Protocol Specifications

Server/Host/Client Protocols

This document details the protocols that the i-scream server uses to talk to hosts, which are remote applications feeding data in, and remote clients, which are reading the data as it flows through the system.

Revision History

24/03/01	Initial creation – converted from an online format.				
		Committed by:	ajm4	Verified by:	pjm2
				Date:	27/03/01
		Committed by:		Verified by:	
				Date:	
		Committed by:		Verified by:	
				Date:	
		Committed by:		Verified by:	
				Date:	
		Committed by:		Verified by:	
				Date:	

Introduction

This document details the various custom protocols that are in use throughout the i-scream system. These protocols are used to negotiate with the various components in the i-scream system and obtain configuration and status information. These protocols are not used to transmit i-scream data, but are often to negotiate a data link. The actual i-scream data packets that are sent by the various i-scream components contain XML formatted data. For information on the format of this data see the XML via UDP specification document which is also available online at:

http://www.i-scream.org.uk/cgiin/docs.cgi?doc=specification/xml_via_udp.txt

Conventions

All protocols in this document assume a valid connection of the appropriate type has been made, and that data streams are already available. All strings should, and will be, terminated with a \n (newline character) to indicate the end of the message. All messages are sent as ASCII Strings.

Fixed protocol messages will appear in [] brackets, if there is a variety of options they will be separated with the | character. For example:

[OK|ERROR]

Indicates that the message "OK" OR the message "ERROR" will be sent depending on the result of the last action.

Data messages will appear in {} brackets, where the name in the brackets indicates the type (as in kind, not data type) of data that will be returned. For example:

{lastmodified}

Indicates that data for use as 'last modified' will be returned.

If [ERROR] is sent back at any time, this indicates that all communication is over. EXCEPT where otherwise specified!

Host Connection Protocol

The initial connection of a host to the i-scream server is through the FilterManager. A Host gets its configuration and then gets assigned a Filter to connect to and start sending data.

The port number of the FilterManager is fully configurable, however the default at this time is 4567.

Host	**(direction)**	**Server**	**Comment**
[STARTCONFIG]	---------->		Requests to start receiving config information
	<----------	[OK|ERROR]	If the server ok's the request or not
[LASTMODIFIED]	---------->		Asks when the config was last modified (used when checking if the config has changed)
	<----------	[{lastmodified}|ERROR]	Returns a long int time since epoch eg, 123456789 To indicate what format this is in, I quote from the Java 1.3 JDK API "measured in milliseconds since the epoch"
[FILELIST]	---------->		Asks the server for the

			list of files that were used to build the config (used when checking if the config has changed)
	<----------	[{filelist}|ERROR]	Returns a semi-colon separated list of filenames eg,a.conf;b.conf;c.conf
[FQDN]	---------->		Asks the server to send the FQDN (fully qualified domain name) of the connecting host.
	<----------	[{fqdn}|ERROR]	Returns the FQDN of the connected host, or error if it can't be looked up.
	* LOOP START *		This loop reads configuration properties from the config
[{property}]	---------->		Sends the name of a property to be retrieved from the config file eg, UDPUpdateTime
	<----------	[{value}|ERROR]	Returns the value of the requested config property eg, 120 If it is unable to find the requested property it returns ERROR to indicate that fact, but communication still proceeds.
	* LOOP UNTIL *		The loop continues until the host sends the following message
[ENDCONFIG]	---------->		Indicates that the host requires no more config
	* LOOP END *		Communication continues
	<----------	[OK]	Indicates that the server is ready to proceed
[FILTER]	---------->		Asks the server to send it the host information of a filter that it should connect to
	<----------	[{filter}|ERROR]	Returns a semi-colon separated list of FQDN hostname, UDP port number and TCP port number that a configured Filter is running on and assigned to the calling

			host eg, raptor.ukc.ac.uk; 1234;5678 or ERROR if none of configured Filter's could be found.
[END]	---------->		Indicates to the server that the host has finished an will disconnect
	<----------	[OK|ERROR]	Indicates that the server is either ok or that it thought there was an error

Host Heartbeat Protocol

When a host is configured after it has connected it should obtain a property TCPUpdateTime. This indicates how often a host should send a "Heartbeat", which is a pro-active connection to the servers Filter a host is using to indicate that it is still alive. This "Heartbeat" also allows a host to see its configuration has changed and act accordingly. In a well-written host this should just be a case of dropping out of a loop and heading back to the start (connecting to the filter manager).

Host	**(direction)**	**Server**	**Comment**
[HEARTBEAT]	---------->		Starts the heartbeat protocol off
	<----------	[OK|ERROR]	Indicates that the server is ok or not
[CONFIG]	---------->		Indicates that the host wants to check its config
	<----------	[OK|ERROR]	Indicates that the server is ok or not
[{filelist}]	---------->		Send a semi-colon separated list of filenames eg, a.conf;b.conf;c.conf This should be identical to the one received in the connection protocol
	<----------	[OK]	Indicates that the server is ok
[{lastmodified}]	---------->		Returns a long int time since epoch eg, 123456789 This should be identical to the one received in the connection protocol To indicate what format this is in, I quote from the Java 1.3 JDK API "measured in milliseconds since the epoch"
	<----------	[OK|ERROR]	The server then checks all the files in

			the file list to see if they have been modified more recently than the lastmodified value. If they HAVE that indicates that the configuration has changed and the host should re-configure, thus it sends ERROR. If the files remain unchanged the server will return OK to indicate the host should continue as before.
[ENDHEARTBEAT]	---------->		Indicates to the server that the host has finished an will disconnect
	<----------	[OK|ERROR]	Indicates that the server is either ok or that it thought there was an error

Host Data Protocol

The UDP data packets that are sent by the host contain simply XML data. For information on the format of this data see the XML via UDP specification document which is also available online at: http://www.i-scream.org.uk/cgi-bin/docs.cgi?doc=specification/xml_via_udp.txt

Client Control Protocol

The client control protocol channel is opened by the client and allows a variety of actions to be carried out by the client at anytime. Unlike previous protocols, this is NOT sequential, all of the requests can be carried out in any order. All client protocols are backwards compatible, and the version in use is indicated by the protocol identifier.

There are three sections to this protocol.

1) Initialise (sent only at start)
2) Send command(s) - unlimited number in any order
3) Disconnect (sent only at end)

If at anytime the client sends something the server does not understand, an [ERROR] will be sent.

All are indicated with <> brackets.

v1.0

Protocol identifier: "PROTOCOL 1.0" (without quotes)
Supported commands:

STARTCONFIG
STARTDATA
STOPDATA

Client	**(direction)**	**Server**	**Comment**
<initialise>			
	<----------	[{protocol}]	The server sends the protocol identifier
[{name}]	---------->		The client sends its "name". This name is used to identify the type of client to the system and obtain its config eg, Conient
	<----------	[OK]	Indicates the server is ok to proceed
<command #1> (allows client to obtain its configuration)			
[STARTCONFIG]	---------->		Tell the server we want to start this command
	<----------	[OK]	Indicates the server is ok to proceed
	* LOOP START *		This loop reads configuration properties from the config
[{config};{property}]	---------->		Sends the name of a configuration and property to be retrieved from the config file Clients can obtain host information eg, Host.UDPUpdateTime Otherwise it must prefix requests with "Client." All other requests will be ignored as if it was unable to retrieve the property
	<----------	[{value}|ERROR]	Returns the value of the requested config property eg, 120 If it is unable to find the requested property it returns ERROR to indicate that fact
	* LOOP UNTIL *		The loop continues until the client sends the following message
[ENDCONFIG]	---------->		Indicates that the client requires no more config
	* LOOP END *		Communication continues
	<----------	[OK]	Indicates that the

			server is ready to proceed
<command #2> (tells the server to start the data link)			
[STARTDATA]	---------->		Tell the server we want to start this command
	<----------	[{portnumber}|ERROR]	The server then sets itself listening for a connection on its data link socket for this client. It returns the port no. that it is listening on eg, 12367 If the data link is already started the server will return ERROR
	<----------	[OK]	Indicates the client has successfully connected to the data socket.
<command #3> (tells the server to stop the data link)			
[STOPDATA]	---------->		Tell the server we want to start this command. The server then shuts down the data link to the client
	<----------	[OK|ERROR]	Returns OK is the server is ready to proceed, or ERROR if the data link was not open in the first place.
<disconnect>			
[DISCONNECT]	---------->		Tells the server the client wants to close the control link
	<----------	[OK]	The last word from the server, it will disappear after this, and close the data link if required

v1.1

Protocol identifier: "PROTOCOL 1.1" (without quotes)
Supported commands:
SETHOSTLIST

Client	(direction)	Server	Comment
<command #4> (indicates to the server which hosts the client wants data from)			
[SETHOSTLIST]	---------->		Tell the server we want to start this command.
	<----------	[OK|ERROR]	The server will return an ERROR if the data link is open, as a host list must ONLY be set if the data link is closed If the server is ok to proceed with the command it says

			[OK]
[{hostlist}]	---------->		This is a semi-colon separated list of FQDN hostnames that the client wishes to receive data from eg, raptor.ukc.ac.uk;killigrew.ukc.ac.uk; If the list is sent blank (or if no list is set at all) then data from ALL hosts will be sent to the client (this is the default if no SETHOSTLIST is not run by the client
	<----------	[OK]	Indicates the server is ok to proceed and the host list has been accepted

Client Data Protocol

Once the data link has been opened by the control link, the server will send XML formatted data packets separated by the \n (newline) character. For information on the format of this data see the XML via UDP specification document which is also available online at: http://www.i-scream.org.uk/cgi-bin/docs.cgi?doc=specification/xml_via_udp.txt