a working connection to the internet.

MCE Control Specification

Protocol Conventions

Commands are case-insensitive. All commands and their responses are terminated with a carriage return / linefeed pair (CRLF)

Initiating control sessions

The TCP/IP client should initiate the control session with the MCE Control Server by opening a socket to the server port specified in the configuration utility (default of 5004). When a connection has been established, the server will answer with the following string.

```
Welcome to the Autonomic Home MCE Control Server version 0.9.2351.661 Beta.  
Type '?' for help or 'help <command>' for help on <command>.
```

Starting the MCE Session

The client should take steps to insure that the MCE session is started on the host computer before attempting to control the session or receive feedback.

This can be accomplished by issuing the command `GetMCEStatus`. If there is no instance currently running on the MCE computer, the server will respond as follows:

```
ReportState FamilyRoom Running=False
```

If an instance is currently running, the server will respond with a series of `ReportStatus` responses:

```
ReportState Administrator@00-00-00-00-00-00 Volume=25  
ReportState Administrator@00-00-00-00-00-00 SessionStart=FS_Home  
ReportState Administrator@00-00-00-00-00-00 Running=True
```

In addition to the `Running=True/False` token, these `ReportState` responses may include media information, volume, and transport status. These can be used to initialize the client UI. `ReportState` and `StateChanged` tokens are fully documented later in this manual.

If there is no MCE session started (i.e. `Running=False`), the client can begin a session with the command `StartMCE`.

```
Command:   StartMCE  
Response:  StartMCE OK
```

Selecting an MCE Instance

If your client will be configured so as to only control the current MCE session, this step is not necessary. The server will default to the session currently started on the MCE computer.

If the client will be controlling multiple devices, or sessions, the first command issued should browse and store available instances:

Example:

```
Command:  BrowseInstances.  
Response: BeginInstances Total=2  
          Family Room  
          XBOX360  
          EndInstances NoMore
```

In this example, there are two instances, which have been given friendly names in the configuration utility. The client may now select an instance to control, or present a selection list in its user interface to allow selection of control instance.

```
Command:  SetInstance "Family Room"  
Response: Instance=Family Room
```

Clients may change this instance as often as necessary during a session; however, **BrowseInstances** should be used to refresh the available instances as they can go offline at any time. (i.e., if the end-user turns off the extender).

To explicitly select the current instance running on the MCE host computer, use:

```
Command:  SetInstance *  
Response: Instance=*
```

List Processing

Many commands result in the server returning a list of items. Sometimes, these lists can be very long, and will frequently exceed the number of items that can be displayed on a MCCP client. The protocol includes methods for retrieving partial results, and randomly traversing sections of the list.

Lists are requested by Browse[type] commands which are fully documented in the command reference.

All list requests and messages follow the same convention:

Command: Browse[*browsetype*] [*startposition*] [*count*]

Header: Begin[*listtype*] Total=[*count*]

Items: [*itemtype*] [*field1*] [*field2*] ...
 ...
 ...

Terminator: End[*listtype*] [*More*] | [*NoMore*]

Where:

browsetype this specifies the list being requested.

startposition this specifies where the server should start returning list items.

requestcount this specified how many items should be returned.

listtype this is the type of list being sent (i.e. **BeginAlbums**, **BeginArtists**, etc..)

count the total number of items in the list. This number may be larger than the number of items that will be returned in one response, as determined by the **returncount** parameter in the **Browse** command which initiated the list.

itemtype This is the type of list item being sent. (i.e. Album, Artist, Genre, etc.)

field1, field2 The content of these fields are dependant on the type of list. See the command reference for more information.

More/NoMore Indicates the availability of more list items beyond the requested section.

Asynchronous processing

The Media Center Control Protocol is an asynchronous command protocol. This is necessary because the controlled device (MCE), can be directly manipulated by the user outside of the control of the MCE Control Server. This means that clients must be written so as to properly parse and process responses in any order that they may come in.

For example, if the command `BrowseArtists` is issued to the server, it will begin to send a list of artists. If the user stops the media transport while this list is being processed, the server will send a `StateChanged` message in the middle of the list. This is necessary in order to insure that the client can issue responsive feedback to the user, which is vitally important to the control experience:

Example

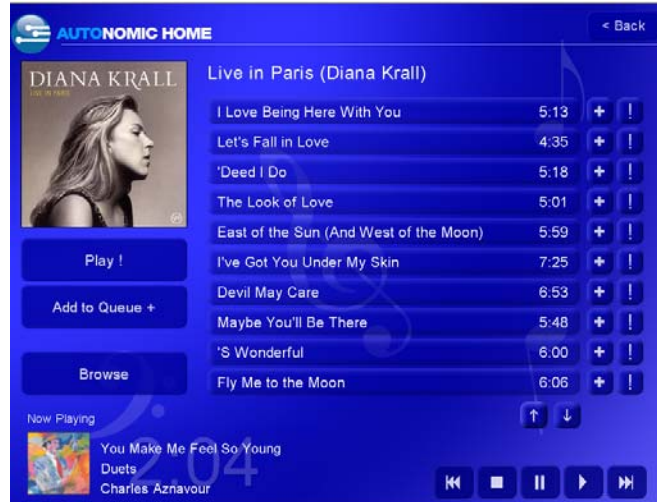
```
Command:      SetMediaFilter Genre=Jazz
Response:     MediaFilter Genre=Jazz
Command:      BrowseArtists
Response:     BeginArtists Total=6
               Artist {19ef-4880-8064-a79e51ee270c} "Brian McKnight"
               Artist {c502-437a-81f6-3cddb30059} "Frank Sinatra"
               Artist {6646-4ce2-b255-240c7b8f483a} "Tevin Campbell"
               StateChanged FamilyRoom MediaControl=Stop
               Artist {9bf8-492b-b124-6879a65d414b} "Shakatak"
               Artist {39b7-47ae-bde5-9f9bd728237a} "James Ingram"
               Artist {bd96-4d32-8fbb-75a42d099370} "George Benson"
EndArtists NoMore
```

The client must be written in a pre-emptive mode in order to properly receive and process messages from the server. Since each message is terminated with a CRLF pair, the client should continually fetch a string from the incoming buffer until a CRLF pair is encountered, search the string for tokens, process the message appropriately, issue appropriate user feedback, and then collect the next message from the buffer, and so on.

In this example, list items could be distinguished from the `StateChanged` message by the pair of spaces preceding the list item, and the unique token `Artist`.

Displaying Album Art

MCECS enables client applications or Ethernet enabled touch panels to display album cover art for any media in the library.



This is accomplished by accessing a web-server which will host the cover art as .jpg files.. This web-server is installed on the Host Media Center computer together with MCECS.

There are two URLS made available by the Autonomic web server to access cover art:

1 - [http://\[server:webport\]/albumart?instance=\[* | instance\]](http://[server:webport]/albumart?instance=[* | instance])

This address will point to a 200x200 pixel bitmap representing the cover art of the currently playing media. Some software and devices will require that dynamic images end with a valid graphics extension. In these cases, the server will allow placing “.jpg” at the end of the address.

Example: http://192.168.1.10:5005/albumart?instance=* will display the album art for the currently playing media on the MCE computer with the static IP address of 192.168.1.10, port 5005

The other URL is used to randomly access the cover art of any album by its GUID. (Globally Unique ID - See Browse... commands in the command reference). The syntax is:

2 - [http://\[server:webport\]/albumart?album=\[guid\]](http://[server:webport]/albumart?album=[guid])

Example: <http://192.168.1.10:5005/albumart?album={33432-33432-95909-33423-34430}.jpg> displays the album art for the media identified by the GUID.

Command Reference

General Commands

Banner

Command: `banner`

Response: `Welcome to the Autonomic Home MCE Control Server version 1.0
Type '?' for help or 'help <command>' for help on <command>.`

Displays the connection banner including version information.

CLS

Command: `cls`

Clears all characters on the ANSI terminal.

Exit

Command: `exit`

Ends the current session and closes the ANSI terminal.

Help

Command: `help [command]`

In the first form, displays a list of all available commands. If the optional [command] parameter is issued, detailed help will be displayed for the command.

GetVersions

Syntax: **GetVersions**

Example:

Command: **GetVersions**

Response: **BeginVersions Total=1**
 AhEhSrvr 1.0.2398.788
 EndVersions NoMore

Returns a list of Autonomic Controls component version numbers on the server.

AhEhSrvr refers to the MCECS - Autonomic Home eHome Server.

GetLicenseMessage

Syntax: **GetLicenseMessage**

Example:

Command: **GetLicenseMessage**

Response: **Licensed by Autonomic Controls, Inc to Joe Smith**
 Demo mode in progress: 12 days remaining
 Unlicensed

Returns the current license message

Time

Syntax: **Time** <format>

Example:

Command: **Time**

Response: **Time: "Saturday, June 10, 2006 4:03:43 PM"**

Echo's the current system time from the MCE computer. The optional <format> parameter can be used to change the format of the return string.

Valid format codes are:

```
"d" : 08/17/2000
"D" : Thursday, August 17, 2000
"f" : Thursday, August 17, 2000 16:32
"F" : Thursday, August 17, 2000 16:32:32
"g" : 08/17/2000 16:32
"G" : 08/17/2000 16:32:32
"m" : August 17
"r" : Thu, 17 Aug 2000 23:32:32 GMT
"s" : 2000-08-17T16:32:32
"U" :Thursday, August 17, 2000 23:32:32
```

BrowseEncodings

Syntax: `BrowseEncodings`

Example:

Command: `BrowseEncodings`

Response: `BeginEncodings Total=95
 37 "IBM EBCDIC (US-Canada)"
 437 "OEM United States"
 737 "Greek (DOS)"
 775 "Baltic (DOS)"
 850 "Western European (DOS)"
 852 "Central European (DOS)"
 ...
 ...
 861 "Icelandic (DOS)"
 862 "Hebrew (DOS)"
EndEncodings NoMore`

Allows for browsing the list of valid text encoding id's.

SetEncoding

Syntax: **SetEncoding**

Example:

Command: **SetEncoding 20105**

Response: **Encoding 20105**

Allows for browsing the list of valid text encoding id's. Encoding 20105 recommended for touchpanels.

Valid format codes are:

37 "IBM EBCDIC (US-Canada)"	1256 "Arabic (Windows)"	28591 "Western European (ISO)"
437 "OEM United States"	1257 "Baltic (Windows)"	28592 "Central European (ISO)"
500 "IBM EBCDIC (International)"	1258 "Vietnamese (Windows)"	28593 "Latin 3 (ISO)"
708 "Arabic (ASMO 708)"	10000 "Western European (Mac)"	28594 "Baltic (ISO)"
720 "Arabic (DOS)"	10004 "Arabic (Mac)"	28595 "Cyrillic (ISO)"
737 "Greek (DOS)"	10005 "Hebrew (Mac)"	28596 "Arabic (ISO)"
775 "Baltic (DOS)"	10006 "Greek (Mac)"	28597 "Greek (ISO)"
850 "Western European (DOS)"	10007 "Cyrillic (Mac)"	28598 "Hebrew (ISO-Visual)"
852 "Central European (DOS)"	10010 "Romanian (Mac)"	28599 "Turkish (ISO)"
855 "OEM Cyrillic"	10017 "Ukrainian (Mac)"	28603 "Estonian (ISO)"
857 "Turkish (DOS)"	10021 "Thai (Mac)"	28605 "Latin 9 (ISO)"
858 "OEM Multilingual Latin I"	10029 "Central European (Mac)"	29001 "Europa"
860 "Portuguese (DOS)"	10079 "Icelandic (Mac)"	38598 "Hebrew"
861 "Icelandic (DOS)"	10081 "Turkish (Mac)"	
862 "Hebrew (DOS)"	10082 "Croatian (Mac)"	
863 "French Canadian (DOS)"	20105 "Western European (IA5)"	
864 "Arabic (864)"	20106 "German (IA5)"	
865 "Nordic (DOS)"	20107 "Swedish (IA5)"	
866 "Cyrillic (DOS)"	20108 "Norwegian (IA5)"	
869 "Greek, Modern (DOS)"	20127 "US-ASCII"	
870 "IBM EBCDIC"	20269 "ISO-6937"	
874 "Thai (Windows)"	20273 "IBM (Germany)"	
875 "IBM (Greek Modern)"	20277 "IBM (Denmark-Norway)"	
1026 "IBM (Turkish Latin-5)"	20278 "IBM (Finland-Sweden)"	
1047 "IBM Latin-1"	20280 "IBM (Italy)"	
1140 "IBM (US-Canada-Euro)"	20284 "IBM (Spain)"	
1141 "IBM (Germany-Euro)"	20285 "IBM (UK)"	
1142 "IBM (Denmark-Norway-Euro)"	20290 "IBM (Japanese katakana)"	
1143 "IBM (Finland-Sweden-Euro)"	20297 "IBM (France)"	
1144 "IBM (Italy-Euro)"	20420 "IBM (Arabic)"	
1145 "IBM (Spain-Euro)"	20423 "IBM (Greek)"	
1146 "IBM (UK-Euro)"	20424 "IBM (Hebrew)"	
1147 "IBM (France-Euro)"	20833 "IBM (Korean Extended)"	
1148 "IBM (International-Euro)"	20838 "IBM (Thai)"	
1149 "IBM (Icelandic-Euro)"	20866 "Cyrillic (KOI8-R)"	
1250 "Central European (Windows)"	20871 "IBM (Icelandic)"	
1251 "Cyrillic (Windows)"	20880 "IBM (Cyrillic Russian)"	
1252 "Western European (Windows)"	20905 "IBM (Turkish)"	
1253 "Greek (Windows)"	20924 "IBM Latin-1"	
1254 "Turkish (Windows)"	21025 "IBM (Serbian-Bulgarian)"	
1255 "Hebrew (Windows)"	21866 "Cyrillic (KOI8-U)"	

MCE Instance Commands

BrowseInstances

Syntax: **BrowseInstances**

Response Syntax:

Header: **BeginInstances**
Items: **[Instance]**
 ...
 ...
Terminator: **EndInstances NoMore**

Example:

Command: **BrowseInstances**

Response: **BeginInstances Total=2**
 FamilyRoom
 XBOX
 EndInstances NoMore

Returns a list of current instances. If friendly names have been created in the configuration utility, they will be used, otherwise, the server will return :

```
sessionname@NIC Address as in Administrator@00-00-00-00-00-00
```

Accounts on the MCE host will always be listed. Extender sessions will only be listed when the extender is on and in the Media Center shell. Note that XBOX 360 extenders will not be listed if they are in game or console mode. To start the XBOX 360 in extender mode, press the green button on the XBOX 360 remote control to start in Media Center mode.

SetInstance

Syntax: **SetInstance** *string*[**instance_id**]

Example:

Command: **SetInstance** "Family Room"

Response: **Instance=FamilyRoom**

Selects a specific Media Center Instance for further commands and events

If no MCE Instance is selected via this command or if [**instance_id**]=* then the "Current" instance is the instance running on the console.

Missing value for [*instance_id*] will simply respond current value.

See the **BrowseInstances** Command for information on how to enumerate the current instances.

Interfacing with the Media Center Shell

MsgBox

Syntax: **MsgBox** <id> <caption> <message> <buttons> <timeout> <image>

Example:

```
Command:        MsgBox        1  
                 "Garage Door"  
                 "The garage door is open, would it closed?"  
                 "Yes;No"  
                 "20"
```

Response: **MsgBox** 1 1



Displays the quote enclosed string in a media center dialog box and waits for user response.

- <id> integer, question id – this will help you to match the response with a question.
- <caption> message Caption
- <message> message for display
- <buttons> semicolon delimited button texts
- <timeout> timeout in seconds defaults to 5.
- <image> UNC path to the PNG-format image to display in the dialog box.

The server will respond in the format **Msgbox** [*id*] [*Button*], where the *id* is the integer supplied in the **MsgBox** command and *Button* is the index of the button that the user selected. (1 based). If the message box times out, no response will be sent.

Media Center Feedback

GetMCEStatus

Syntax: `GetMCEStatus`

Example:

Command: `GetMCEStatus`

Response: `ReportState FamilyRoom TrackName=A Foggy Day
ReportState FamilyRoom MediaControl=Stop
ReportState FamilyRoom SessionStart=StreamingContentAudio
ReportState FamilyRoom Volume=25
ReportState FamilyRoom TrackTime=1
ReportState FamilyRoom TrackDuration=144
ReportState FamilyRoom TotalTracks=50
ReportState FamilyRoom TrackNumber=8
ReportState FamilyRoom RepeatSet=False
ReportState FamilyRoom CD=False
ReportState FamilyRoom ArtistName=Frank Sinatra
ReportState FamilyRoom MediaName=Duels/Duels II
ReportState FamilyRoom Shuffle=True
ReportState FamilyRoom Running=True`

See ReportState Message

GetMCEStatus returns a list of all parameters that are typically sent with a StateChanged message. This can be used to prime the IP client's feedback status.

This function should also be called before attempting to command the MCE interface to insure that the MCE shell is running. This can be determined by the `Running=True|False` token.

The messages returned in response to this command differ from event driven `stateChanged` messages only in the leading token `ReportState`. Clients can use these tokens to distinguish between an event that has just occurred and a requested update. The rest of the response can be handled by the same parsing routine.

SubscribeEvents

Syntax: **SubscribeEvents** *Boolean String* <Events>

Example:

Command: **SubscribeEvents True**

Response: **Events=True**

Turns event messages on or off. (Such as track information, track progress, transport feedback, etc.) If <Events> is omitted, the server will return the current state.

StateChanged Message

Syntax: `StateChanged <instance> <name>=<value>`

Example:

Command: `N/A (see SubscribeEvents)`

Response: `StateChanged FamilyRoom MediaControl=Play`
 `StateChanged FamilyRoom TrackTime=77`
 `StateChanged FamilyRoom TrackTime=78`
 `StateChanged FamilyRoom TrackTime=79`
 `StateChanged FamilyRoom TrackTime=80`
 `StateChanged FamilyRoom TrackTime=81`
 `StateChanged FamilyRoom MediaControl=Stop`

The StateChanged token indicates that the Media Center Control Server is sending a status update to the client. The =<value> field will only be sent if appropriate.

<instance> Indicates the instance of the event being reported.
<name> Indicates the name of the event being reported.
<value> Indicates any value associated with the event.

Valid Name / Values:

ArtistName	The name of the artist of the currently playing media.
CallingPartyName	Caller ID, the name of the calling party.
CallingPartyNumber	Caller ID, the number of the calling party.
CD	CD Playback initiated
CurrentPicture	The name of the current picture displayed (MyPictures)
DiscWriter_ProgressPercentageChanged	Update on the progress of a CD/DVD recording operation
DiscWriter_ProgressTimeChanged	Update on the progress of a CD/DVD recording operation
DiscWriter_SelectedFormat	Selected recording format for a CD/DVD recording operation.
DiscWriter_Start	CD/DVD recording operation has begun.
DiscWriter_Stop	CD/DVD recording operation has concluded.
DVD	DVD Playback has started
Ejecting	The CD/DVD is ejecting
Error	An error occurred in the MCE shell
GuideLoaded	Downloaded a new guide.
MediaControl=Rewind3	Rewind speed 3 initiated
MediaControl=FF1	Fast Forward speed 1 (slow) initiated
MediaControl=FF2	Fast forward speed 2 (medium) initiated
MediaControl=FF3	Fast forward speed 3 (fast) initiated
MediaControl=NextFrame	The next frame transport control was issued.

MediaControl=Pause	Pause transport command issued
MediaControl=Play	Play transport command issued
MediaControl=PrevFrame	The previous frame transport control was issued.
MediaControl=Rewind1	Rewind speed 1 initiated
MediaControl=Rewind2	Rewind speed 2 initiated
MediaControl=SkipNext	The track was skipped forward
MediaControl=SkipPrev	The track was skipped backwards
MediaControl=SlowMotion1	Slow Motion playback (speed 1) has begun
MediaControl=SlowMotion2	Slow motion playback (speed 2) has begun
MediaControl=SlowMotion3	Slow Motion playback (speed 3) has begun
MediaControl=Stop	Stop transport command issued
MediaName	The name of the currently playing media (all media types)
MediaTime	The total duration of the currently playing media (video, music, or TV)
MediaType	The type of the currently playing media
Navigation=Extensibility	Navigating to a hosted HTML application.
Navigation=FS_DVD	Navigating to Play DVD, or the DVD inset was selected.
Navigation=FS_Home	Navigating to Media Center Start Page.
Navigation=FS_TV	Navigating to My TV, or the TV inset was selected.
Navigation=Guide	Navigating to Guide.
Navigation=Music	Navigating to My Music, or the music inset was selected.
Navigation=Photos	Navigating to My Pictures.
Navigation=Radio	Navigating to My Radio.
Navigation=RecordedShows	Navigating to Recorded Shows or scheduled recording pages
Navigation=Unknown	Unknown Media Center status.
Navigation=Videos	Navigating to My Videos, or the video inset was selected.
ParentalAdvisoryRating	The MPAA rating of the current media
PhoneCall	Incoming phone call event.
Radio	The radio has been activated
RadioFrequency	The frequency of the current radio station
Recording	Status of record mode has changed
RepeatSet	Status of repeat mode changed
Running	The MCE Shell is running
SessionEnd	The MCE shell has ended
SessionStart	The MCE shell has started
Shuffle	Status of shuffle mode changed
StreamingContentAudio	Playback of streaming audio has begun
StreamingContentVideo	Playback of streaming video has begun
TitleNumber	The track number of the current media
TotalTracks	The total number of tracks in the current media set (album)
TrackDuration	The total duration of the current track in seconds
TrackName	The name of the current track
TrackNumber	The number of the current track
TrackTime	The progress of track playback in seconds
TransitionTime	The transition time (between pictures in slideshow)
Visualization	The name of the current visualization
Volume	The current volume level (master MCE volume level)

ReportState Message

Syntax: `ReportState <instance> <name>=<value>`

Example:

Command: See `GetMCEStatus` command

Response: `StateChanged FamilyRoom MediaControl=Play`
 `StateChanged FamilyRoom TrackTime=77`
 `StateChanged FamilyRoom TrackTime=78`
 `StateChanged FamilyRoom TrackTime=79`
 `StateChanged FamilyRoom TrackTime=80`
 `StateChanged FamilyRoom TrackTime=81`
 `StateChanged FamilyRoom MediaControl=Stop`

The ReportState message is sent in response to a GetMCEStatus command.

<instance> Indicates the instance of the event being reported.
<name> Indicates the name of the event being reported.
<value> Indicates any value associated with the event.

Set StateChanged command for valid name/value pairs.

Media Center Interface Navigation

Navigate

Syntax: **Navigate <screen>**

Example:

Command: **Navigate MyPictures**

Response: **StateChanged FamilyRoom Navigation=Pictures**

Instructs the Media Center shell on the host computer to navigate to the specified screen. If the client is subscribed to events (see SubscribeEvents), a StateChanged message will be sent confirming the navigation.

Valid values for screen:

FMRadio	Sets Media Center to FM Radio
InternetRadio	Sets Media Center to Internet Radio
LiveTV	Sets Media Center to Live Television
MorePrograms	Sets Media Center to More Programs
MusicAlbums	Sets Media Center to Music Albums
MusicArtists	Sets Media Center to Music Artists
MusicSongs	Sets Media Center to Music / Songs
MyMusic	Sets Media Center to My Music
MyPictures	Sets Media Center to My Pictures
MyTV	Sets Media Center to My TV
MyVideos	Sets Media Center to MY Videos
RecordedTV	Sets Media Center to Recorded TV
RecorderStorageSettings	Sets Media Center to RecorderStorageSettings
ScheduledTVRecordings	Sets Media Center to Scheduled TV Recordings
SlideShow	Sets Media Center to Slide Show
Start	Sets Media Center to Start Page
TVGuide	Sets Media Center to TV Guide
Visualizations	Sets Media Center to Visualizations
PhotoDetails	Sets Media Center to Photo Details
SlideShow	Initiates Media Center as Slide Show
SlideShowSettings	Sets Media Center Slide Show Settings

Transport Control

Transport Commands

Syntax: <command>

Response Syntax: <command> OK

Example:

Command: **Play**

Response: **Play OK**

Issues the specified transport control

Valid values for <command>:

Play	Instructs Media Player to PLAY the media transport.
Stop	Instructs Media Player to STOP the media transport.
Pause	Instructs Media Player to PAUSE the media transport.
SkipNext	Commands Media Player to move to the next song in the queue. (with wraparound)
SkipPrev	Commands Media Player to move to the previous song in the queue (with wraparound)
Random [On Off]	Turns Random Mode on or Off
Repeat [On Off]	Turns Repeat Mode on or Off

Browse Media Commands

BrowseAlbums

Syntax: `BrowseAlbums <start> <reqcount>`

Response Syntax:

Header: `BeginAlbums Total=[count]`

Items: `Album [GUID] [Name] ...`

`...`

`...`

Terminator: `EndAlbums [More]| [NoMore]`

Example:

Command: `BrowseAlbums 11 10`

Response: `BeginAlbums Total=170
Album {a7ca-47a1-bc2b-f4927bbf2ad8} "Chopin: Ballades & Scherzos"
Album {3ce8-4aeb-99f8-43a1a33d30cc} "Chopin: The Complete Nocturnes"
Album {f6f4-4416-a1c3-af2b7bba0e9c} "Cieli Di Toscana"
Album {6c61-454e-9f0e-0a28ae8dde95} "Clapton Chronicles"
Album {5147-4b26-b31f-720230af4278} "Claude Debussy: Preludes"
Album {ebaa-4b38-b4d6-c18d97ff962e} "Come Away With Me"
Album {93a1-4c0b-b8c1-51c3447f05c8} "Come on Over [International]"
Album {0434-49e4-8c2b-8887edeb6258} "Cry Like a Rainstorm"
Album {f8b8-422a-94e4-923e591fb6b2} "The Dance"
Album {bc76-4322-8feb-7cf72fc6230b} "Dances with Wolves"
EndAlbums More`

Allows browsing the media library belonging to the current instance.

`<start>` specifies where to start the listing.

`<reqcount>` specifies the total items requested

`<GUID>` a globally unique ID used for playback commands and further browsing.

`<name>` the name of the album

If `<start>` and `<reqcount>` are omitted, all albums that match the current filter will be returned. (see `SetMediaFilter`, `GetMediaFilter`).

Refer to **List Processing** and **Asynchronous Processing** topics at the beginning of the command reference.

BrowseArtists

Syntax: `BrowseArtists <start> <reqcount>`

Response Syntax:

Header: `BeginArtists Total=[count]`
Items: `Artist [GUID] [Name] ...`
`...`
`...`
Terminator: `EndArtists [More]|[NoMore]`

Example:

Command: `BrowseArtists 1 10`

Response: `BeginArtists Total=344`
`Artist {7d9761cb-ea80-43dd-b63e-90a0b6bd8017} "Unknown"`
`Artist {ef504364-e0ac-4f1a-a276-09dc087bfe6e} "NSYNC"`
`Artist {e28d9be3-6cbd-4678-9205-eed6b90f714e} "3rd Party"`
`Artist {4930f19f-f942-4017-9293-7618c14ef94c} "5ive"`
`Artist {9ed91b23-153d-4857-bd35-b72d3f83d8e9} "7 Mile"`
`Artist {0af5a33d-d486-4d06-9bee-7f2b0cead68f} "112"`
`Artist {5279b9bc-0427-49dc-a1ff-f5e8d17f5717} "Aaron Copland"`
`Artist {87c1d18b-079e-4d10-8564-0a7a5e49b65d} "Aaron Hall"`
`Artist {902e1012-710d-46eb-861d-7a75ca81b96e} "Aaron Neville"`
`Artist {bd3c4dde-b07c-4123-908d-f00d76311ae8} "Abbey Simon"`
`EndArtists More`

Allows for browsing the media library belonging to the current instance.

`<start>` specifies where to start the listing.
`<reqcount>` specifies the total items requested

`<GUID>` a globally unique ID used for playback commands and further browsing.
`<name>` the name of the artist

If `<start>` and `<reqcount>` are omitted, all artists that match the current filter will be returned. (see `SetMediaFilter`, `GetMediaFilter`).

Refer to **List Processing** and **Asynchronous Processing** topics at the beginning of the command reference.

BrowseGenres

Syntax: `BrowseGenres <start> <reqcount>`

Response Syntax:

Header: `BeginGenres Total=[count]`

Items: `Genre [GUID] [Name] ...`

...

...

Terminator: `EndGenres [More]| [NoMore]`

Example:

Command: `BrowseGenres 11 10`

Response: `BeginGenres Total=10
Genre {90c4469e-fa42-4b3d-b9e1-88f0bfe02df9} "Classical"
Genre {1f75c29d-cd2f-4dde-881b-6ba855222afb} "Country"
Genre {161ebefe-bc24-41d4-8e97-8b338f93ec05} "Dance / Electronic"
Genre {69d2c275-e25f-4b7c-a2ca-e2975960636c} "Hip-Hop"
Genre {e71ed659-e54b-4c5d-99c6-35a1487727c7} "Jazz"
Genre {33d4d6d6-a3fa-4797-80c9-755ac5c79eae} "New Age"
Genre {ba3d9dff-55f1-41e3-9660-3ee7054c9a52} "Pop"
Genre {2e2c338d-24d2-4860-8be6-ccbe8fdfa119} "R&B"
Genre {9c5b5efe-3934-437b-8a83-7bb903be6d25} "Rock"
Genre {d61f8edd-d7bf-4d79-8aa5-a91c857ffd2c} "Soundtrack"
EndGenres NoMore`

Allows browsing the media library belonging to the current instance.

`<start>` specifies where to start the listing.

`<reqcount>` specifies the total items requested

`<GUID>` a globally unique ID used for playback commands and further browsing.

`<name>` the name of the genre

If `<start>` and `<reqcount>` are omitted, all genres that match the current filter will be returned. (see `SetMediaFilter`, `GetMediaFilter`).

Refer to **List Processing** and **Asynchronous Processing** topics at the beginning of the command reference.

BrowseNowPlaying

Syntax: `BrowseNowPlaying <start> <reqcount>`

Response Syntax:

Header: `BeginNowPlaying Total=[count]`
Items: `Title [GUID] [Name] [time]`
...
...
Terminator: `EndNowPlaying [More]|[NoMore]`

Example:

Command: `BrowseNowPlaying 1 10`

Response:

```
BeginNowPlaying Total=98
  Title {3216-457f-87c2-b5da6541b895} "A Foggy Day" "00:02:25"
  Title {92ca-4ccf-983c-7b2760cca26d} "All or Nothing at All" "00:04:00"
  Title {4306-8067-6154c7eb5d7b} "All the Way" "00:03:54"
  Title {f3f7-4d65-b616-6d98d3e442a6} "All the Way/One for My Baby" "00:06:04"
  Title {40bc-48c6-be87-4586684d95c1} "Bewitched" "00:03:32"
  Title {c950-4521-acb7-026c9dc0ed8b} "Come Fly with Me" "00:03:09"
  Title {13a3-4f30-8980-4a6ad5fa9569} "Come Rain or Come Shine" "00:04:05"
  Title {c24f-42d9-a19e-98826e66c747} "Embraceable You" "00:03:46"
  Title {94ed-4b7d-b419-ea06d0d21436} "Fly Me to the Moon" "00:03:07"
  Title {104f-46e6-8fc4-6371aa86e14a} "Fly Me to the Moon" "00:02:32"
EndNowPlaying More
```

Allows browsing the queue for the current instance.

<start> specifies where to start the listing.
<reqcount> specifies the total items requested

<GUID> a globally unique ID used for playback commands and further browsing.
<name> the name of the track
<time> the length of the track

If *<start>* and *<reqcount>* are omitted, all titles in the queue will be returned.

Refer to **List Processing** and **Asynchronous Processing** topics at the beginning of the command reference.

BrowsePlaylists

Syntax: `BrowsePlaylists <start> <reqcount>`

Response Syntax:

Header: `BeginPlaylists Total=[count]`

Items: `Playlist [GUID] [Name] ...`
...
...

Terminator: `EndPlaylists [More]|[NoMore]`

Example:

Command: `BrowsePlaylists`

Response: `BeginPlaylists Total=5`
`Playlist {90e56f8e-c900-44fc-8cd7-fdac81b6f215} "Diana"`
`Playlist {5fd3175e-2688-4617-bcf7-575c71b1e0bc} "Napster Tracks"`
`Playlist {efdd1f28-63ff-4dfd-b244-b2f6868af4a6} "Popular"`
`Playlist {c386107f-5e7e-45c4-a270-42c6de8aeb84} "Soft Background"`
`Playlist {039a8699-506f-4567-a0ac-90fc2778a2f5} "The Movies!"`
`EndPlaylists NoMore`

Allows browsing play lists in the current instance.

<start> specifies where to start the listing.

<reqcount> specifies the total items requested

<GUID> a globally unique ID used for playback commands and further browsing.

<name> the name of the play list

If *<start>* and *<reqcount>* are omitted, all play lists that match the current filter will be returned. (see `SetMediaFilter`, `GetMediaFilter`).

Refer to **List Processing** and **Asynchronous Processing** topics at the beginning of the command reference.

BrowseTitles

Syntax: `BrowseTitles <start> <reqcount>`

Response Syntax:

Header: `BeginTitles Total=[count]`
Items: `Title [GUID] [Name] [time]`
...
...
Terminator: `EndTitles [More]|[NoMore]`

Example:

Command: `BrowseTitles 1 10`

Response:

```
BeginTitles Total=98
  Title {3216-457f-87c2-b5da6541b895} "A Foggy Day" "00:02:25"
  Title {92ca-4ccf-983c-7b2760cca26d} "All or Nothing at All" "00:04:00"
  Title {4306-8067-6154c7eb5d7b} "All the Way" "00:03:54"
  Title {f3f7-4d65-b616-6d98d3e442a6} "All the Way/One for My Baby" "00:06:04"
  Title {40bc-48c6-be87-4586684d95c1} "Bewitched" "00:03:32"
  Title {c950-4521-acb7-026c9dc0ed8b} "Come Fly with Me" "00:03:09"
  Title {13a3-4f30-8980-4a6ad5fa9569} "Come Rain or Come Shine" "00:04:05"
  Title {c24f-42d9-a19e-98826e66c747} "Embraceable You" "00:03:46"
  Title {94ed-4b7d-b419-ea06d0d21436} "Fly Me to the Moon" "00:03:07"
  Title {104f-46e6-8fc4-6371aa86e14a} "Fly Me to the Moon" "00:02:32"
EndTitles More
```

Allows browsing titles in the media library of current instance.

<start> specifies where to start the listing.
<reqcount> specifies the total items requested

<GUID> a globally unique ID used for playback commands and further browsing.
<name> the name of the track
<time> the length of the track

If *<start>* and *<reqcount>* are omitted, all genres that match the current filter will be returned. (see `SetMediaFilter`, `GetMediaFilter`).

Refer to **List Processing** and **Asynchronous Processing** topics at the beginning of the command reference.

Play Media Commands

PlayAlbum

Syntax: **PlayAlbum** [**guid**] [**enqueue**]

Example:

Command: **PlayAlbum** {**ab3794df-30a8-4a19-b5cf-c75740743ffa**} **True**

Response: **PlayAlbum OK**

Plays all tracks in the specified Album. The <guid> resource must be obtained with a BrowseAlbums command.

<*guid*> a globally unique ID obtained with BrowseAlbums.

<*enqueue*> If “true”, the tracks will be added to the queue without interrupting playback
 If “false”, the queue will be cleared before the tracks are added.

If there are no songs in the current queue, or if <enqueue> is “true”, then playback of the first track will begin automatically.

PlayArtist

Syntax: `PlayArtist [guid] [enqueue]`

Example:

Command: `PlayArtist {ab3794df-30a8-4a19-b5cf-c75740743ffa} True`

Response: `PlayArtist OK`

Plays all tracks of the specified Artist. The <guid> resource must be obtained with a BrowseArtists command.

<guid> a globally unique ID obtained with **BrowseArtists**.
<enqueue> If “true”, the tracks will be added to the queue without interrupting playback
If “false”, the queue will be cleared before the tracks are added.

If there are no songs in the current queue, or if <enqueue> is “true”, then playback of the first track will begin automatically.

PlayGenre

Syntax: **PlayGenre** [**guid**] [**enqueue**]

Example:

Command: **PlayGenre** {**ab3794df-30a8-4a19-b5cf-c75740743ffa**} **True**

Response: **PlayGenre OK**

Plays all tracks of the specified Genre. The <guid> resource must be obtained with a BrowseGenres command.

<*guid*> a globally unique ID obtained with **BrowseGenres**.

<*enqueue*> If “true”, the tracks will be added to the queue without interrupting playback
 If “false”, the queue will be cleared before the tracks are added.

If there are no songs in the current queue, or if <enqueue> is “true”, then playback of the first track will begin automatically.

PlayPlaylist

Syntax: **PlayPlaylist** [guid] [enqueue]

Example:

Command: **PlayPlaylist** {ab3794df-30a8-4a19-b5cf-c75740743ffa} **True**

Response: **PlayPlaylist** OK

Plays all tracks of the specified Playlist. The <guid> resource must be obtained with a BrowsePlaylists command.

<*guid*> a globally unique ID obtained with **BrowsePlaylists**.
<*enqueue*> If “true”, the tracks will be added to the queue without interrupting playback
 If “false”, the queue will be cleared before the tracks are added.

If there are no songs in the current queue, or if <enqueue> is “true”, then playback of the first track will begin automatically.

PlayTitle

Syntax: **PlayTitle** [**guid**] [**enqueue**]

Example:

Command: **PlayTitle** {**ab3794df-30a8-4a19-b5cf-c75740743ffa**} **True**

Response: **PlayTitle OK**

Plays all tracks of the specified Title. The <guid> resource must be obtained with a BrowseTitles command.

<*guid*> a globally unique ID obtained with **BrowseTitles**.

<*enqueue*> If “true”, the tracks will be added to the queue without interrupting playback
 If “false”, the queue will be cleared before the tracks are added.

If there are no songs in the current queue, or if <enqueue> is “true”, then playback of the first track will begin automatically.

JumpToNowPlayingItem

Syntax: `JumpToNowPlayingItem [guid] | [index]`

Example:

Command: `JumpToNowPlayingItem {ab3794df-30a8-4a19-b5cf-c75740743ffa}`
`JumpToNowPlayingItem 10`

Response: `JumpToNowPlayingItem OK`

Jumps directly to a title in the now playing list and begins playback.

Form 1 `JumpToNowPlayingItem [guid]`

Jumps to the title specified by [guid]. The [guid] resource is provided in response to the Browse commands.

Form 2 `JumpToNowPlayingItem [index]`

Jumps to the title specified by [index], which is the 1 based count from the top of the queue.

RemoveNowPlayingItem

Syntax: `RemoveNowPlayingItem [guid] | [index]`

Example:

Command: `RemoveNowPlayingItem {ab3794df-30a8-4a19-b5cf-c75740743ffa}`
 `RemoveNowPlayingItem 10`

Response: `RemoveNowPlayingItem OK`

Jumps directly to a title in the now playing list and begins playback.

Form 1 `RemoveNowPlayingItem [guid]`

Removes the title specified by [guid] from the now playing queue. The [guid] resource is provided in response to the Browse commands.

Form 2 `RemoveNowPlayingItem [index]`

Removes the title specified by [index] from the now playing queue, which is the 1 based count from the top of the queue.

Filter Media Library Commands

SetMediaFilter

Syntax: `SetMediaFilter [tag]=([guid] | search=[string]) | [searchstring] | Clear`

Example:

Command: `SetMediaFilter Album={75ecb109-df1b-4e0f-8cbb-584017ef28da}
SetMediaFilter Artist="Peter Frampton"
SetMediaFilter Search="*Diana*"
SetMediaFilter Clear`

Response: `MediaFilter Album={75ecb109-df1b-4e0f-8cbb-584017ef28da}`

SetMediaFilter Filters future list requests. Text strings may be substituted for [guid], but this practice is not recommended as there is no guarantee that the text string will be unique within the media library, in which case the server will return the first matching entry.

Issuing successive *SetMediaFilter* commands are additive to the filter. This is useful for providing users with a browsing interface with drill down capabilities.

Form 1 `SetMediaFilter [tag] = [guid]`

The [guid] resource must be obtained from one of the Browse commands. [guid] strings are not guaranteed to persist across sessions.

Valid Values for [tag] are: Artist, Album, Genre, Playlist, or Title

Form 2 `SetMediaFilter [tag] = [string]`

Finds exact matches for tag=string. Case sensitive.

Form 3 `SetMediaFilter Search=[searchstring]`

Entering a search string will filter all subsequent Browse commands to items matching the string. The [*] wildcard character is allowed, so `"*Diana*"`, will find all items with the word "Diana" in them, while `"Diana*"` will find all items that *begin* with "Diana". Not Case Sensitive.

Form 4 `SetMediaFilter Clear`

Accumulated filters can be cleared with a single `SetMediaFilter Clear` command

PushMediaFilter

Syntax: `PushMediaFilter [tag]=[guid] | [clear]`

Example:

Command: `PushMediaFilter Album={75ecb109-df1b-4e0f-8cbb-584017ef28da}`

Response: `MediaFilterStack "Popular / On the 6"`

PushMediaFilter pushes a media filter onto a filter stack for future list requests. Unlike *SetMediaFilter*, [guid] resources must be used with this command.

Issuing successive SetMediaFilter commands are additive to the filter. This is useful for providing users with a browsing interface with drill down capabilities.

Form 1 `PushMediaFilter [tag] = [guid]`

The [guid] resource must be obtained from one of the Browse commands. [guid] strings are not guaranteed to persist across sessions.

Valid Values for [tag] are: Artist, Album, Genre, Playlist, or Title

Form 2 `PushMediaFilter [searchstring]`

Entering a search string will filter all subsequent Browse commands to items matching the string. The [*] wildcard character is allowed, so `"*Diana*`", will find all items with the word "Diana" in them, while `"Diana*"` will find all items that *begin* with "Diana"

PopMediaFilter

Syntax: **PopMediaFilter**

Example:

Command: **PopMediaFilter**

Response: **MediaFilterStack "Popular"**

PopMediaFilter pops a media filter off the filter stack for future list requests.

Issuing successive *PopMediaFilter* commands will remove filters from the stack one at a time. This is useful when used together with *PushMediaFilter* to incorporate “Browse Back” functionality in a user interface.

When successful, the response will include the text based representation of the current media filter stack. A return value of “**MediaFilterStack**” with no text afterwards indicates that the stack is empty and no filters are applied.

Accumulated filters can be cleared at once with a single **SetMediaFilter Clear** command and is recommended if the desired effect is to completely clear the filter stack.

GetMediaFilter

Syntax: **GetMediaFilter**

Response Syntax:

Header: **BeginMediaFilters Total=[count]**
Items: **BeginMediaFilters [tag]=[guid] | [name]**
 ...
 ...
Terminator: **EndMediaFilters [More]|[NoMore]**

Example:

Command: **GetMediaFilter**

Response: **BeginMediaFilters Total=2**
 Genre="e71ed659-e54b-4c5d-99c6-35a1487727c7"
 Artist="587bea39-b3b2-4771-baa3-f61a45419d4d"
 EndMediaFilters NoMore

Returns a list of currently applied media filters.

The filters will be expressed as [guid]s or [name]s according to how they were specified with **SetMediaFilter**

IR Commands

SendKeys

Syntax: **SendKeys** [*irkey*]

Example:

Command: **SendKeys** 1

Response: **NA**

Sends the specified IR key to the server to be executed on the MCE instance as though the user had send the command from the hand held remote control.

These commands are fundamentally different from other commands in the protocol that seemingly overlap. The action taken by MCE in response to an **SendKeys** command will be determined by the current MCE application.

For example, an MCE add-in application might use the next and previous buttons on the remote to allow the user to navigate a list. In this example, issuing a **SendKeys Replay** would be interpreted by the add-in as a list navigation and would have a different effect than issuing the **SkipPrev** command listed in the **Transport** section of this protocol, which will always move to the next track in the current media queue.

Valid values for [*irkey*]

Navigation	Transport:	AV and Power Control	Data Entry:
Home	Play	Volume+	0
Up	Pause	Volume-	1
Down	Stop	Chan/Page+	2
Left	Record	Chan/Page-	3
Right	FastForward	Mute	4
OK	Rewind	DVDMenu	5
Back	Skip	Standby	6
Details	Replay		7
Guide			8
Jump			9
MoreInfo			Clear
			Enter