Cantabile Network Server

Configuration Guide and Protocol Specification

Table of Contents

1 Introduction

Cantabile 2.0 includes a simple network server that can be used to control certain aspects of Cantabile by either typing commands at a telnet client or by writing software that programmatically controls it.

This document provides a brief overview of configuring the server and basic details on its communication protocol.

Functionality Covered

The functionality provided by the network server does not cover all of Cantabile's features. Most significantly it doesn't include editing capability (eg: inserting plugins, changing session lists etc...) as it is expected this would still be done through Cantabile's UI.

The following is a brief summary of what is possible:

- On connection, Cantabile reports certain information such as version and edition.
- Responds to simple commands that are in a format not unlike most command line utilities.
- Has a command that allows invoking any MIDI assignable functionality in Cantabile. (ie: anything that can be assigned to a MIDI controller can also be directly controlled through the network interface – obviously, without the need to actually create the assignment)
- Allows listing out the sub-sessions of the loaded session and the content of the loaded session list.
- Allows querying for certain information such as the filename of the loaded session, the name of the selected sub-session, the name of the loaded session list etc...
- Provides notifications of certain events such as a different session or sub-session being loaded, transport and engine state changes etc...
- Includes built in help for listing available commands, the syntax of each command and also for listing out MIDI assignable items that can be invoked.

Supported Editions

The network server is support in Cantabile 2.0 Solo and Cantabile 2.0 Performer on both x86 and x64 platforms.

It is not included in Cantabile Lite primarily since it doesn't support MIDI Controller Assignments - making the network server less useful anyway.

Proposed Changes and Improvements

The following describes currently proposed changes and improvements to the network server:

- A notification message on Cantabile shutting down.

- Ability to activate a session list entry by actual index rather than relying on invoking MIDI assignments.
- Virtual directories and remove directory listing. Currently the network API uses fully qualified path names for session and session list filenames. These names are somewhat useless to remote client so propose a virtual directory structure (eg: $/SessionLists/...). These root directories should be configurable through the network settings and network commands added to list files in those folders.
- Unicode support. Currently all strings are in ansi format though internally Cantabile uses Unicode.

Feedback and Support

Currently, the network server functionality is experimental and not yet officially supported by Topten Software. Both the server functionality and this document are subject to change and provided only to assist developers who have expressed an interest in automated control over Cantabile.

IMPORTANT

If you write software against this protocol, expect to have to change your code - this protocol will almost certainly be changing before being officially supported.

Feedback and contact details can be found here:

- http://www.cantabilesoftware.com/contact

Support is available from the normal support channels described here:

- http://www.cantabilesoftware.com/support

Right to Use

Topten Software grants any legitimate software developer the right to use the information described in this document to develop third party tools and applications that can communicate with Cantabile on the following conditions:

- The product is developed solely to benefit its users and not designed to cause harm, loss of property, loss of information, or otherwise interfere with in a negative or destructive way Cantabile's functionality or the functionality of the PC on which Cantabile is running.
- The developer understands that users of software they develop using this interface must have a properly licensed installation of Cantabile before using the network server functionality.
- Topten Software is made aware of the final product before it is made available to the general public.

Topten Software claims no right to software developed using this interface and developers many sell or give away such products as they see fit.

Revision History

12 Feb 2009	First Draft
18 Feb 2009	Change response code for list items from “R” to “L”
24 Feb 2009	Added proposed changes and improvements
25 Feb 2009	First Public Draft

2 Configuring Cantabile

Enabling the Network Server

By default the network server is disabled. Since this is currently an unsupported feature there is no UI for enabling it and it must be manually enabled in Cantabile's settings file.

- Open Cantabile's settings.ini file, typically located in the following folder:
- On Windows 2000/XP - C:\Documents and Settings\<yourusername>\Application Data\Topten Software\Cantabile 2.0
- Vista - C:\Users\<yourusername>\AppData\Roaming\Topten Software\Cantabile 2.0
- Add a section to the file with the following settings (If the port setting is omitted, 61023 is used).

```
TelnetServer
{
  Enabled=1;
  Port=61023;
}
```

- Start Cantabile.
- You may be prompted by Window's firewall to unblock access, or you may need to configure other firewall software to allow access before it will work.
- View Cantabile's log files to confirm the network server started correctly. You should see entries similar to:

```
08:17:24 [A2]: network - starting server on port 61023
08:17:24 [A2]: network - started
```

Testing with a Telnet Client

Once configured correctly and running, you should be able to connect to Cantabile using any telnet client. Windows XP and Vista both include a telnet client though on Vista it's not installed by default. Eg: from a command prompt:

```
telnet localhost 61023
```

On successful connection you will be presented with some informational messages about Cantabile and a prompt:

```
I:Cantabile TelNet Server
I:Edition:Performer
I:Version:2.0.0.2011
I:SessionID:1
>>
```

From here, you can type commands to control Cantabile. Type ‘help’ to get a list of available commands and ‘help <commandname>’ to get help for a specific command. Eg:

```
>> help engine stop
R:engine stop - Stops the audio engine
>>
```

3 Network Protocol

Cantabile's network protocol is a simple text based interface that is designed to be easily used manually or programmatically. Mostly it is self explanatory by playing with it from a telnet client however certain aspects should be formalized.

Connection

On connection to the server, Cantabile will immediately send a series of telnet configuration codes which can be ignored by most applications.

Following this will be sent a series of information messages.

The simplest way to detect the end of the telnet configuration codes is to wait for the first 'I:' message.

Interactive vs Non-interactive Modes

The network server operates in one of two modes – either interactive or non-interactive. The mode is determined automatically and interactive mode is selected if the server detects a telnet client is connected.

In interactive mode, the network server writes prompts and allows various editing operations.

In non-interactive mode, the communication is strictly text in/text out.

Response Codes

All communication from the network server is line based, commencing with a single character response code, followed by a colon, the line content and terminated by a carriage return/line feed pair.

The following response codes are used:

R	The line content is the response to the last received command.
L	An entry in a list in response to the last received command.
E	An error in the last received command.
I	An informational message, currently only sent upon connection.
N	An asynchronous notification (ie: an event).

Command Format

All commands conform to a simple command line like syntax, commencing with one or more key words (eg: "application exit") and followed by zero or more arguments, separated by space characters.

Arguments containing spaces should be quoted. Eg: load "my file with spaces.cantabile"

Quoted arguments with embedded quotes should escape the contained quotes using a double quote. Eg:

```
load "my file with ""spaces"" and quotes.cantabile"
```

Named arguments are specified using /<name>:<value> syntax with no spaces around the colon. Eg:

```
load /subsession:"MySubsession"
```

Response String Escaping

All strings returned from the server are escaped using the same rules as for command arguments except they're always quoted even if there are no internal spaces.

Returning Lists

Some commands return a list of items in which case each item will be on its own line prefixed with an 'L' response code. Fields within each row are comma separated.

Following the list will be a single 'R' response code.

Notification Codes

Most notifications are sent automatically and may be redundant (eg: pausing the transport may result in multiple transport pause notifications).

Notifications of list changes are only sent if the list has been queried since the last change notification.

For example, changes to the set of sub-sessions in session will only be sent if the client has previously queried for the list of sub-sessions. Similarly only one notification will be sent until the list is asked for again by the client, after which another notification will be sent the next time the list is changed.

4 Invoking MIDI Assignable Items

Since Cantabile's MIDI Controller Assignment functionality already allows control of many aspects of Cantabile, rather than recoding all of this for the network server, it simply allows invocation of the these items.

Anything that can be controlled through a MIDI controller assignment can also be controlled through the network server.

Object and Controls

The set of MIDI assignable controls is divided into groups called "objects". Each object has a number of controls. To get a list of objects, use the "invoke list" command.

```
>> invoke list
L:"general"
L:"masterLevels"
L:"subSessions"
L:"sessionLists"
L:"triggers"
L:"metronome"
L:"transport"
L:"audioRecorder"
L:"midiRecorder"
L:"globalRackControls"
L:"globalPluginControls"
L:"piano_concerto_58_1_(c)galimberti.mid"
L:"Rack 1"
L:"Rack 1 - mda ePiano"
R:OK
>>
```

To get a list of controls on an object, use the invoke list command again also passing the object name

```
>> invoke list masterLevels
L:variable,"masterLevels","masterInputGain","Master Input Gain"
L:latch,"masterLevels","masterInputEnabled","Master Input Enabled"
L:variable,"masterLevels","masterOutputGain","Master Output Gain"
L:latch,"masterLevels","masterOutputEnabled","Master Output Enabled"
R:OK
>>
```

Each item in the response includes the following information:

- The control type (see below),
- The object name
- The control name
- The control description

Variables, Latches and Commands

Each control is one of three types:

Commands	No parameters and simply invoke an action (eg: transport start)
Latches	Have one of three arguments on/off/toggle. eg: (transportPlayPause)
Variables	Take a value between 0.0 and 1.0 and control a variable setting (eg: master output gain).

Note that variables are always in the range 0.0 to 1.0 which corresponds to MIDI Values 0 to 127. There it not always a direct relationship between the variable value and its meaning. For example although master gain is controllable using a value from 0 to 1, the actual gain applied varies from muted to about +7dB, with 0.5 being 0dB.

Think of the variable value being the position on the slider as opposed to an actual value.

Control Parameters

Some controls have an additional parameter that determines what it affects. Controls with parameters are indicated in the list of controls by type description of <type>+param (eg: command+param).

To get a list of parameters for a control, use the invoke list command specifying the object and control name as arguments. Eg:

```
>> invoke list subSessions subsessionN
L:#0,"#1: sub1"
L:#1,"#2: sub2"
L:#2,"#3: n/a"
L:#3,"#4: n/a"
L:#4,"#5: n/a"
L:#5,"#6: n/a"
(etc... omitted for brevity)
```

To invoke a control that has a parameter, include the parameter number as an argument after the control name. Eg:

```
>> invoke subSessions subsessionN 1
N:subsession loaded "sub2"
R:OK
>>
```

Invoke Examples

The following examples show how to use the invoke command. The list commands are included for reference only and are not required before the invocation.

Example, invoking a command:

```
>> invoke list transport
L:command,"transport","transportPlay","Play"
L:command,"transport","transportPause","Pause"
```

```
L:command,"transport","transportStop","Stop"
L:latch,"transport","transportPlayPause","Play/Pause"
L:latch,"transport","transportPlayStop","Play/Stop"
L:command,"transport","transportRewind","Rewind"
L:command,"transport","transportRestartEngine","Restart Audio Engine"
R:OK
>> invoke transport transportPlay
N:transport Playing
R:OK
>>
```

Example, invoking a latch:

```
>> invoke transport transportPlayPause on
N:transport Playing
R:OK
>> invoke transport transportPlayPause off
N:transport Paused
R:OK
>>
```

Example, invoking a variable

```
>> invoke list masterLevels
L:variable,"masterLevels","masterInputGain","Master Input Gain"
L:latch,"masterLevels","masterInputEnabled","Master Input Enabled"
L:variable,"masterLevels","masterOutputGain","Master Output Gain"
L:latch,"masterLevels","masterOutputEnabled","Master Output Enabled"
R:OK
>> invoke masterLevels masterInputGain 0.5
R:OK
>>
```