grass valley

AMP
ADVANCED MEDIA PROTOCOL

AMP Sockets Development

VERSION 1.1

AUG 2007

the most watched worldwide

A THOMSON BRAND

Revision Status

Version Number	Revision Date	Description
1.0	01/18/2006	Initial Release
1.1	08/24/2008	Revised Material

1 Sockets Introduction

Ethernet access is possible using Sockets on the Profile, Turbo, M-Series, and K2 platforms. Interfacing with Sockets is very similar to interfacing with DCOM.

Transitioning from DCOM to Sockets is very simple. No changes are required to the specific packets that contain AMP commands and responses. The transition only requires that your send and receive functionality be changed from DCOM send/receive units to Sockets send/receive units.

This document describes how to use Sockets interface to send and receive commands on the Profile, Turbo, M-Series, and K2 platforms.

2 IANA Port Number Requirement

AMP assigned **IANA** port number is **3811** for both TCP and UDP.

3 TCP Messages

The TCP client waits after each command sent to receive a response back from the server. A new line character (i.e. '\n') should be the last character of a TCP message. It acts as a separator between two TCP messages.

There are two kinds of TCP Messages:

- Control Messages
- Protocol Messages

3.1 Control Messages

Control Messages (or commands) are used to perform the following functions:

- Create new AMP servers for a specific channel
- Connect to previously existing AMP servers for a specific channel
- Stop the server and re-acquire the channel.

Note: The Message Terminator ('\n') should always be appended to commands sent and should not be counted in the Total number of chars.

There are several types of Control commands, as follows:

3.1.1 CRAT

This command is used to create a client connection to an AMP server. The **CRAT** command format is specified in the following description.

CRAT<Total no. of characters to follow><MODE>[<Channel-Number Length><CHANNEL NUMBER>]

On receiving this request from the client, The AMP Ethernet server creates the AMP server for that particular channel, provided it is not already created.

Mode character describes the mode in which AMP server is going to be created. The following table shows various modes.

The Channel Number Length and the Channel Number are sent only for mode 2.

Mode	Mode Description
2	TCP Channel Connection
4	TCP Channel-less Connection

CRAT TABLE—C++: The following table illustrates in C++ how to create a socket connection to a server. This table also illustrates how to create an AMP connection within that server to a channel or to the channel-less mode. This table may span more than one page. Please refer to the AMPSocketC project at www.gvgDevelopers.com.

Create Socket Connection and send CRAT Command (Creates a client connection on an AMP Server)	**Description**
`_sockIF = new CWinSocket();`	Create a TCP Socket.
`_sockIF->ConnectToHost(serverName, 3811)`	Connect to host described by endpoint.

`switch(mode)`
`{`
`  case TCP_CHN:`
`  sprintf(sendBuffer,`
`  "CRAT%.4d%.d%.2d%s\n",`
`  strlen(channelName) + 3,`
`  mode,`
`  strlen(channelName),`
`  channelName);`
`  break;`
`  default: //TCP_CHL`
`  sprintf(sendBuffer,"CRAT0001%d\n",`
`  TCP_CHL);`
`}`
	This sample program uses either channel-less TCP (MODE 1) or Channel TCP (Mode 2). For the channel TCP format, this example uses the “CRAT<Total Number Of Character To Follow><MODE>[<Channel Number Length><Channel Number>]” format

Create Socket Connection and send CRAT Command (Creates a client connection on an AMP Server)	Description
`unsigned int numSent = _sockIF->Send(sendBuffer, numToSend);`	Send the command.

`char *rcvdBCDFormat = (char *) malloc(numToRecv * 2);`
`unsigned int numRcvd =`
`_sockIF->Receive(rcvdBCDFormat, numToRecv * 2);`
 	
`if (numRcvd == (numToRecv * 2))`
`{`
`BCDToHex(numRcvd, rcvdBCDFormat, recvBuffer);`
`}`
 	
`SAFEFREE(rcvdBCDFormat);`
	Receive the response.

CRAT TABLE—C#: The following table illustrates in C# how to create a socket connection to a server. This table also illustrates how to create an AMP connection within that server to a channel or to the channel-less mode. This table may span more than one page. Please refer to the AMPSocket project at www.gvgDevelopers.com.

Create Socket Connection and send CRAT Command (Creates a client connection on an AMP Server)	Description
`addrHost = Dns.Resolve(hostName).AddressList[0];`	Resolve host's IP Address. This may take some time, depending on how many devices are on the network. If you already know the host IP address, skip this step.
`IPHost = new IPEndPoint(addrHost, 3811);`	Create an endpoint. NOTE: AMP uses **port 3811**.

`socket = new Socket(AddressFamily.InterNetwork,`
`SocketType.Stream,`
`ProtocolType.Tcp );`
	Create a TCP Socket.

`socket.Connect(IPHost);`	Connect to host described by endpoint.

`if (channel == null || channel.Length == 0)`
`szData = "CRAT00014\n";`
`else`
`szData=string.Format("CRAT{0:0000}{1}{2:00}{3}\n",`
`channel.Length+3, 2, channel.Length, channel );`
	This sample program uses either channel-less TCP (MODE 1) or Channel TCP (Mode 2). For the channel TCP format, this example uses the "CRAT<Total Number Of Character To Follow><MODE>[<Channel Number Length><Channel Number>]" format

Create Socket Connection and send CRAT Command (Creates a client connection on an AMP Server)	Description
byData = System.Text.Encoding.ASCII.GetBytes(szData);	Convert the command to ASCII format.
socket.Send(byData);	Send the command.
int iRx = socket.Receive(buffer);	Receive the response.

3.1.2 STOP

This command is sent to disconnect a client from the AMP server. If no other clients are connected to this AMP Server, and if the AMP server is not busy performing operations, then the AMP server is also shutdown. The format of the **STOP** command is as follows:

STOP<Total no of characters>

STOP Total No. of chars '\n'

4char 4char 1 char

If the requested client is not the last client connected to the server, the Message-Processor only closes the connection between the server and the specified client.

In a **STOP** request packet, the Total Number of chars will always be zero, since it does not take any parameters.

STOP COMMAND: The following table illustrates in C++ how to close the AMP connection and the socket connection. Please refer to the AMPSocketC project at www.gvgDevelopers.com.

Send STOP Command	Description
char sendBuffer[12] = "STOP0000\n";	Build the command using the "STOP<Total # Characters To Follow>\n" format. Total characters is always equal to '0000'
unsigned int numSent = _sockIF->Send(sendBuffer, numToSend);	Send the command.
char *rcvdBCDFormat = (char *) malloc(numToRecv * 2); unsigned int numRcvd =	An ACK (10.01) response confirms the AMP Server is

Send STOP Command	Description

`_sockIF->Receive(rcvdBCDFormat, numToRecv * 2);`
`if (numRcvd == (numToRecv * 2))`
`{`
`BCDToHex(numRcvd, rcvdBCDFormat, recvBuffer);`
`}`
`SAFEFREE(rcvdBCDFormat);`
	closed.

`_sockIF->DisconnectFromHost();`	Close the socket connection.

STOP COMMAND: The following table illustrates in C# how to close the AMP connection and the socket connection. Please refer to the AMPSocket project at www.gvgDevelopers.com.

Send STOP Command	Description
`szData = "STOP0000\n";`	Build the command using the "STOP<Total # Characters To Follow>\n" format. Total characters is always equal to '0000'
`byData = System.Text.Encoding.ASCII.GetBytes(szData);`	Convert to ASCII format.
`socket.Send(byData);`	Send the command.
`int iRx = socket.Receive(buffer);`	An ACK (10.01) response confirms the AMP Server is closed.
`socket.Close();` `socket = null;`	Close the socket connection.

3.2 Protocol Messages

Protocol Messages (or commands) are used to send AMP commands. Please see the *AMP Secifiication* for information pertaining to specific commands.

3.2.1 Send Protocol Commands

Protocol commands contain the AMP command byte-stream. The format of this command is as follows:

CMDS < Total no. Of characters to follow><Actual AMP Command Byte Stream>

CMDS Total no. Of chars Actual AMP Command Byte Stream '\n'

4char 4char 1 char

Note: The AMP command byte stream is in ASCII for compatibility with the rest of the command characters

Reply:

As per the AMP specification. The reply stream is in ASCII to maintain consistency with the data sent.

SEND COMMAND: The following table illustrates in C++ how to send a command and wait for a response from the server. Please refer to the AMPSocketC project at www.gvgDevelopers.com.

Send Command	Description

`sprintf(sendBuffer, "CMDS%.4d", cmdLen * 2);`
`memcpy(sendBuffer + MSG_HDR_LEN + 4, BCDCmd,`
`cmdLen * 2);`
`strcat(sendBuffer, "\n");`
	Build the command using the "CMDS<Total # Characters To Follow><AMP Command>\n" format.

`unsigned int numSent = _sockIF->Send(sendBuffer,` `numToSend);`	Send the command.

`char *rcvdBCDFormat = (char *) malloc(numToRecv * 2);`
`unsigned int numRcvd =`
`_sockIF->Receive(rcvdBCDFormat, numToRecv * 2);`
`if (numRcvd == (numToRecv * 2))`
`{`
`BCDToHex(numRcvd, rcvdBCDFormat, recvBuffer);`
`}`
`SAFEFREE(rcvdBCDFormat);`
	Receive the Response.

SEND COMMAND: The following table illustrates in C# how to send a command and wait for a response from the server. Please refer to the AMPSocket project at www.gvgDevelopers.com.

Send Command	Description
`szData = string.Format( "CMDS{0:0000}{1}\n",` `command.Length, command );`	Build the command using the "CMDS<Total # Characters To Follow><AMP Command>\n" format.

Send Command	Description
byData = System.Text.Encoding.ASCII.GetBytes(szData);	Convert to ASCII format.
socket.Send(byData);	Send the command.
int iRx = socket.Receive(buffer);	Receive the Response.