MusicMaster for Windows
Nexus Server API

Updated: December 4, 2015
API Version: 5008

Introduction:

The MusicMaster Nexus Server provides an API that can be used to interface third-party software products with MusicMaster in a version-independent manner, and without the risk of direct database access. Section I of this document will describe the general concepts of this API and explain how to make a connection to a server and use the API. Sections II contains a reference of the available API functions and command messages that can be used. Section III documents revisions made to the API.

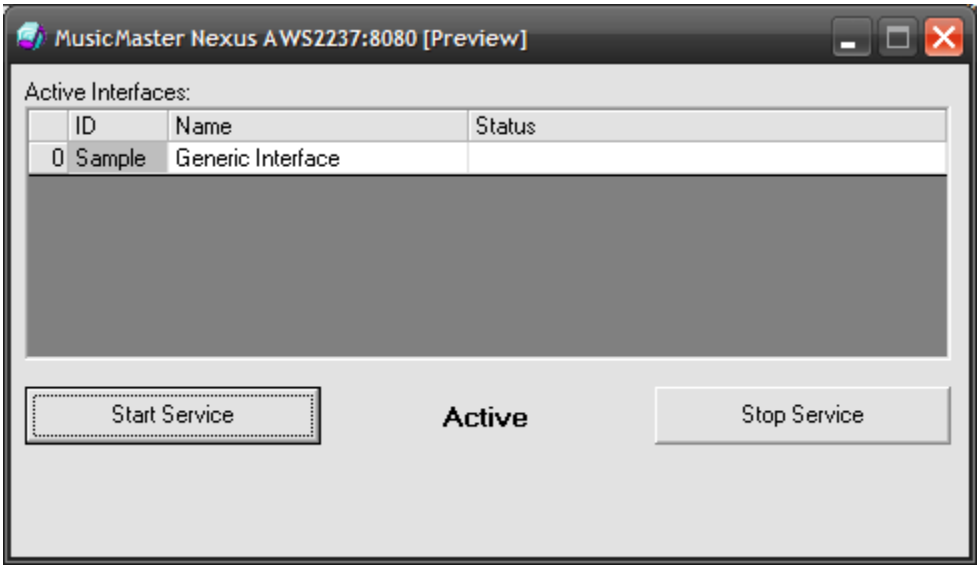
Notice:

The Nexus API specification is provided as-is and not guaranteed for any specific purpose. The Nexus API is subject to change at any time without notice. Every attempt is made to assure that future versions of this API are backward-compatible with earlier versions.

Section I: General Concepts

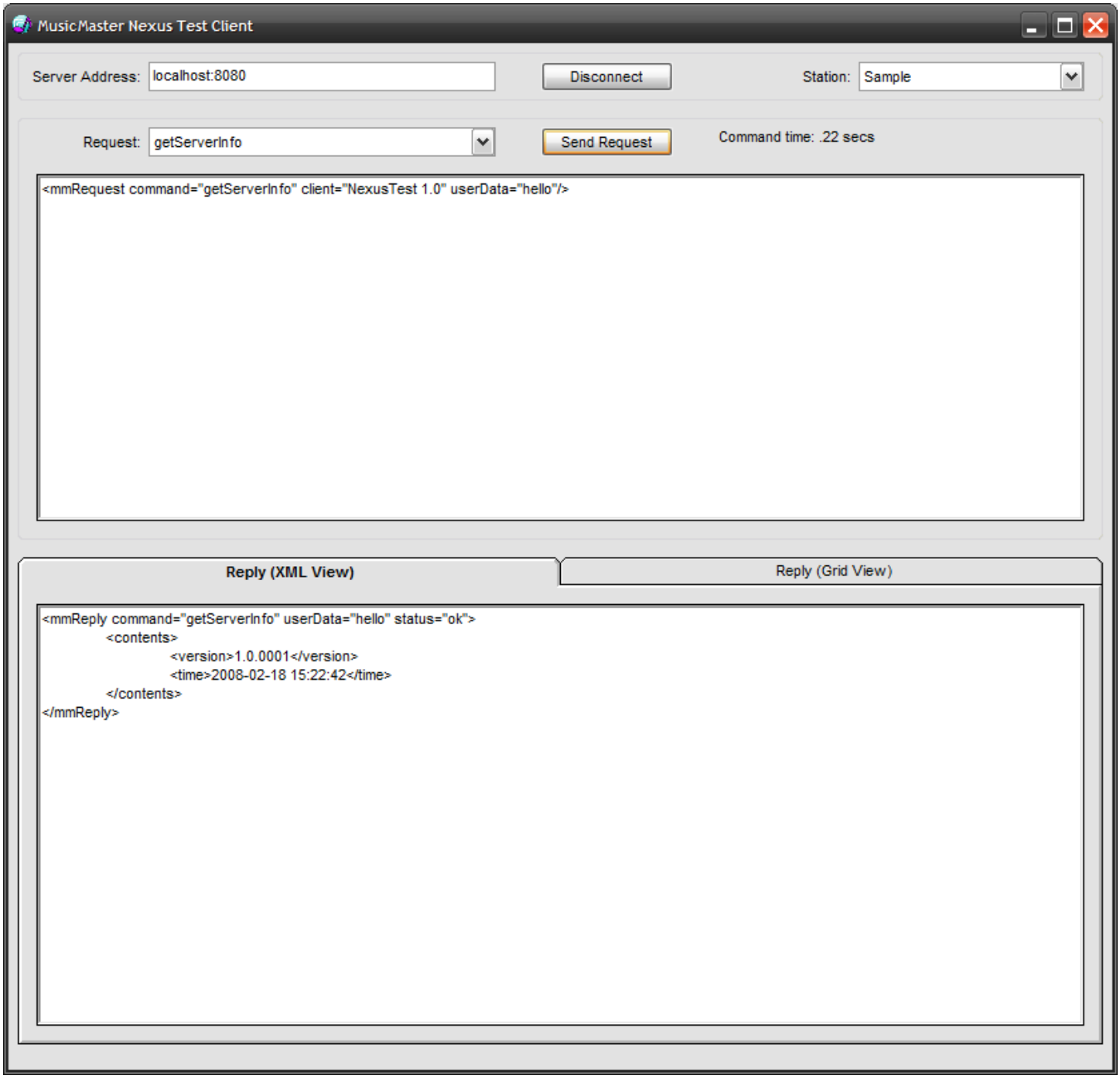
MusicMaster Server:

MusicMaster 4.0 now includes MusicMaster Nexus Server. This is an HTTP Server application that can respond to HTTP Post requests on the local machine, or across a network. In addition, each instance of the server can serve requests for one or more MusicMaster databases.



Test / Demonstration Client:

The MMServerTest.exe application can be used to experiment with the MMServer interface. It allows you to specify the address and port of the server to connect to, and once connected allows you to select a database to query. Additional controls let you test various server functions:



Configuring the Nexus Server:

The current version of MMServer must be operated on a computer that already has a copy of MusicMaster 4.0 SR-9 or later installed. The folder that contains MMServer.exe must also contain MMServer.ini. This file contains server settings, and the list of databases that you wish to serve:

Sample MMServer.INI file:

```
[Settings]
; This is the TCP/IP port that the server will operate on
Port=8080
; The following 4 properties require MusicMaster 5.0 SR-11 or later
; Used to control the encoding of the XML reply messages (Default = UTF-8)
Encoding=UTF-8
; Indicates if XML header is included (Default=1-yes, 0 to disable)
ByteOrder=1
; XML headers on reply messages (Default = 1)
XMLHeader=1
; Used to control the XML indenting on reply messages (Default = 1)
XMLIndent=1

; Create an Interface section for each database to be serviced. Each section must
; be named with the word 'Interface' and the next number in sequence. While this
; allows you to define a large number of databases, computer resources and Nexus
; utilization may not support more than a few databases per server computer.
[Interface1]
; ID is a short unique text string to identify each database. This is
; the data that is used in the station attribute when sending requests to the server.
ID=Oldies
; System indicates the automation system interface to use. Generic
; provides no special functionality.
; Logs are exported exactly as they would be from within MusicMaster.
; Other systems will be supported in the future.
System=Generic
; This is the database path and filename as seen from where the server is
; installed. It is highly recommended that you install the server on the same
; physical computer as the database files.
Data=D:\MMWin\Oldies.mmd
; If you wish to have logs exported, indicate the name of the export design to use
; This name must match exactly how the name appears in MusicMaster
LogDesign=MusicMaster Online
; If you wish to enable access to a station in a read only manner, add the
; ReadOnly property and set it to one. This will restrict access to any functions
; that can modify the database. [Note: Requires MusicMaster 4.0 SR-19 or later]
ReadOnly=1
```

Once everything is set up, MMServer may be launched. Once it is running, click Start Service to activate the HTTP server. Once this is done, HTTP requests will be received and processed. MMServer can be minimized to the system tray where a MusicMaster icon will appear.

If you wish to enable the Publish Schedule button in the MusicMaster Schedule Editor, each serviced database must have an INI file that matches the name of the database itself, in the same folder as the database. (Example: Sample.mmd requires Sample.ini) Only a few settings are used in here: [Note: This is only needed if you wish to enable the Schedule Editor publish schedule button]

```
[Server]
; This is the address of the server, name or IP address and port. (Default = 80)
Address=localhost:8080
; This needs to match the ID string in MMServer.INI for the matching database.
DataID=Sample
```

Using the API:

Request messages are sent to the server in an HTTP POST message with the XML message in the body of the POST message. Request messages are always XML messages with a root tag of mmRequest. The desired command is specified in the command attribute of the root tag. This tag will also require a station attribute for most messages. You can also optionally specify a version attribute for backward-compatibility. When not specified, the version will always default to the latest version of the interface, so it is wise to specify a version="1" attribute. You should also add a client attribute to your request message with a string that defines the product and version that is making the request. A userData attribute is also available. Whatever data is passed here will be returned in the reply message unaltered.

Server responses are always XML messages with a root tag of mmReply. The command name attribute will be repeated in the reply, and the version attribute will always be included to indicate the version of the command was used. If userData was included in the request, it will be included in the reply as specified in the request. A status attribute will always be included in reply messages. If this returns as 'ok' then the command completed normally and the appropriate contents will be found within this tag. If this returns 'error' then only error message information will be returned. See the section on error messages for more information.

As of MusicMaster 5.0 SR-11, all reply messages are encoded as UTF-8 with XML header, by default. Request messages should also be encoded as UTF-8, especially if they contain any non-English characters.

Using the ActiveX DLL:

Beginning with MusicMaster 4.0 SR-17, Nexus also features an alternate ActiveX DLL interface.

When installed, the NexusAPI.DLL will be registered in the system and can be called by any application or programming language that can interface with a COM compatible component. This component must be installed on a PC where a copy of MusicMaster is already installed. This computer does not necessarily have to house the MusicMaster data. To use this API, create an instance of the NexusAPI.NexusStation object.

All API commands are constructed by the calling application as XML messages. These messages are sent to the MMRequestXML function. The only parameter is the Message itself. The reply is another XML message with the results of the command message.

The Connect and Disconnect functions are used to connect to and disconnect from a database. You must establish a database connection before you can use the MMRequestXML function. The Connect function has only one parameter, which is the path and filename of the MusicMaster database you wish to connect to. This function returns a Boolean value that indicates the connection status. The Disconnect function takes no parameters, and also returns a Boolean result value.

The only other functions are the GetProperty and SetProperty functions. These are used to alter the behavior of the component.

NOTE: The database property of the Connection function is not currently used. To specify the database, you must first call the SetProperty function and set the “DATA” property to the full path and filename of the database before you call Connect or MMRequestXML.

When using the DLL, the ping, getStations, and getServerInfo commands are not available.

The following functions are available within the ActiveX DLL interface:

Boolean Connect(String DatabaseName)

Used to connect to the specified database. You must supply the complete path/filename specification.

Boolean Disconnect()

Used to disconnect from the active database.

String GetProperty(String PropName, Optional String DefaultValue)

Used to check the value of a component property.

Void SetProperty(String PropName, String Value)

Used to set the value of a component property.

The following properties are currently available:

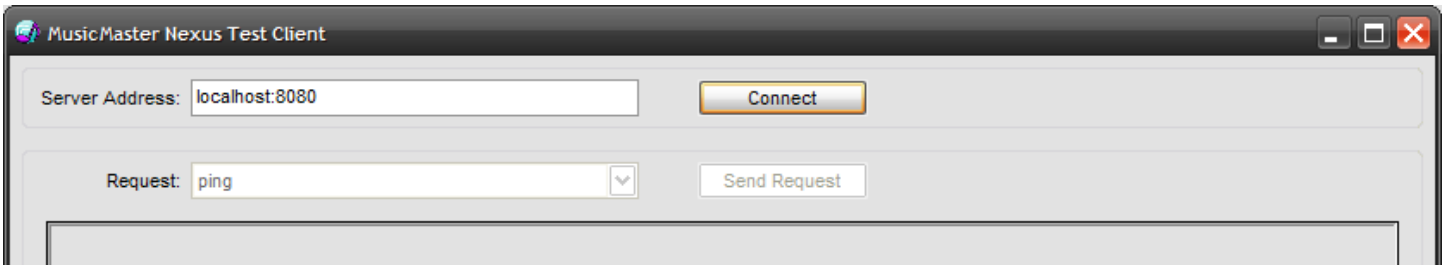
Property Name	Description
Data	Full path and filename of the Access database you wish to work with. Must set this before calling Connect or MMRequestXML.
PrettyXML	Determines whether the response messages are formatted with line breaks and indentation. 0 = No (Default) 1 = Yes

String MMRequestXML(String Request)

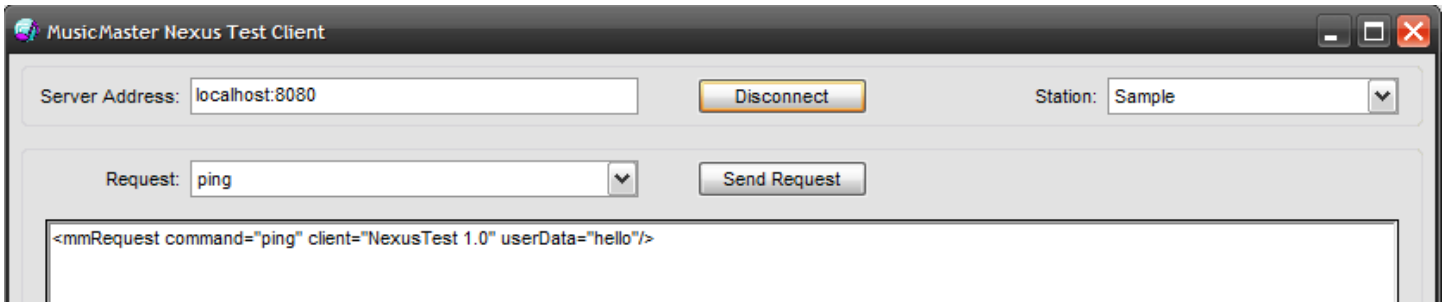
Used to send commands to the API using XML messages. XML response messages are returned. For details, see section II of this document.

Working with the API:

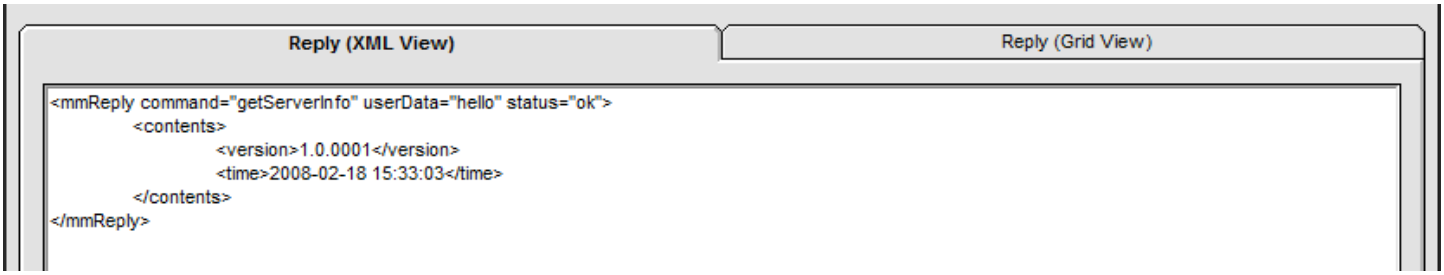
You can experiment with the API using the MMServerTest.exe executable. When launched, you will be prompted to enter the server address of an active MusicMster Nexus Server. Once entered, click the Connect button.

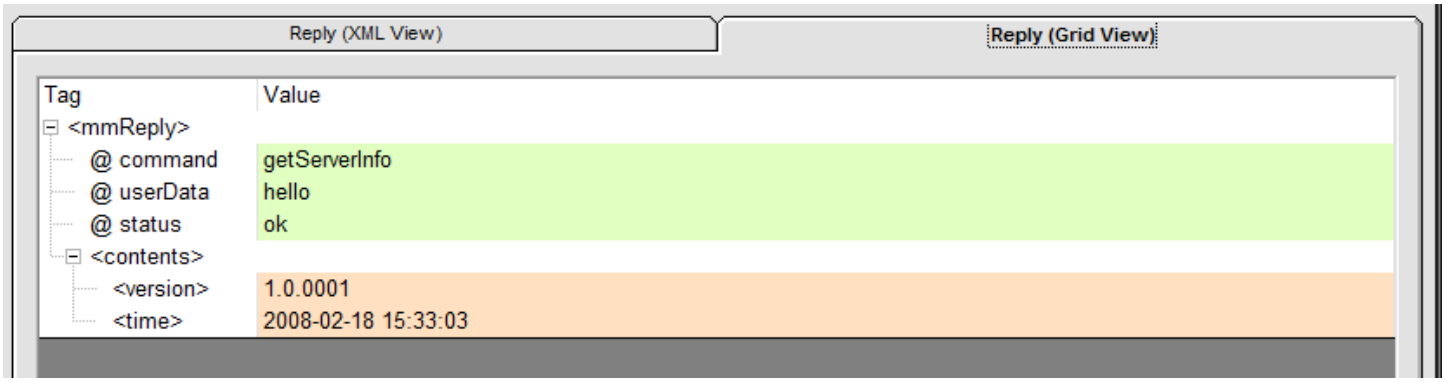


If a successful connection is made, the Connect button will change to Disconnect, and a new selection box will appear to the right where you can select the station you wish to work with:



Now you can select from the available server request messages, and a sample of the message will appear in the large request text box. You can manually edit this request as you wish, and then click “Send Request” to send that request message to the server. The server reply message will appear in the lower reply box. There are two tabs available, one to show the raw XML reply message, and another to format the replay in a grid:





Section II: API Function Reference

This section documents the available command messages that may be sent to the Nexus Server. A description and example of each message is provided, along with a description and example of each command response message.

Message Syntax Notes:

Please note that optional attributes are shown in [brackets]. These brackets are not included in an actual XML message.

API Versioning:

Starting with MusicMaster 5.0 SR-3, the Nexus API will contain a revision ID that you can use to check which revision of the API is available.

ping

Description:
Verifies communications with the Nexus Server.

Sample Request:
```
<mmRequest command="ping" [version="1"] [client=""] [userData=""] />
```

Request Notes:
This function does not reference a specific database on the server, so no station attribute is needed.

Sample Reply:
```
<mmReply command="ping" version="1" [userData=""] status="ok" />
```

Reply Notes:
none

Compatibility and Version Info:
Available in all versions

getServerInfo

Description:
Returns information about the MusicMaster Nexus Server.

Sample Request:
```
<mmRequest command="getServerInfo" [version="1"] [client=""] [userData=""] />
```

Request Notes:
This function does not reference a specific database on the server, so no station attribute is needed.

Sample Reply:
```
<mmReply command="getServerInfo" status="ok">
   <contents>
      <version>5.0.0050</version>
      <time>2012-06-01 10:00:00</time>
      <timeZone>Central Daylight Time</timeZone>
      <timeOffset>GMT-0500</timeOffset>
   </contents>
</mmReply>
```

Reply Notes:
version – returns the MusicMaster Nexus version
time – returns the current date/time on the server in ISO format
timeZone – name of the time zone on the server computer
timeOffset – time offset from GMT on the server computer

Compatibility and Version Info:
Available in all versions

getStations

Description:
Returns the list of stations served by the server you are connected to.

Sample Request:
```
<mmRequest command="getStations" [version="1"] [client=""] [userData=""] />
```

Request Notes:
This function does not reference a specific database on the server, so no station attribute is needed.

Sample Reply:
```
<mmReply command="getStations" version="1" [userData=""] status="ok">
   <contents>
      <stations>
         <station>
            <id>Oldies</id>
            <description>MusicMaster Oldies</description>
            <interface>generic</interface>
            <status>0</status>
         </station>
         <station>
            <id>Sample</id>
            <description>Sample Database</description>
            <interface>generic</interface>
            <status>0</status>
         </station>
      </stations>
   </contents>
</mmReply>
```

Reply Notes:
Each station server by the server is returned in a station tag. These are all returned in a stations tag.
id – returns the unique ID for each station
description – the MusicMaster database logo
interface – this value is either 'generic' of the name of the custom interface in use
status – returns a numeric value indicating the status of the station database connection
    0=ok
    1=busy (temporary condition while the database is being opened and inspected)
    -1=error (unable to locate or open the specified database)
Note: You will use the value of the id tag when calling other database-specific functions.

Compatibility and Version Info:
Available in all versions
Documentation of the station status tag updated 10-28-2010

getStationProperty

Description:
Returns one or more property settings for a station.

Sample Request:
```
<mmRequest command="getStationProperty" station="ID" [version="1"] [client=""] [userData=""]  />

or

<mmRequest command="getStationProperty" station="ID" [version="1"] [client=""] [userData=""]>
   <contents>
      <properties>
         <property name="propname1" />
         <property name="propname2" />
         <property name="propname3" />
      </properties>
   </contents>
</mmRequest>
```

Request Notes:
If no properties are specified, as in the first example, all available properties will be returned.

Sample Reply:
```
<mmReply command="getStationProperty" version="1" [userData=""] status="ok">
   <contents>
      <properties>
         <property name="propname1">value</property>
         <property name="propname2">value</property>
         <property name="propname3">value</property>
      </properties>
   </contents>
</mmReply>
```

Reply Notes:
Each property is returned in a property tag. These are all returned in a properties tag.
name attribute – name of the property
the property tag value is the value of the property

See Appendix A – Station Properties for the defined properties that may be returned here

Compatibility and Version Info:
Available in all versions

setStationProperty

Description:
Allows you to set one or more property settings for a station.

Sample Request:
```
<mmRequest command="setStationProperty" station="ID" [version="1"] [client=""] [userData=""]>
   <contents>
      <properties>
         <property name="propname1">value</property>
         <property name="propname2">value</property>
         <property name="propname3">value</property>
      </properties>
   </contents>
</mmRequest>
```

Request Notes:
Each property is specified in a property tag. These are all included in a single properties tag.
name attribute – name of the property
the property tag value is the value of the property

Sample Reply:
```
<mmReply command="setStationProperty" version="1" [userData=""] status="ok" />
```

Reply Notes:
See Appendix A – Station Properties for the defined properties that may be adjusted here

Compatibility and Version Info:
Available in all versions

getStationInfo

Description:
Returns basic identification data for the active database.

Sample Request:
```
<mmRequest command="getStationInfo" station="ID" [version="1"] [client=""] [userData=""] />
```

Request Notes:
none

Sample Reply:
```
<mmReply command="getStationInfo" station="ID" version="1" [userData=""] status="ok">
   <contents>
      <logo>MusicMaster Oldies</logo>
      <guid>B0D0CE53-3EE3-4274-B54A9C903DDAD883</guid>
      <lastScheduled>2006-10-31T23:00:00</lastScheduled>
   </contents>
</mmReply>
```

Reply Notes:
logo - returns the database logo as set by the user.
guid - returns the database GUID which is designed to be a unique identifier for each database. Note that the GUID may not be unique if the database was copied outside of MusicMaster.
lastScheduled - returns the date and time of the last hour scheduled in ISO format

Compatibility and Version Info:
Available in all versions

getAPIVersion

Description:
Returns a value indicating the revision level of the Nexus API.

Sample Request:
```
<mmRequest command="getAPIVersion" />
```

Request Notes:
none

Sample Reply:
```
<mmReply command="getAPIVersion" status="ok">
   <contents>
      <version>5001</version>
   </contents>
</mmReply>
```

Reply Notes:
If the reply does not contain status=”ok” then the Nexus API is from before this function became available and you will need to check the version of MusicMaster to know what API functions are available.

Version – an integer numeric ID indicating the revision level of the Nexus API. This value will be increased on future releases. These numbers may not be sequential. See Section III for the API revision history.

Compatibility and Version Info:
Available with MusicMaster 5.0 SR-3 and later [5001]

getFieldList

Description:
Returns a list of database fields in the active database. This list includes all of the user fields, and the built-in fields that could potentially be modified by an external application.

Sample Request:
```
<mmRequest command="getFieldList" station="ID" [version="1"] [client=""] [userData=""] />
```

Request Notes:
none

Sample Reply:
```
<mmReply command="getFieldList" station="ID" version="1" [userData=""] status="ok">
   <contents>
      <fields>
         <field id="3" type="99" len="0" name="Category" abbr="CAT" locked="0"/>
         <field id="11" type="9" len="1" name="Mark Status" abbr="MK" locked="0"/>
         <field id="12" type="13" len="5" name="Run Time" abbr="RT" locked="0"/>
         <field id="21" type="99" len="0" name="Dayparting" abbr="DP" locked="0"/>
         <field id="22" type="99" len="0" name="Packet" abbr="PK" locked="0"/>
         <field id="46" type="2" len="3" name="Rotation Weight" abbr="WT" locked="0"/>
         <field id="47" type="2" len="2" name="Max Per Day" abbr="DayMax" locked="0"/>
         <field id="51" type="11" len="0" name="Start Day/Hour" abbr="Starts" locked="0"/>
         <field id="52" type="11" len="0" name="End Day/Hour" abbr="Ends" locked="0"/>
         <field id="53" type="2" len="4" name="Kill Plays" abbr="KillPly" locked="0"/>
         <field id="102" type="1" len="255" name="Artist Name" abbr="ART" locked="0"/>
         <field id="103" type="1" len="255" name="Song Title" abbr="TTL" locked="0"/>
         <field id="104" type="22" len="64" name="Artist Keywords" abbr="ARTS" locked="0"/>
         <field id="105" type="22" len="64" name="Title Keywords" abbr="TTLS" locked="0"/>
         <field id="106" type="22" len="64" name="CD/LP Titles" abbr="LP" locked="0"/>
         <field id="107" type="1" len="16" name="Source Type" abbr="SRC" locked="0"/>
         <field id="108" type="1" len="64" name="Audio Notes" abbr="Audio" locked="0"/>
      </fields>
   </contents>
</mmReply>
```

Reply Notes:
The list of fields is returned in the **contents/fields** section. Each field has a **field** tag.
The following attributes are available for each field:

Id	Internal field ID number
type	MusicMaster field type code: 1=Text 2=Numeric-Fixed 3=Numeric-Float 4=Currency 9=Boolean 11=Date 12=Time 13=Length 21=Keyword-Single 22=Keyword-Multiple 31=Attribute-Single 32=Attribute-In/Out 33=Attribute-Combo 34=Attribute-Multiple 72=Memo 99=Pointer
len	Maximum field entry length where applicable
name	User full field name
abbr	User field abbreviated name

locked	1=field locked from changes 0=field is not locked
Priority	1=primary field 2=secondary field 0=other

Compatibility and Version Info:
Available in all versions

getAttributeList

Description:
Returns a list of all available defined attribute codes in the active database.

Sample Request:
```
<mmRequest command="getAttributeList" station="ID" [version="1"] [client=""] [userData=""] />
```

Request Notes:
none

Sample Reply:
```
<mmReply command="getAttributeList" station="ID" version="1" [userData=""] status="ok">
   <contents>
      <attributes>
         <attribute field="110" code="66" name="British" value="0"/>
         <attribute field="110" code="67" name="Country" value="0"/>
         <attribute field="110" code="73" name="Instrumental" value="0"/>
         <attribute field="110" code="77" name="Motown" value="0"/>
         <attribute field="110" code="83" name="Surfer" value="0"/>
         <attribute field="111" code="70" name="FEMALE" value="0"/>
         <attribute field="111" code="77" name="MALE" value="0"/>
         <attribute field="112" code="49" name="Slow" value="1"/>
         <attribute field="112" code="51" name="Medium Slow" value="3"/>
         <attribute field="112" code="53" name="Medium" value="5"/>
         <attribute field="112" code="55" name="Medium Fast" value="7"/>
         <attribute field="112" code="57" name="Fast" value="9"/>
      </attributes>
   </contents>
</mmReply>
```

Reply Notes:
The list of attribute codes is returned in the **contents/attributes** section. Each attribute code has an **attribute** tag. The following attributes are available for each defined attribute code:

Field	Internal MusicMaster field ID code
Code	Attribute entry code (ASCII character code)
Name	User attribute descriptive name
Value	User attribute value

Compatibility and Version Info:
Available in all versions

getCategoryList

Description:
Returns a list of all available defined library categories in the active database.

Sample Request:
```
<mmRequest command="getCategoryList" station="ID" [version="1"] [client=""] [userData=""] />
```

Request Notes:
none

Sample Reply:
```
<mmReply command="getCategoryList" station="ID" version="1" [userData=""] status="ok">
   <contents>
      <categories>
         <category id="1" code="H" nonmx="0" name="Heart" backcolor="255" forecolor="16777215"/>
         <category id="2" code="S" nonmx="0" name="Soul" backcolor="16711782"
forecolor="16777215"/>
         <category id="3" code="N" nonmx="0" name="Novelty" backcolor="65535" forecolor="1"/>
         <category id="4" code="F" nonmx="0" name="Fringe" backcolor="65280" forecolor="1"/>
         <category id="5" code="E" nonmx="0" name="Beatles" backcolor="52479" forecolor="1"/>
         <category id="6" code="K" nonmx="0" name="Kickoffs" backcolor="13434624"
forecolor="1118481"/>
         <category id="7" code="J" nonmx="1" name="Jingles" backcolor="11184810" forecolor="1"/>
         <category id="8" code="P" nonmx="1" name="Promos" backcolor="3355443"
forecolor="16777215"/>
      </categories>
   </contents>
</mmReply>
```

Reply Notes:
The list of categories is returned in the **contents/categories** section. Each category has a **category** tag. The following attributes are available for each defined category:

id	Internal MusicMaster category ID code
code	User category code (3 chars max)
name	User category descriptive name
nonmx	Identifies the category type: 0=Music category 1=Non-Music category
backcolor	Background display color
forecolor	Foreground display color

Compatibility and Version Info:
Available in all versions

getCategoryGroups

Description:
Returns a list of all available defined library category groups in the active database.

Sample Request:
```
<mmRequest command="getCategoryGroups" station="ID" [version="1"] [client=""] [userData=""] />
```

Request Notes:
none

Sample Reply:
```
<mmReply command="getCategoryGroups" station="ID" version="1" [userData=""] status="ok">
   <contents>
      <categoryGroups>
         <categoryGroup name=”Entire Library”>
            <category>1</category>
            <category>2</category>
            <category>3</category>
         </categoryGroup>
      </categoryGroups>
   </contents>
</mmReply>
```

Reply Notes:
The list of category groups is returned in the **contents/categoryGroups** section. Each category group includes a list of the categories contained in the group. These are category ID codes that can be looked up using the getCategories function.

Compatibility and Version Info:
Available in MusicMaster 4.0 SR-20 and later

getSongLists

Description:
Returns a list of all available user-created saved song lists

Sample Request:
```
<mmRequest command="getSongLists" station="ID" [version="1"] [client=""] [userData=""] />
```

Request Notes:
none

Sample Reply:
```
<mmReply command="getSongLists" station="ID" version="1" [userData=""] status="ok">
   <contents>
      <songLists>
         <songList id=”1” name=”Love Songs”/>
         <songList id=”3” name=”Research List”/>
      </songLists>
   </contents>
</mmReply>
```

Reply Notes:
The list of saved song lists is returned in the contents/songLists section. Each song list has a **songlist** tag. The following attributes are available for each song list:

Id	Internal MusicMaster Song List ID code
Name	Song List name

Compatibility and Version Info:
Available in all versions

getSongsByList

Description:
Returns a list of songs from a saved song list.

Sample Request:
```
<mmRequest command="getSongsByList" station="ID" [version="1"] [client=""] [userData=""]>
   <contents>
      <songList id=”1”/>
      [<fields>
         <field id="101" />
         <field name="Title" />
      </fields>]
   </contents>
</mmRequest>
```

Request Notes:
The song list you wish to load is specified in the contents/songList tag. The id attribute is the MusicMaster ID number of the desired list. You can obtain a list of available saved song lists with the getSongLists message.

The fields section must contain one field tag for each song field you wish to include in the reply message. You can use the id attribute with the internal field ID codes, or a name attribute with the user-specified field name. Only the specified fields will be returned for each song. However, the internal song ID number will always be included in the reply. You can obtain a list of available fields with the getFields message.

Note that the fields section is optional. If not specified, only the song id, runtime and the primary and secondary fields will be returned. Typically, primary and secondary fields are defined as title and artist.

Sample Reply:
```
<mmReply command="getSongsByList" station="ID" version="1" [userData=""] status="ok">
   <contents>
      <songList>
         <song songId="1">
            <field id="100" name="Description">Beatles / Yesterday</field>
            <field id="101" name="Artist">Beatles</field>
         </song>
      </songList>
   </contents>
</mmReply>
```

Reply Notes:
The list of songs is returned in the contents/songlist section. Each song has a song tag. The songid attribute contains the MusicMaster internal song ID for each song. One of more field tags may be included with the value of each requested field. Each field tag includes an id attribute with the internal field ID number and a name attribute with the user defined field name.

Compatibility and Version Info:
Available in all versions

getSongsByQuery

Description:
Returns a list of matching songs from the library based on the specified search criteria.

Sample Request:
```
<mmRequest command="getSongsByQuery" station="ID" [version="1"] [client=""] [userData=""]>
   <contents>
      <query>
         [<categories>
            <category [id=”1”] [code="A"]/>
         </categories>]
         [<filters>
            <filter [type=”fieldId|fieldName”] target="Title" operator="contains" value="love"/>
         </filters>]
         [<sortKeys>
            <sortKey [type=”fieldId|fieldName”] target="Artist" order="asc|desc"/>
         </sortKeys>]
         [<properties>
            <property name=”propname”>value</property>
         </properties>]
      </query>
      [<fields>
         <field id="101" />
         <field name="Title" />
      </fields>]
   </contents>
</mmRequest>
```

Request Notes:
query - The actual search request is provided in the contents/query section. All sections and tags are optional. You only need to include the portions necessary to define the desired search query. See appendix B for complete details of the Nexus query tags.

fields - The fields section must contain one field tag for each song field you wish to include in the reply message. You can use the id attribute with the internal field ID codes, or a name attribute with the user-specified field name. Only the specified fields will be returned for each song. However, the internal song ID number will always be included in the reply. You can obtain a list of available fields with the getFields message. See appendix C for addition options that can be used with your fields specification.

Note that the fields section is optional. If not specified, only the song id, runtime and the primary and secondary fields will be returned. Typically, primary and secondary fields are defined as title and artist.

Sample Reply:
```
<mmReply command="getSongsByQuery" station="ID" version="1" [userData=""] status="ok">
   <contents>
      <songList>
         <song songId="1">
            <field id="100" name="Description">Beatles / Yesterday</field>
            <field id="101" name="Artist">Beatles</field>
         </song>
      </songList>
   </contents>
</mmReply>
```

Reply Notes:
The list of songs is returned in the contents/songlist section. Each song has a song tag. The songid attribute contains the MusicMaster internal song ID for each song. One or more field tags may be included with the value of each requested field. The field tags include an id attribute with the internal field ID number and a name attribute with the user defined field name.

Compatibility and Version Info:
Available in all versions

updateSongs

Description:
Updates fields on one or more existing songs.

Sample Request:
```
<mmRequest command="updateSongs" station="ID" [version="1"] [client=""] [userData=""]>
   <contents>
      <songList>
         <song [songId="1"] [cutId=”x”] />
            <field id="108">NewValue</field>
         </song>
      </songList>
   </contents>
</mmRequest>
```

Request Notes:
The actual update request is provided in the contents/songList section.

Include one song tag for each song you wish to update. The songId attribute must specify the MusicMaster id number of the song to update. Alternatively, you may use the cutId attribute to specify the song. For each song, specify one or more FIELD tags. You should only include field tags for the fields you wish to update. Any existing values will be replaced with the data provided. The field tag requires an id attribute with internal field ID number, or a name attribute with the user-defined field name. You can obtain a list of available fields with the getFields message. The value of the tag should be the data you want to write to the database.

Sample Reply:
```
<mmReply command="updateSongs" station="ID" version="1" [userData=""] status="ok" />
```

Reply Notes:

Compatibility and Version Info:
Available in all versions
Added cutId support with API revision 5005

deleteSongs

Description:
Deletes one or more existing songs.

Sample Request:
```
<mmRequest command="deleteSongs" station="ID" [version="1"] [client=""] [userData=""]>
   <contents>
      <songList>
         <song [songId="1"] [cutId=”x”] />
         </song>
      </songList>
   </contents>
</mmRequest>
```

Request Notes:
The actual delete request is provided in the contents/songList section.

Include one song tag for each song you wish to update. The songId attribute must specify the MusicMaster id number of the song to update. Alternatively, you may use the cutId attribute to specify the song. These deletes cannot be undone.

Sample Reply:
```
<mmReply command="deleteSongs" station="ID" version="1" [userData=""] status="ok" />
```

Reply Notes:

Compatibility and Version Info:
Available in MusicMaster 4.0 SR-17 and later
Added cutId support with API revision 5005

importSongs

Description:
Adds new songs to the database.

Sample Request:
```
<mmRequest command="importSongs" station="ID" [version="1"] [client=""] [userData=""]>
   <contents>
      <songList>
         <song>
            <field id="108">NewValue</field>
         </song>
      </songList>
      [<fields>
         <field id="101" />
         <field name="Title" />
      </fields>]
   </contents>
</mmRequest>
```

Request Notes:
The actual update request is provided in the contents/songList section.

song – You must add a song tag for each song you wish to add. The song tag includes the information you wish to import for this new song in one or more field tags. You should only include field tags for the fields you wish to update on the new songs. Any default values will be replaced with the data provided. The field tag requires an id attribute with internal field ID number, or a name attribute with the user-defined field name. You can obtain a list of available fields with the getFields message. The value of the tag should be the data you want to write to the database.

fields - The fields section must contain one field tag for each song field you wish to include in the reply message. You can use the id attribute with the internal field ID codes, or a name attribute with the user-specified field name. Only the specified fields will be returned for each song. However, the internal song ID number will always be included in the reply. You can obtain a list of available fields with the getFields message.

Note that the fields section is optional. If not specified, only the song id, runtime and the primary and secondary fields will be returned. Typically, primary and secondary fields are defined as title and artist.

Sample Reply:
```
<mmReply command="importSongs" station="ID" version="1" [userData=""] status="ok">
   <contents>
      <songList>
         <song songId="1">
            <field id="100" name="Description">Beatles / Yesterday</field>
            <field id="101" name="Artist">Beatles</field>
         </song>
      </songList>
   </contents>
</mmReply>
```

Reply Notes:
The list of songs imported is returned in the contents/songlist section. Each song has a song tag. The songid attribute contains the MusicMaster internal song ID for each song. One or more field tags may be included with the value of each requested field. The field tags include an id attribute with the internal field ID number and a name attribute with the user defined field name.

Compatibility and Version Info:
Available in all versions

getSongInfo

Description:
Returns information about a specific list of songs.

Sample Request:
```
<mmRequest command="getSongInfo" station="ID" [version="1"] [client=""] [userData=""]>
   <contents>
      <songList>
         <song [songId="1"] [cutId=”x”] />
      </songList>
      [<fields>
         <field id="101" />
         <field name="Title" />
      </fields>]
   </contents>
</mmRequest>
```

Request Notes:
The list of songs to include must be specifically included in the songList tag with one song tag for each song you want to include.

The fields section must contain one field tag for each song field you wish to include in the reply message. You can use the id attribute with the internal field ID codes, or a name attribute with the user-specified field name. Only the specified fields will be returned for each song. You can obtain a list of available fields with the getFields message. See appendix C for addition options that can be used with your fields specification.

Note that the fields section is optional. If not specified, this message will return all available fields.

Sample Reply:
```
<mmReply command="getSongInfo" station="ID" version="1" [userData=""] status="ok">
   <contents>
      <songList>
         <song songId="1">
            <field id="100" name="Description">Beatles / Yesterday</field>
            <field id="101" name="Artist">Beatles</field>
         </song>
      </songList>
   </contents>
</mmReply>
```

Reply Notes:
The list of songs is returned in the contents/songlist section. Each song has a song tag. The songId attribute contains the MusicMaster internal song ID for each song. One or more field tags may be included with the value of each requested field. The field tags include an id attribute with the internal field ID number and a name attribute with the user defined field name.

Compatibility and Version Info:
Available in all versions

getSchedule

Description:
getSchedule is called by a server client to retrieve schedule information for a specified time period. The schedule data is returned in the reply message. A minimal amount of customization is available.

Sample Request:
```
<mmRequest command="getSchedule" station="ID" [version="1"] [client=""] [userData=""]>
   <contents>
      <startTime>yyyy-mm-ddThh:00:00</startTime>
      <endTime> yyyy-mm-ddThh:00:00</endTime>
      [<properties>
         <property name=”propName”>value</property>
      </properties>]
      [<fields>
         <field id="101" />
         <field name="Title" />
      </fields>]
   </contents>
</mmRequest>
```

Request Notes:
startTime – The first hour to include in the reply. This is the first full hour that will be included.

endTime – The last hour to include in the reply. This is the last full hour that will be included.

Note that currently requests for more than 24 hours of schedule data will not be accepted. Also note that it is not currently possible to request a portion of an hour.

properties – Include one property tag for each query property you wish to specify. The following properties are currently available:
description = Include element description in element tags (default=True)
estimate = Include the estimated length of unscheduled positions when calculating airtime (default=False)
songList = Include detailed song section after elements (default=True)
mergeTraffic = Include merged spots in the returned schedule, if available (default=False)
lockHours = Lock Format Clocks and save these hours in history, if needed (default=False)
elementFilter = List of element types to include (default=Include All)
note: This is a string of letters indicating the elements that you want included. Leave out any codes for the elements to ignore:
S = Stopset (This is actually any Lognote type element with the Sweep marker flag set)
L = LogNote
* = Timing Checkpoint
@ = Format List
M = Scheduled Music
U = Unscheduled Music

fields - The fields section must contain one field tag for each song field you wish to include in the reply message. You can use the id attribute with the internal field ID codes, or a name attribute with the user-specified field name. Only the specified fields will be returned for each song. However, the internal song ID number will always be included in the reply. You can obtain a list of available fields with the getFields message. See appendix C for addition options that can be used with your fields specification.

Note that the fields section is optional. If not specified, only the song id, runtime and the primary and secondary fields will be returned. Typically, primary and secondary fields are defined as title and artist.

Sample Reply:
```
<mmReply command="getSchedule" station="ID" version="1" [userData=""] status="ok">
   <contents>
```

```
      <schedule>
         <element airDate="" airTime="" runTime="" class="" type="" [sweep="1"] [fill="1"]
[songId="1"] properties="ER" historyId="">Description</element>
      </schedule>
      <songList>
         <song songId="1">
            <field id="100" name="Description">Beatles / Yesterday</field>
            <field id="101" name="Artist">Beatles</field>
         </song>
      </songList>
   </contents>
</mmreply>
```

The schedule data is returned in the schedule section. Each scheduled element has an element tag. The following attributes are included for each element:

Airdate	Scheduled air date (yyyy-mm-dd)
Airtime	Scheduled air time (hh:mm:ss)
Runtime	Scheduled run time (mm:ss)
Class	Element class code: H = Hour Marker L = Lognote S = Stopset T = Traffic (spot) * = Timing Element M = Scheduled Song U = Unscheduled Song
Type	Element type code: 0 = Hour Marker 101 = Song Element 201 = Lognote/Stopset 231 = Traffic Merge 232 = Commercial 301 = Timing Element
Sweep	1 = element is a sweep marker
Fill	1 = element is a fill song
Transition	Transition code (text – user-defined)
Runtime	Scheduled run time (mm:ss)
Properties	String of letter codes. L=Locked, E=No Export, R=No Reconcile, P=No Print, M=Manually Scheduled.
historyId	Unique ID for each history element

The text of the element tag is the element description.

Additional field information about the songs scheduled in the time period is returned in the songlist section. Each unique scheduled song has a song tag. The following attributes are included for each song:

songId	Internal MusicMaster Song ID number
addDate	Date and time this song was added to the library
playCount	Number of times this song played in the returned schedule time period.

One or more field tags may be included with the value of each requested field. The field tags include an id attribute with the internal field ID number and a name attribute with the user field descriptive name.

Compatibility and Version Info:

Available in all versions
Attribute: transition added with MusicMaster 4.0 SR-12
Property: estimate added with MusicMaster 4.0 SR-23
Attribute: properties added with MusicMaster 5.0 SR-8 (API version 5003)

publishMetadata

Description:
Returns a list of songs that have changed (according to the song edit date) since a specified time or since the last time this function was called.

Sample Request:
```
<mmRequest command="publishMetadata" station="ID" [version="1"] [client=""] [userData=""]>
   <contents>
      [<properties>
            <property name=”lastedit”>2015-01-01T00:00:00</property>
      </properties>]
      [<fields>
         <field id="12" />
         <field name="Artist" />
         <field name="Title" />
      </fields>]
   </contents>
</mmRequest>
```

Request Notes:
This function basically performs the same operation as if you used getSongsByQuery to search for songs where the last edit date (field 27) is later than a specific time. However, this function records the last time it was called and will automatically return the list of songs changed since that time if no date/time is specified.

The entire contents section is optional. If you wish to know which songs have changed since a specific time, specify that in the “lastedit” property. If you wish to include additional song fields, specify those in the options “field” section.

Sample Reply:
```
<mmReply command="publishMetadata" station="ID" version="1" [userData=""] status="ok">
   <contents>
      <songList recordCount=”2”>
         <song songId="1" />
         <song songId="2" />
      </songList>
</mmReply>
```

Reply Notes:
By default, this function will only return the song ID of the songs that have changed. You can include the options Fields section in your request if you wish to return additional data.

Compatibility and Version Info:
Available in MusicMaster 5.0 SR-15 and later

This function may have a vendor specific override available and when that is the case, this function may be automated to push changes to your automation system by adding a command to the Interface section of MMServer.ini:
PublishMetaData=[interval]
Where [interval] is specified as the number of seconds (S,n), minutes (M,n) or hours (H,n) between calls to this function. Example: PublishMetaData=S,15 (will call this function every 15 seconds)

publishSchedule

Description:
Causes the server to export a schedule. The actual schedule data is not returned in the reply message.

Sample Request:
```
<mmRequest command="publishSchedule" station="ID" [version="1"] [client=""] [userData=""]>
   <contents>
      <startTime>yyyy-mm-ddThh:00:00</startTime>
      <endTime> yyyy-mm-ddThh:00:00</endTime>
      [<design>design name</design>]
      [<fileSpec>full path and file</fileSpec>]
   </contents>
</mmRequest>
```

Request Notes:
startTime – The first hour to include in the reply. This is the first full hour that will be included.

endTime – The last hour to include in the reply. This is the last full hour that will be included.

Note that currently requests for more than 24 hours of schedule data will not be accepted. Also note that it is not currently possible to request a portion of an hour.

design – specifies the name of the MusicMaster export design to use. By default, this is defined in the server INI and should not need to be included here. Optional

filename – can be used to override the path/filename in the export design. Optional

Sample Reply:
```
<mmReply command="publishSchedule" station="ID" version="1" [userData=""] status="???????">
   <errors>
      <error>message</error>
   </errors>
</mmReply>
```

Reply Notes:
This function does not return any schedule data. Instead, the server exports the log as it would be exported from within MusicMaster. If a non-generic interface is in use, the method used to export the schedule may vary.

Compatibility and Version Info:
Available in all versions

This function may have a vendor specific override available and when that is the case, this function may be triggered by the Publish Schedule button in the MusicMaster Schedule Editor.

reconcileSchedule

Description:
This is used to start reconciliation in the server. This would generally be started by a timer in the server. See section III for the server automatic function reference. A Server client could also call this as soon as it makes a log available to make the process happen as quickly as possible. Note that this is only a full log reconcile function from a disk file like what exists within MusicMaster.

Sample Request:
```
<mmRequest command="reconcileSchedule" station="ID" [version="1"] [client=""] [userData=""]>
   <contents>
      <startTime>yyyy-mm-ddThh:00:00</startTime>
      <endTime> yyyy-mm-ddThh:00:00</endTime>
      [<definition>def file name</definition>]
   </contents>
</mmRequest>
```

Request Notes:
startTime – The first hour to include in the reply. This is the first full hour that will be included.

endTime – The last hour to include in the reply. This is the last full hour that will be included.

Note that currently requests for more than 24 hours of schedule data will not be accepted. Also note that it is not currently possible to reconcile a portion of an hour.

definition – specifies the name of the MusicMaster reconcile definition to use. Optional

Sample Reply:
```
<mmReply command="reconcileSchedule" station="ID" version="1" [userData=""] status="???????">
   [<errors>
      <error>message</error>
   </errors>]
   [<log>
      log text with any messages that were not considered to be errors
   </log>]
</mmReply>
```

Reply Notes:
This function does not return any schedule data. Instead, the server performs the reconciliation and returns a status code and any error messages. If a non-generic interface is in use, the method used to reconcile the schedule may vary.

Compatibility and Version Info:
Available in all versions

reconcileElements

Description:
Performs individual history updates in MusicMaster. You can use this message to indicate that one or more elements have changed. This would generally be used to report a dropped or inserted song, or a more complex move or swap operation. You can also use this message to report the actual air time that an element started or ended playing.

Sample Request:

```
<mmRequest command="reconcileElements" station="ID" [version="1"] [client=""] [userData=""]>
   <contents>
      <elements>
         <element mode=”add|drop|playStart|playEnd” [airTime=”yyyy-mm-ddThh:nn:ss”]
[runTime=”mm:ss”] [pos=”before|after|replace”] [songId=”x”] [cutId=”x”] [historyId=”x”]
[airTimeNew=”yyyy-mm-ddThh:nn:ss”] [songIdNew=”x”] [cutIdNew=”x”] />
      </elements>
   </contents>
</mmRequest>
```

Request Notes:
The elements section contains the actual element(s) that were affected by the change. Only the elements actually affected should be included, with one element tag for each. While a simple add or drop would generally only affect a single element, a move might include a drop and an add, and a swap might include two adds with the mode=replace attribute. The following attributes are available for the element tags:

mode – indicates the type of change for this element: add, drop, playStart and playEnd are currently supported.

airTime – indicates that starting time that the element aired or is scheduled to air. This is required for every request and is used to determine which hour the desired element is located in. You can narrow down the search for your target element and make a more exact match by using one or more of the additional optional properties that follow. Starting with MusicMaster 5.0 SR-3, this is optional when you also specify a valid value in the historyId property. In that case the historyId value will be used to determine the hour containing the desired element.

historyId – the MusicMaster History ID of the element, if known. This can be obtained using getSchedule and is the most exact way to target a specific element. If a valid History ID is specified, this will be used to determine which hour the desired element is located in. (Changed in 5.0 SR-3)

runtime – indicates the actual or schedule run length. This is optional. If not included, the MusicMaster runtime will be used.

Pos – indicates the relative position to the specified element for an add operation. You can replace the specified, or insert just before or after it. (before|after|replace)

songId – the MusicMaster Song ID of the element, if known.

cutId – the element ID from the third party system, if known.

airTimeNew – indicates that starting time that the element aired or is scheduled to air. This is only used for add operations.

songIdNew – the MusicMaster Song ID of the element, if known for the new song. This is only used for add operations.

cutIdNew – the element ID from the third party system for the new song. This is only used for add operations.

Notes:

See also the newer function modifySchedule
When searching for an element by airtime only, the element selected will be the first one from the start of the hour that is scheduled to start at the specified airtime, or the one that starts before and runs through the specified airtime.
Even though songIdNew and cutIdNew are both listed as optional, one or the other must be included when performing an add operation.
Performace can be improved by using the MusicMaster songId for this function.

Sample Reply:

```
<mmReply command="reconcileElements" station="ID" version="1" [userData=""] status="???????">
   [<errors>
      <error>message</error>
   </errors>]
   [<results>
      <result [mode] [songId] [historyId] status=″ok|error″></result>
   </results>]
</mmReply>
```

Reply Notes:
This function does not return any schedule data. Instead, the server performs the reconciliation and returns a status code and any error messages. If a non-generic interface is in use, the method used to reconcile the schedule may vary.

Compatibility and Version Info:
Available in all versions

Changed the histId attribute to historyId in 4.0 SR-19. histId is still valid here, but historyId is more consistent with other functions.

Starting with 5.0 SR-3, the airTime property is no longer required when you have the history ID of the element you wish to change. It is recommended that you still specify an air time for compatibility with older versions. The history ID value will now take precedence over the specified airtime. [5001]

Added the results section to the reply message in 5.0 SR-3. [5001]

modifySchedule

Description:
Performs individual schedule updates in MusicMaster. You can use this message to change one or more elements. This would generally be used to drop or insert song, or a more complex move or swap operation. You can also use this message to change non-music element types and element properties.

Sample Request:

```
<mmRequest command="modifySchedule" station="ID" [version="1"] [client=""] [userData=""]>
   <contents>
      <changes>
         <change>
            <target airTime="" [class=""] [historyId=""] [runTime=""] [songId=""] [cutId=""] />
            <action mode="add|drop|modify" [pos="before|after|replace"] [airTime=""] [runTime=""]
[class=""] [songId=""] [cutId=""] [category=””] [sweep=""] [text=""] [mergeStart=""] [mergeEnd=""]
[mergeCode=""] [markType=””] [markMinTime=””] [markMaxTime=””] [markResetTime=””] [markReset=””]
[properties=""] [transition=""] [user1=””] [user2=””] />
         </change>
      </changes>
   </contents>
</mmRequest>
```

Request Notes:
The changes section contains the actual change(s) you wish to perform. Each will contain a child tag with the target element where you wish to make the change, and the action you wish to perform on that element. You should only include the required attributes and the optional ones that you actually need.

The following attributes are available for the change target tag:

airTime – indicates that starting time that the element aired or is scheduled to air. This is required for every request and is used to determine which hour the desired element is located in. You can narrow down the search for your target element and make a more exact match by using one or more of the additional optional properties that follow. Starting with MusicMaster 5.0 SR-3, this is optional when you also specify a valid value in the historyId property. In that case the historyId value will be used to determine the hour containing the desired element.

historyId – the MusicMaster History ID of the element, if known. This can be obtained using getSchedule and is the most exact way to target a specific element. If a valid History ID is specified, this will be used to determine which hour the desired element is located in. (Changed in 5.0 SR-3)

class – indicates the type of element you are searching for. M=Scheduled Music, U=Unscheduled Music, T=Traffic Merge, L=Lognote, H=Hour Marker, *=Timing Checkpoint

runtime – indicates the actual schedule run length of the element you are searching for.

songId – the MusicMaster Song ID of the element, if known.

cutId – the element ID from the third party system, if known.

The following attributes are available for the change action tag:

mode – indicates the type of change for this element: add, drop, and modify are currently supported. This is required.

Pos – indicates the relative position to the specified element for an add operation. You can replace the specified element, or insert an element just before or after it. (before|after|replace). Replace is the default value if not specified.

airTime – indicates that starting time that the element aired or is scheduled to air. This is only used for add operations. (mm:ss)

class – the type of element you wish to add. You can also use this to change an existing element to a different type. M=Scheduled Music, Unscheduled Fixed Category, T=Traffic Merge, L=Lognote, *=Timing Checkpoint. You must specify the class if you are changing it to something different from the current value.

Additional properties are available based on the type of element you are modifying, based on the value of the class property:

Property	Purpose	Valid Classes				
		M	U	L	T	*
runTime	Runtime of the element you are adding or changing. When modifying an existing element the existing runtime will remain if not specified here (mm:ss)	X		X	X	
songId	MusicMaster Song ID of the element, if known for the new song	X				
cutId	Element ID from the third party system for the new song	X				
category	Category code (not the numeric ID)		X			
sweep	Used to indicate whether the element should act as a sweep marker			X	X	X
text	The actual text for a lognote (255 char max)			X		
mergeStart	Start time for spot catching (mm:ss)				X	
mergeEnd	End time for spot catching (mm:ss)				X	
mergeCode	Merge code for spot catching				X	
markType	This is the type of timing checkpoint ('hour', 'sweep', 'none')					X
markMinTime	Earliest acceptable airtime for a tested checkpoint (mm:ss)					X
markMaxTime	Earliest acceptable airtime for a tested checkpoint (mm:ss)					X
markResetTime	Forced reset airtime for a timing checkpoint					X
markReset	Indicates where the forced airtime should be reset. You can turn each individual property on or off by specifying the letter code preceded by a plus or minus symbol. (-L +L –E +E –R +R –P +P)					X
properties	This is used to adjust the element properties of any element. If not included, the existing properties will not be changed. You can turn each individual property on or off by specifying the letter code preceded by a plus or minus symbol. (-P +P –R +R –L +L) P=reset in play history, R=reset for rolling air times, L=reset when exporting program logs	X	X	X	X	
transition	this is used to adjust the transition code element property on any element. If not included, the existing transition code will not be changed	X	X	X	X	
user1, user2	these are used to adjust user note fields that are associated with schedule elements	X	X	X	X	

Notes:
When searching for an element by airtime only, the element selected will be the first one from the start of the hour that is scheduled to start at the specified airtime, or the one that starts before and runs through the specified airtime.

Performance and accuracy can be improved by using the historyId attribute for the target element and the songId attribute when adding or changing music elements.

Sample Reply:

```
<mmReply command="modifySchedule" station="ID" version="1" [userData=""] status="???????">
   [<errors>
      <error>message</error>
   </errors>]
   [<results>
      <result [mode] [songId] [historyId] status=”ok|error”></result>
   </results>]
</mmReply>
```

Reply Notes:

This function does not return any schedule data. Instead, the server performs the reconciliation and returns a status code and any error messages. If a non-generic interface is in use, the method used to reconcile the schedule may vary.

Compatibility and Version Info:

Available in MusicMaster 4.0 SR-19 and later

Starting with 5.0 SR-3, the airTime property is no longer required when you have the history ID of the element you wish to change. It is recommended that you still specify an air time for compatibility with older versions. The history ID value will now take precedence over the specified airtime. [5001]

Added the results section to the reply message in 5.0 SR-3. [5001]

Added the ability to create and modify timing checkpoint elements in 5.0 SR-16 [5008]

getReplacementSongs

Description:
Returns a list of songs to potentially be scheduled at a specified time that test well against the rules defined in MusicMaster.

Sample Request:

```
<mmRequest command="getReplacementSongs" station="ID" [version="1"] [client=""] [userData=""]>
   <contents>
      <element airTime="yyyy-mm-ddThh:nn:ss" [pos="before|after|replace"] [songId="x"] [cutId="x"]
[historyId="x"] />
      [<testFilter>perfectOnly|breakable</testFilter>]
      [<maxSongs>10</maxSongs>]
      [<goalSort>1</goalSort>]
      [<query>
         [<categories>
            <category [id="1"] [code="A"]/>
         </categories>]
         [<filters>
            <filter [type="fieldId|fieldName"] target="Artist" operator="contains" value="love"/>
         </filters>]
         [<sortKeys>
            <sortKey [type="fieldId|fieldName"] target="Artist" order="asc|desc"/>
         </sortKeys>]
         [<properties>
            <property name="propname">value</property>
         </properties>]
      </query>]
      [<fields>
         <field id="101" />
         <field name="Title" />
      </fields>]
   </contents>
</mmRequest>
```

Request Notes:
The element tag defines the element location where you want to see replacement songs. You can choose to replace an existing element, or insert an element before or after an existing element. The element tag has the following attributes:

airTime – indicates that starting time that the element aired or is scheduled to air. This is required for every request and is used to determine which hour the desired element is located in. You can narrow down the search for your target element and make a more exact match by using one or more of the additional optional properties that follow. Starting with MusicMaster 5.0 SR-3, this is optional when you also specify a valid value in the historyId property. In that case the historyId value will be used to determine the hour containing the desired element.

historyId – the MusicMaster History ID of the element, if known. This can be obtained using getSchedule and is the most exact way to target a specific element. If a valid History ID is specified, this will be used to determine which hour the desired element is located in. (Changed in 5.0 SR-3)

Pos – indicates the relative position to the specified element for an add operation. You can replace the specified, or insert just before or after it. (before|after|replace)

songId – the MusicMaster Song ID of the element, if known.

cutId – the element ID from the third party system.

Note that while songId and cutId are both listed as optional, one or the other must be included. Performace can be improved by using the MusicMaster songId for this function.

testFilter – specifies the quality of replacement songs that may be returned. The value perfect means to only return songs that violate no rules. If there are no such songs, then none will be returned. Breakable means to only return only perfect songs and songs that violate breakable rules if necessary to reach the maxSongs count. If not specified, the default value is breakable.

maxSongs – the maximum number of best replacement songs that will be returned. Currently this has a maximum value of 99. If not specified, the default value is 10.

goalSort – indicates that you would like the song list that will be returned tested against the Optimum Scheduling Goals (if available). When this is specified, the song with the best goal score will be returned first and the worst will be last. To activate this function, add this tag with a value of 1. [Requires 4.0 SR-20 or later]

query – You can optionally search the library for specific replacement songs using a query. The actual search request is provided in the contents/query section. All sections and tags are optional. You only need to include the portions necessary to define the desired search query. Using a query will override the list of categories that would normally be searched for the replacement songs. See appendix B for complete details of the Nexus query tags.

Note that when searching for replacement songs for an existing scheduled element, the default list will be all songs from the same song as the one being replaced. When inserting a position, you should define at least the categories part of a query to indicate what songs to include in the search. Creating a query that will test a very large list of potential songs can be very time consuming and should be avoided.

fields - The fields section must contain one field tag for each song field you wish to include in the reply message. You can use the id attribute with the internal field ID codes, or a name attribute with the user-specified field name. Only the specified fields will be returned for each song. However, the internal song ID number will always be included in the reply. You can obtain a list of available fields with the getFields message. See appendix C for addition options that can be used with your fields specification.

Note that the fields section is optional. If not specified, only the song id, runtime and the primary and secondary fields will be returned. Typically, primary and secondary fields are defined as title and artist.

Sample Reply:

```
<mmReply command="getReplacementSongs" station="ID" version="1" [userData=""] status="???????">
   <contents>
      <songList>
         <song songId="1">
            <field id="100" name="Description">Beatles / Yesterday</field>
            <field id="101" name="Artist">Beatles</field>
         </song>
      </songList>
   </contents>
</mmReply>
```

Reply Notes:

The songid attribute of the SONG tag is the MusicMaster song ID number. A description including the primary and secondary fields is also included, as well as the song category. You can use the GetSongsByQuery function to return additional song details if you wish. A custom list of return fields may be supported by this command in a future release. Currently, this function only returns the top X perfect songs in most to least rested order.

Compatibility and Version Info:

Available in all versions

Starting with 5.0 SR-3, the airTime property is no longer required when you have the history ID of the element you wish to change. It is recommended that you still specify an air time for compatibility with older versions. The history ID value will now take precedence over the specified airtime. [5001]

getHistory

Description:
Returns a list of song airplay dates for a specified song.

Sample Request:
```
<mmRequest command="getHistory" station="ID" [version="1"] [client=""] [userData=""]>
   <contents>
      <song [songId="1"] [cutId="x"] />
      [<maxHistories>10</maxHistories>]
   </contents>
</mmRequest>
```

Request Notes:
The song tag defines the song that you wish to retrieve histories for. You can lookup the song by it's MusicMaster Song ID or it's element ID from a third party system.

songId – the MusicMaster Song ID of the element, if known.

cutId – the element ID from the third party system.

Note that while songId and cutId are both listed as optional, one or the other must be included. Performace can be improved by using the MusicMaster songId for this function.

maxHistories – the maximum number of histories that will be returned. By default, all available active histories will be returned. These always begin with the newest history and work backward in time.

Sample Reply:
```
<mmReply command="getHistory" station="ID" version="1" [userData=""] status="???????">
   <contents>
      <historyList>
         <history historyId="1" airdate=″yyyy-mm-dd″ airTime=″hh:mm:ss″ runtime=″mm:ss″ />
      </historyList>
   </contents>
</mmReply>
```

Reply Notes:

Compatibility and Version Info:
Available with MusicMaster 4.0 SR-18 and later

getSongHistory

Description:
Returns a list of airplay histories for one or more specified songs.

Sample Request:
```
<mmRequest command="getSongHistory" station="ID" [version="1"] [client=""] [userData=""]>
   <contents>
      [<maxHistories>10</maxHistories>]
      <songList>
         <song [songId="1"] [cutId="x"] />
      </songList>
   </contents>
</mmRequest>
```

Request Notes:
maxHistories – the maximum number of histories that will be returned. By default, all available active histories will be returned. These always begin with the newest history and work backward in time.

The song tags define the songs that you wish to retrieve histories for. You can look up songs by their MusicMaster Song ID or their element ID from a third party system.

songId – the MusicMaster Song ID of the element, if known.

cutId – the element ID from the third party system.

Note that while songId and cutId are both listed as optional, one or the other must be included. Performace can be improved by using the MusicMaster songId for this function.

Sample Reply:
```
<mmReply command="getSongHistory" station="ID" version="1" [userData=""] status="???????">
   <contents>
      <songList>
         <song songId=″1″>
            <historyList>
               <history historyId="1" airDate=″yyyy-mm-dd″ airTime=″hh:mm:ss″ runTime=″mm:ss″ />
            </historyList>
         </song>
      </songList>
   </contents>
</mmReply>
```

Reply Notes:

Compatibility and Version Info:
Available with MusicMaster 5.0 SR-14 and later

getKeywordHistory

Description:
Returns a list of airplay histories for one or more specified keywords.

Sample Request:
```
<mmRequest command="getKeywordHistory" station="ID" [version="1"] [client=""] [userData=""]>
   <contents>
      [<maxHistories>10</maxHistories>]
      <keywordList>
         <keyword [fieldId="101"] [fieldName="Artist"] value="Beatles" />
      </keywordList>
   </contents>
</mmRequest>
```

Request Notes:
maxHistories – the maximum number of histories that will be returned. By default, all available active histories will be returned. These always begin with the newest history and work backward in time.

The keyword tags define the keywords that you wish to retrieve histories for. You must also indicate the keyword field to search. Fields can be specified by name or ID.

fieldId – the MusicMaster Field ID to search

fieldName – the name of the MusicMaster Field to search

value – the Keyword you wish to search for

Note that while fieldId and fieldName are both listed as optional, one or the other must be included.

Sample Reply:
```
<mmReply command="getKeywordHistory" station="ID" version="1" [userData=""] status="???????">
   <contents>
      <keywordList>
         <keyword fieldId="102" fieldName="Artist" value="Beatles">
            <historyList>
               <history historyId="1" airDate=″yyyy-mm-dd″ airTime=″hh:mm:ss″ runTime=″mm:ss″ />
            </historyList>
         </keyword>
      </keywordList>
   </contents>
</mmReply>
```

Reply Notes:

Compatibility and Version Info:
Available with MusicMaster 5.0 SR-14 and later

getScheduleHistory

Description:
Returns a list of song play histories matching a time range and library query.

Sample Request:
```
<mmRequest command="getScheduleHistory" station="ID" [version="1"] [client=""] [userData=""]>
   <contents>
      <startTime>yyyy-mm-ddThh:00:00</startTime>
      <endTime> yyyy-mm-ddThh:00:00</endTime>
      <query>
         [<categories>
            <category [id=”1”] [code="A"]/>
         </categories>]
         [<filters>
            <filter [type=”fieldId|fieldName”] target="Artist" operator="contains" value="love"/>
         </filters>]
         [<sortKeys>
            <sortKey [type=”fieldId|fieldName”] target="Artist" order="asc|desc"/>
         </sortKeys>]
         [<properties>
            <property name=”propname”>value</property>
         </properties>]
      </query>
   </contents>
</mmRequest>
```

Request Notes:
startTime – The first hour to include in the reply. This is the first full hour that will be included.

endTime – The last hour to include in the reply. This is the last full hour that will be included.

properties – Include one property tag for each query property you wish to specify. The following properties are currently available:

query - The actual search request is provided in the contents/query section. All sections and tags are optional. You only need to include the portions necessary to define the desired search query. See appendix B for complete details of the Nexus query tags.

Note: For this function, the sortKeys portion of the query does not have any effect.

Sample Reply:
```
<mmReply command="getScheduleHistory" station="ID" version="1" [userData=""] status="???????">
   <contents>
      <historyList>
         <history songId=”999” categoryId=”1” airdate=”yyyy-mm-dd” airTime=”hh:mm:ss”
runtime=”mm:ss” />
      </historyList>
   </contents>
</mmReply>
```

Reply Notes:

Compatibility and Version Info:
Available with MusicMaster 4.0 SR-20 and later

getFormatLists

Description:
This will return a list of the names of all available Format Lists.

Sample Request:
```
<mmRequest command="getFormatLists" station="ID" [version="1"] [client=""] [userData=""] />
```

Sample Reply:
```
<mmReply command="getFormatLists" station="ID" version="1" [userData=""] status="???????">
   <contents>
      <formatLists>
         <formatList name="Format List 1"/>
         <formatList name="Another Format List"/>
         <formatList name="Really Cool Format List"/>
      </formatLists>
   </contents>
</mmReply>
```

Reply Notes:
This will return a single formatLists section containing one formatList tag for each available Format Lists. The name attribute will return the text name of each list.

Compatibility and Version Info:
Available with MusicMaster 5.0 SR-14 and later [5006]

getRules

Description:
This will return a list of rules assigned to the specified category that is similar to the Category Rule List function available in the Rule Tree Editor. This rule list is for review only. No changes to rules can be made via Nexus.

Sample Request:
```
<mmRequest command="getRules" station="ID" [version="1"] [client=""] [userData=""]>
   <contents>
          <categories>
             <category [id="1"] [code="A"]/>
          </categories>
   </contents>
</mmRequest>
```

Request Notes:
You must specify a list of one or more valid categories that you wish to see rules for. You can use the category ID or Code values as returned from the getCategories function.

Sample Reply:
```
<mmReply command="getRules" station="ID" version="1" [userData=""] status="???????">
   <contents>
      <ruleList category=”X” catId=”1”>
         <rule allCategories="True" folder="0" folderName="Unbreakable" description="Artist
Keywords - Keyword Time Separation [fixed: 0:20]"/>
         <group allCategories="True" folder="0" folderName="Unbreakable" description="Rule Group
(Gender) [Always Tested]">
            <rule allCategories="True" inGroup="True" folder="0" folderName="Unbreakable"
description="Gender F-Female - Maximum in Sequence [1]"/>
         </group>
         <rule folder="0" folderName="Unbreakable" description="Artist Keywords - Keyword Adjusted
Time Separation [75% - auto: 2:10 - By Cat]"/>
         <rule folder="0" folderName="Unbreakable" description="Minimum Rest [3:30]"/>
      </ruleList>
   </contents>
</mmReply>
```

Reply Notes:
One ruleList section will be included for each category requested. This will include tags for each rule and rule group available for that category. This will include All Categories rules if appropriate. Built-in rules and scheduling goals are not included. The list will be in the same order as would be shown in the Category Rule List viewer in MusicMaster. Each rule tag indicates the rule folder level, an indication of whether the rule is in a group or not, and the text rule description as shown in the Rule Tree editor. The following tag attributes may be found here:

allCategories – set to True is this was an All Categories rule. Not included for individual category rules.

folder - rule folder level. 0 = Unbreakable
folderName – the text name of the rule folder

inGroup – set to True is this rule is in a rule group. Not included otherwise.

description – text description of the rule or rule group

Compatibility and Version Info:
Available with MusicMaster 5.0 SR-3 and later [5001]

Section III: Revision History

API Version	MusicMaster Version	Summary of changes
5008	5.0 SR-16 6.0 SR-2	Added support for timing checkpoints and unscheduled fixed category elements to modifySchedule Added vendor-specific publishMeta support for Enco (6.0 only)
5007	5.0 SR-15	Added the publishMetaData function Added field 27 (Last Edit Date) to the default list of fields returned bygetSongInfo.
5006	5.0 SR-14	Added getFormatLists, getSongHistory and getKeywordHistory functions
5005	5.0 SR-12	Added cutId support to updateSongs and deleteSongs. These functions will also return an error message if an invalid song is specified.
5004	5.0 SR-11	Messages sent by Nexus Server will be encoded as UTF-8 by default When performing a song query on a Length type field, including Runtime, you can now specify the target value in either total seconds ":sss" or "mm:ss" format.
5003	5.0 SR-8	The getSchedule function will now include a properties attribute with each schedule element that includes flags for that element. L=Locked, E=No Export, R=No Reconcile, P=No Print, M=Manually Scheduled.
5002	5.0 SR-4	When using the Nexus ReconcileElements function with mode=playStart or playEnd, it is now possible for the target element to move to a different hour if the new airtime is in an hour that is not the same as the original position. Note that the intention of this function was not to move elements around in the log, but rather to adjust airtimes from what was estimated at schedule time to what actually happened at air time. It is recommended that you use your own logic to move elements around as needed, rather than relying on the guesswork that this function can provide. This additional logic was mainly added to improve the situation where an element was scheduled very close to an hour border and the actual air time pushed it just inside the bordering hour. Note that an element that ends up moving to a different hour will receive a new history ID, which is the same as if you had moved the element yourself.
5001	5.0 SR-3	Added the getAPIVersion command Added the getRules command Added the results section to the reply messages for reconcileElements and modifySchedule Changed the element targeting behavior of reconcileElements, modifySchedule and getReplacementSongs to use the historyId value to locate the proper hour to load, if specified, instead of relying on the specified airTime

Appendix A: Station Properties

The following properties can be retrieved and adjusted using the getStationProperty and setStationProperty API functions.

normalizeTimes - Determines whether minutes in times may exceed 59 in standard MM form. 0 is the default value and allows hour times to exceed 59:59. If set to 1, hour times beyond 59:59 will be returned as 59:59.

Appendix B: Nexus Library Query Tags

Request messages that allow a custom query of the music library all use the same query tag set. This allows you to specify the categories to search, search filters, and sort keys. The tag can contain any or all of these sections:

```
<query>
   [<categories>
      <category [id=”1”] [code="A"]/>
      </categories>]
   [<filters>
      <filter [type=”fieldId|fieldName”] target="Artist" operator="contains" value="love"/>
   </filters>]
   [<sortKeys>
      <sortKey [type=”fieldId|fieldName”] target="Artist" order="asc|desc"/>
   </sortKeys>]
   [<properties>
      <property name=”propname”>value</property>
   </properties>]
</query>
```

categories - Include one category tag for each category you wish to include in the search. You may use the id attribute to specify a category Id number, or the code attribute to specify the user-defined code for a category. You can obtain a list of available categories with the getCategories message. You can also use a code value of * to search the entire library, and ? to search the uncategorized songs. Note that all categories are searched when the categories section is omitted.

filters - Include one filter tag for each search filter you wish to define. You can specify which field to filter on using an Id code or text name. You can specify which type you’ve used in the type attribute and the actual code or id in the target attribute. If the type attribute is not included, a best-guess will be made based on the target attribute entry. You can obtain a list of available fields with the getFields message. The operator attribute specifies the type of search to perform on this field and the value attribute defined the value you to search for. Please note that each type of field supports a specific subset of the available operators. These are the valid operators that you may use:

Operator	**Meaning**	**Supported Field Type(s)**
x.. or beginswith	Field begins with	Text, Keyword, Memo, Attribute In-Out/Combo
Blank	Field is blank	All except Boolean
..x.. or contains	Field contains	Text, Keyword, Memo
Containsall	Field contains all of	Attribute Multiple
Containsany	Field contains any of	All Attribute types
Containsonly	Field contains only	Attribute Multiple
..x or endswith	Field ends with	Text, Keyword, Memo, Attribute In-Out/Combo
= or equals	Field equals	All except Boolean
=0= or false	Field is False	Boolean only
> or greaterthan	Field value greater than	Text, Keyword, Numeric, Date, Time, Runtime
>= or greaterorequal	Field value greater than or equal to	Text, Keyword, Numeric, Date, Time, Runtime
Notrivia	Has No Trivia	Keyword, Song
Hastrivia	Has Trivia	Keyword, Song
< or lessthan	Field value less than	Text, Keyword, Numeric, Date, Time, Runtime
<= or lessorequal	Field value less than or equal to	Text, Keyword, Numeric, Date, Time, Runtime
/x.. or notbeginswith	Field does not begin with	Text, Keyword, Memo, Attribute In-Out/Combo

Notblank	Field is not blank	All except Boolean
/..x.. or notcontains	Field does not contain	Text, Keyword, Memo
Notcontainsall	Field does not contain all of	Attribute Multiple
Notcontainsany	Field does not contain any of	All Attribute types
Notcontainsonly	Field does not contain only	Attributes Multiple
/..x or notendswith	Field does not end with	Text, Keyword, Memo, Attribute In-Out/Combo
/= or notequal	Field does not equal	All except Boolean
Notoverall	Field overall value is not	Attribute Combo
Overall	Field overall value is	Attribute Combo
=1= or true	Field is True	Boolean only
~ or closeto	Field value is close to	Runtime
/~ or notcloseto	Field value not close to [Note: Added in 4.0 SR-24]	Runtime

sortKeys - Include one sortKey tag for each sort level you wish to define. You can specify which field to filter on using an Id code or text name. You can specify which type you've used in the type attribute and the actual code or id in the target attribute. If the type attribute is not included, a best-guess will be made based on the target attribute entry. You can obtain a list of available fields with the getFields message. Note that we plan to support some special non-song field type values for search filters and sort keys in the future, which is why we've used this type of syntax. The order attribute defines the sort order and can be specified as asc or desc.

properties – Include one property tag for each query property you wish to specify.

filterMatchMode – The filterMatchMode property determines how songs will match when more than one query filter has been specified. By default, AND mode is used so songs must match all of the specified query filters. You can specify AND mode in this property by setting this value to 'all'. You can also change it to use OR mode where any one of your filters must match by setting this value to 'any':

```
<properties>
   <property name=”filterMatchMode”>[any][all]</property>
</properties>
```

firstRec – firstRec (and maxRecs) can be used to retrieve a long list of songs in pages. These properties allow you to specify the first record from the list to retrieve, and the total number you would like to get. You can perform the same query multiple times with different values to get the list in pieces.

maxRecs – maxRecs (and firstRec) can be used to retrieve a long list of songs in pages. These properties allow you to specify the first record from the list to retrieve, and the total number you would like to get. You can perform the same query multiple times with different values to get the list in pieces. [Note: available in 4.0 SR-19 and later only]

```
<properties>
   <property name=”firstRec”>1</property>
   <property name=”maxRecs”>25</property>
</properties>
```

Appendix C: Song Field Request Options

When requesting a list of songs to be returned, the <fields> tag is used to indicate what song fields you would like to have included.

```
<fields>
   <field id="101" />
   <field name="Title" />
</fields>
```

The fields section must contain one field tag for each song field you wish to include in the reply message. You can use the id attribute with the internal field ID codes, or a name attribute with the user-specified field name. You can obtain a list of available fields with the getFields message.

In addition to the normal fields, a special field 100 is available. When specified, the song “description” is returned, which includes the user-selected primary and secondary fields.

For multiple keyword fields, you can add the keyword attribute that allows you to select a specific keyword from the list. This example shows how to get the first two keywords from a field.

```
<fields>
   <field id="101" />
   <field name="Artist Keywords" keyword=″1″ />
   <field name="Artist Keywords" keyword=″2″ />
</fields>
```

You can also use the trivia attribute to return song or keyword trivia lines. This example shows how to retrieve the active trivia for the song, and the fixed trivia lines for the specified Artist Keyword.

```
<fields>
   <field id="100" trivia=″active″ />
   <field name="Artist Keywords" keyword=″1″ trivia=″fixed″ />
</fields>
```

The options for the trivia attribute are fixed (all fixed trivia lines only), rotate (the next available rotating trivia line), and active (The active fixed lines and next rotating line).

Finally, a script function option is available for field 100. You can include a script tag that includes a script function like the ones that can be used in the standard log export design editor and other places in MusicMaster. This can allow you to have some data formatting done before the information is returned by Nexus.

Compatibility and Version Info:
These additional options were added along with MusicMaster 4.0 SR-12