Telos Systems Corp. Maciej Szlapka

LIVEWIRE ROUTING PROTOCOL

PROTOCOL VERSION 1.2 rev 1

Document version 2.2

Release history:

2003.12.11	Protocol document version: 2.0.1 Initial release
2004.08.18	Added silence and clipping detector (LVL command)
2004.12.08	Documented IP command
2005.09.27	Fixed typo in GPO command. The name of the attribute is DURATION, not TIME
2005.10.11	Protocol version: 2.0.2 Removed dual-stream source configuration tags Added LABL source attribute (10-character label) Added GPI/GPO custom message support Added NCHN attribute to sources and destinations (surround audio support)
2005.10.19	Suggested URL encoding of custom GPIO commands
2009.09.28	Protocol version: 2.0.3 Allows text object IDs
2010.02.08	Protocol version: 2.0.4 Removed FUNC option from CFG GPO command (never implemented or used)
2016.08.18	Added STAT operation - protocol version 1.2
2016.08.19	Added Grammar section

1 Introduction

Livewire Routing protocol provides control for node devices that are part of distributed audio routing system. This protocol specify how access configuration data of particular device individually addressed. Every node in Livewire system has unique network address, so it can be individually accessed.

Livewire systems also use higher levels of abstraction to address audio sources with a channel number. This protocol does not provide any support for that. It uniquely provides functions to configure individual devices.

Main function of routing protocols is changing audio routes. This goal is achieved by configuring receive address at the audio destination ports. Similar approach is applied to GPIO. General purpose inputs are routed to outputs, by configuring the GPIO destinations.

This protocol also allows access to user-friendly names associated with ports. It also allows access to basic network related settings of the device.

Livewire Routing Protocol is a native interface for Axia Livewire Audio and General Purpose I/O Nodes.

Livewire Protocol uses very standard and well-known protocols and interfaces designed for Internet. This assures easy implementation on any hardware platform, and operating system. The design does not require any libraries for processing exchanged messages. It is ASCII based, and all commands are short and descriptive. Using ASCII format allows communication without having need to specify special data types of data.

The protocol support object oriented architecture of distributed system that supports automatic discovery of device capabilities. This way having a system built from different devices can be easily built and will not cause any configuration problems. The protocol can be only partially implementd, when certain functions are not applicable to certain devices do not have to be implemented. The common and required part is reduced to a single command that allows connected client to learn about capabilities of device it has connected to. For example: a GPIO device will not specify any audio ports, and has not implemented any of commands related to audio routing.

Router Management Systems designed for controlling integrated audio routers can implement Livewire Routing Protocol or use a software gateway between this protocol and their native protocol. First solution is preferable, as it enables Router Management Systems to fully control every device and it is free from potential problems with the gateway reliability. To avoid multiple TCP/IP connections from Router Management System, the other solution may be employed. There must be gateway/translator software developed for every protocol that has to be supported. Another option is to develop new protocol that would encapsulate Livewire Routing Protocol messages and a gateway that redirects them to individual audio terminal units. It has to be determined whether such protocol can be widely accepted by Router Management System developers.

2 Communication (Physical/Network/Transport layers)

LWRP messages are transported using TCP/IP reliable network connections. This interface is supported by every popular operating system and is available in all commonly used development environment.

Technically, every device provides a TCP/IP server bound to TCP port number 93. TCP provides reliable point-to-point communication channel, so no CRC is needed to ensure delivery of messages. TCP also assures that message duplicated by the network are rejected. In this case identifier is needed in data information exchange to identify duplicated messages in application.

Client writes text commands to the socket. It also receives configuration parameters change indication and error messages from device. Client should not assume that server indications will be send back immediately after command was issued. Multiple clients can control a single device and can determine its state. Indications are not responses to commands.

Protocol implements pure push message exchange model. Client should not assume any state of a device unless it get an indication of current state of configuration parameter. This is very important detail of the protocol, as it allows multiple clients to control one device and mainain a coherent state.

3 Response time

Network communication enables very fast message exchange. It can deliver response time to commands within about 10ms.

4 Messages (Application layer)

Livewire Routing Protocol is a text-based protocol using simple ASCII text messages. Every command has to be terminated with <LF> or <CR>+<LF> characters. Messages sent by the client to the server are named commands. Messages generated by the server (device) are called indications.

See the "Grammar" section for formal syntax definition.

General commands are required in any implementation of Livewire Routing Protocol. Devices that do not have audio ports are not supporting audio routing commands. Devices that do not have general purpose input output ports are not supporting General Purpose I/O commands.

Following sections define commands and their syntax. Parameters included in pair of square brackets are optional.

There are following top-level elements defined:

Command	Description
VER	Request version and general information This is the only command that is required to be supported by every device that implements Livewire Routing Protocol.
LOGIN	Provides security access to commands changing state of the device
PASSWD	Change access password
SET	Set global parameters
SRC	Configure audio source
DST	Configure audio destination
MTR	Request audio level meter data
LVL	Silence and clipping detector
ADD GPI [ports]	Enable state change indications on GPIO input ports
DEL GPI [ports]	Disable state change indications on GPIO input ports
ADD GPO [ports]	Enable state change indications on GPIO output ports
DEL GPO [ports]	Disable state change indications on GPIO output ports
GPO [port] [state]	Change state of a GPIO output port
CFG GPO	Configure GPO port
IP	Allows basic IP configuration
STAT	Get statictics and runtime information associated wit an object

Livewire Routing Protocol Indications Summary:

Indication	Description
VER	Version and general information
SET parameters	Global parameters
SRC port parameters	Configure audio source
DST port parameters	Configure audio destination
MTR	Request audio level meter data
GPO port state	State of a GPIO output port changed
GPI port state	State of a GPIO input port changed
CFG GPO	Configuration of GPO port
ERROR number message	Error message

4.1 General commands

4.1.1 Login

client: LOGIN [<password>]

Client must provide a valid password using the LOGIN command, to execute any commands changing configuration of the unit. LOGIN with no parameters sets read-only permissions again. If password is not set all access rights are given to the client.

Local connections using 127.0.0.1 do not require password to have all access rights.

4.1.2 Change password

client: PASSWD [<old password>] [<new password>]

4.1.3 Version information

client: VER

server: VER <name1>:<value1>[[<BLANK><name2>:<value2>]]

Following names are defined:

LWRP	Routing protocol version: 1.4.2	
DEVN	Device name: LiveIO, GPIO	
SYSV	System version: 1.1.1	
NSRC	Number of sources. Decimal number (ex: 8). Stereo ports are assumed. To specify different number of channels use N/c syntax. Eg. NSRC:8/1 means 8 mono sources.	If missing, assume 0
NDST	Number of outputs: Decimal number (ex: 8)	If missing, assume 0
NGPI	Number of General Purpose inputs	If missing, assume 0
NGPO	Number of General Purpose outputs	If missing, assume 0

Client should ignore parameters it does not process.

4.1.4 IP: Set IP parameters

client: IP [address <d.d.d.d>] [netmask <d.d.d.d>] [gateway <d.d.d.d>] [hostname <name>]

To query current IP configuration send IP command with no parameters.

Hostname must comply with the DNS host name standard. In particualr it can not include white spaces, and has to be 12 charactes long or less.

4.1.5 SET: Set global parameters

client: SET <name1>:<value1>[[<BLANK><name2>:<value2>]]

Allows setting device specific global parameters. No tags are defined by this protocol.

Client should ignore parameters it does not process.

4.1.6 SAVE: Saving configuration

client: SAVE

4.1.7 BEGIN ... END: Batch commands

client: BEGIN <LF> command1 <LF> .... commandN <LF> END

Those keywords identify a group of commands or indications that are sent as a group. They allow optimized processing. For example device may parse multiple DST commands and update actual audio routing information once, at the end, instead of doing it after every single DST command. Grouping also allows simultanous route changes on multiple ports.

END is optional when EXE follows programs definition.

4.1.8 EXE: Execute batch created with BEGINEND

client: EXE

4.2 Audio routing commands

4.2.1 SRC: Configure source

client: SRC <port> <name1>:<value1>[[<BLANK><name2>:<value2>]]

<port>: output port ID

The field can be a physical port number or the name of a port as defined by the specific hardware being controlled. Devices providing fixed set of ports will usually use numbers 1-NSRC. But when number of ports can be changing or new types can be introduced by software revisions, text names provide a more flexible way of addressing.

Following attribute names are defined:

PSNM	Primary Source Name. ASCII text, up to 16 characters
RTPE	RTP Stream Enable: 0 or 1
RTPA	RTP Stream destination address in following format: 0—<ip number in a.b.c.d format>[:<udp port number>] Default udp port number is 5004 assigned to RTP standards
PSNM	Primary source name: 16-character ASCI string (printable characters only)
INGN	Input gain in 10dB units (range depends on a device)
LABL	Source label. 10-character ASCII string for the control surface LCD display
NCHN	Number of audio channels provided by this source (assume 2 if not specified)

Client should ignore parameters it does not process.

Examples:

SRC 1 PSNM:"CD 1" RTPE:1 RTPA:"239.192.0.1" INGN:-100

SRC ENC#1 RTPA:"239.192.0.1"

4.2.2 DST: Configure destination

client: DST <port> [ADDR:<url>] [NAME:<string>]

<port>: output port ID

<url>: <d.d.d.d IP address>[:UDP port] [<source name>]

See note about port ID in SRC command description.

Following names are defined:

NAME	Destination name. ASCII text, up to 16 characters
ADDR	RTP Stream destination address in following format: 0|<ip number in a.b.c.d format>[:<udp port number>] Default udp port number is 5004 assigned to RTP standards
NCHN	Number of audio channels the destination is configured for

Examples:

DST 1 ADDR:"239.192.0.1 <CYGNUS 1@CYGNUS>" NAME:"Omnia"

DST DEC#1 ADDR:"239.192.0. 1"

4.2.3 Change indication

If another client successfully executes OUT, an asynchronous notification is sent to all other clients.

server: DST | SRC | SET

or

server: BEGIN <LF> (multiple indications) <LF> END <LF>

As a result of execution of an EXE command, multiple indications can be sent with BEGIN and END keywords marking beginning and end of the sequence.

See DST, SRC and SET command for syntax of indications.

4.2.4 Status query

client: DST

server: BEGIN <LF> <multiple DST indications> <LF> END <LF>

or

client: SRC

server: BEGIN <LF> <multiple SRC indications> <LF> END <LF>

or

client: SET

server: SET indication will be generated as requested.

4.2.5 Meter data query

client: MTR [ICH|OCH]

server: BEGIN <LF> <multiple MTR indications> <LF> END <LF>

For each input (ICH) channel and output (OCH) channel peek and RMS values are returned. These values are tagged with PEEK and RMS respectively. They are followed by :, then left channel level, : and right channel level. Levels are communicated in 1/10dBFS units (-1000 corresponds to 100dBFS). Numbers are transmitted in ASCII format.

If ICH or OCH is specified in MTR command, only data for input or output channels, respectively, will be returned.

Live I/O terminals return level values in range from 100dBFS to 0dBFS.

Meter data query does not require login.

Example server response:
BEGIN
MTR ICH 1 PEEK:-1000:-1000 RMS:-1000:-1000
MTR ICH 2 PEEK:-1000:-1000 RMS:-1000:-1000
MTR ICH 3 PEEK:-1000:-1000 RMS:-1000:-1000
MTR ICH 4 PEEK:-1000:-1000 RMS:-1000:-1000

MTR ICH 5 PEEK:-1000:-1000 RMS:-1000:-1000
MTR ICH 6 PEEK:-1000:-1000 RMS:-1000:-1000
MTR ICH 7 PEEK:-1000:-1000 RMS:-1000:-1000
MTR ICH 8 PEEK:-1000:-1000 RMS:-1000:-1000
MTR OCH 1 PEEK:-1000:-1000 RMS:-1000:-1000
MTR OCH 2 PEEK:-1000:-1000 RMS:-1000:-1000
MTR OCH 3 PEEK:-1000:-1000 RMS:-1000:-1000
MTR OCH 4 PEEK:-1000:-1000 RMS:-1000:-1000
MTR OCH 5 PEEK:-1000:-1000 RMS:-1000:-1000
MTR OCH 6 PEEK:-1000:-1000 RMS:-1000:-1000
MTR OCH 7 PEEK:-1000:-1000 RMS:-1000:-1000
MTR OCH 8 PEEK:-1000:-1000 RMS:-1000:-1000
END

4.2.6 Silence and clipping detector

client: LVL ICH|OCH <port> CLIP.LEVEL:<level> CLIP.TIME:<time> LOW.LEVEL:<level> LOW.TIME:<time>

server: LVL ICH|OCH <port>.<speaker channel> LOW|NO-LOW|CLIP|NO-CLIP

For each input (ICH) channel and output (OCH) channel peek and silence detection can be enabled. Levels are expressed in 1/10dBFS units (-1000 corresponds to 100dBFS); time constants in milliseconds. <speaker channel> can be L (left) or R (Right). Numbers are transmitted in ASCII format.

Example:

Client command:

LVL OCH 1 CLIP.LEVEL:-100 CLIP.TIME:3000 LOW.LEVEL:-600 LOW.TIME:3000

This command enables silence and clipping level detection on output port number 1. Clipping treshold is set to 10dBFs. Exceeding clip level will cause a message to be sent to the client immediately. There is a separate detector for left and right channel so two messages are possible:
LVL ICH 1.L CLIP
LVL ICH 1.R CLIP

After specified CLIP.TIME of 3000ms and no clipping detected, client will be notified with:
LVL ICH 1.L NO-CLIP
LVL ICH 1.R NO-CLIP

At the same time audio level should fall below 60dBFs threshold for 3000ms (LOW.TIME), following messages would be generated:
LVL OCH 1.L LOW
LVL OCH 1.R LOW

When audio is present again:
LVL OCH 1.L NO-LOW
LVL OCH 1.R NO-LOW

Live I/O terminals return level values in range from 100dBFS to 0dBFS.

4.3 General Purpose Input Output

Every Livewire GPIO device has a number of input and output ports. Each port is a group of pins electric circuits. A client application does not have to know the exact model of GPIO device. The interface and communication protocol remains the same for all of them. Number of ports can be discovered with VER command described in previous section.

For wiring and pin assignment details, please refer to documentation of particular device.

In GPIO related commands, state of ports is encoded using 5-character strings. Each character corresponds to one pin. There are 4 different characters in use representing state or state transition. Lower case letters are sent when state is steady, while capital letters indicate state transition:

l	steady low
h	steady high
L	high to low transition
H	low to high transition

4.3.1 ADD: Adds ports to indication list

client: ADD GPI [<port>]

or

client: ADD GPO [<port>]

Client sends this command to enable port state change indications.

Optional <port> argument defines port of interest. If <port> is not specified, client will receive all ports change indications.

Device immediately sends current state indication for all listed ports.

ADD GPO can be used for debugging and monitoring purposes. Implementation should not be using indications as an acknowledgement of the action. TCP/IP guarantees message delivery of messages. Device will send an error indication in case of syntax error.

Example: Client will receive indications of status change in all ports.

ADD GPI

Example response of 8-port GPIO device:
BEGIN
GPI 1 lhhhh
GPI 2 hhhhh
GPI 3 hhhhh
GPI 4 hhhhh
GPI 5 hhhhh
GPI 6 hhhhh
GPI 7 hhhhh
GPI 8 hhhhh
END

4.3.2 DEL: Delete ports from indication list

client: DEL GPI [<list>]

or

client: DEL GPO [<list>]

Client sends this command to disable port state change indications on ports specified by list. Syntax is the same as ADD command.

4.3.3 GPO: Output port change command

client: GPO <port> state [DURATION:<delta time>]

<port>: See note about port ID in SRC command description.

State is a string containing h, l and x characters one per pin:

H/h change state to high L/l change state to low X/x no change

<delta time> indicates time when the change should happen. It is expressed in milliseconds since the last change of the output port.

Example below shows how to generate a 250-millisecond pulse in the 5th pin of 2nd port of the GPIO. This double command sequence instructs the unit to change state of the pin to high and then to low after the given time interval:
GPO 2 xxxxH
GPO 2 xxxxL DURATION:250

4.3.4 GPI/GPO: State change indications

server: GPI <port> state

or

server: GPO <port> state

<port>: See note about port ID in SRC command description.

example:

GPI 7 hLhhh

Server sends this message to all clients that have added indications for port n, every time ports state changes.

State is a string containing h, l, H, L characters one per pin:

H/L transition from low to high/high to low l/h steady low/high state

4.3.5 GPI/GPO: Custom messages

server: GPI <port> CMD:<custom command>

or

server: GPO <port> CMD:<custom command>

<port>: See note about port ID in SRC command description.

example:

GPI 7 CMD:BREAK 100 DURATION 3000

CMD

ASCII string up to 128 characters. Only printable characters allowed.

Recommended URL encoding of custom commands.

Only alphanumerics [0-9a-zA-Z], the special characters "$-_.+!*'()," [not including the quotes - ed], and reserved characters used for their reserved purposes may be used unencoded within a command.

URL encoding of a character consists of a "%" symbol, followed by the two-digit hexadecimal representation (case-insensitive) of the ISO-Latin code point for the character. Example:

* Percent character (%) = decimal code point 37 in the ISO-Latin set.

* 37 decimal = 25 in hexadecimal

* The URL encoded representation will be "%25"

This message allows distribution of user custom commands using GPIO routing capabilities of Livewire.

All clients subscribed to indications on port n get a copy of the command.

4.3.6 CFG GPO: Set/query output ports attributes

client: CFG GPO <port> attr1 attr2 .

<port>: See note about port ID in SRC command description.

Following attributes are defined:

SRCA

Address of another GPIO to enable input output following function. The address is in url format: <ip address>[:<udp port number>]/<GPI port number>

4.3.7 STAT: Get statictics and runtime information associated with an object

client: STAT <object identifier>

server: STAT <object identifier> <available information> <LF> END <LF>

or

server: ERROR 1005 "STAT <object identifier>" not supported <LF> END <LF>

STAT operation is only available in devices which indicate LWRP Protocol version 1.2 or higher (see VER).

Either a valid response or error message containing queried object ID is required. It allows clients to discover supported features by the device.

Example 1:
STAT ICH
BEGIN
STAT ICH 1 AESSYNC:ERR
STAT ICH 2 AESSYNC:ERR
END

Example 2:
STAT OCH
ERROR 1005 "STAT OCH" not supported

4.4 Error messages

4.4.1 ERROR indication

server: ERROR <number> <message>

Server sends error messages to signal command parsing errors or system failures. They are sent to client that issued bad command. Client software should log these error messages for further troubleshooting.

Codes and messages:

1000	bad command (or not authorized access)
1001	syntax error
1002	bad port number
1003	bad program number
1004	invalid password

5 Grammar

5.1 EBNF

```
LWRPCommand         ::= ProtocolCommand | ObjectAccessCommand | ErrorStatus

ProtocolCommand     ::= (VER | BEGIN | EXE | END | LOGIN | SAVE | PASSWD | ADD | DEL)
                        (Object)?

ObjectAccessCommand ::= (Operation)? Object (Index)? (NameAndValue | ValueUnquoted)*
Operation           ::= CFG | MTR | STAT
Object              ::= (CharCommon)*
Index               ::= (digit)*
NameAndValue        ::= [A-Z]*":" Value
Value               ::= (ValueQuoted | ValueUnquoted)
ValueUnquoted       ::= (CharUnquoted)*
ValueQuoted         ::= CharQuote (CharQuoted)* CharQuote
CharQuoted          ::= CharUnquoted | CharSpace
CharUnquoted        ::= EscapedChar | CharCommon
EscapedChar         ::= CharEscape (CharSpace | CharQuote | CharEscape | CharCommon)
CharCommon          ::= any character excluding
                        (CharSpace | double quote | backward slash)
CharEscape          ::= escape character
CharQuote           ::= double quote
CharSpace           ::= space

ErrorStatus         ::= ERROR ErrorNumber (ErrorDescription)?
ErrorNumber         ::= [0-9]*
ErrorDescription    ::= (Char)*
```

5.2 Regex

```
NameAndValue  := [A-Z_]+:("(\\.|[^"\\])*"|(\\.|[^\s"\\])*)

Value         := ("(\\.|[^"\\])*"|(\\.|[^\s"\\])*)

ValueUnquoted := (\\.|[^\s"\\])*
ValueQuoted   := "(\\.|[^"\\])*"

CharQuoted    := (\\.|[^"\\])
CharUnquoted  := (\\.|[^\s"\\])

EscapedChar   := \\.

CharCommon    := [^\s"\\]
CharEscape    := \\
CharQuote     := "
CharSpace     := \s
```

It is recommended to use ValueQuoted for maximum compatibility.

Escaped Space is not recommended. New protocol implementations must not produce values containing Escaped Space. Values containing spaces must be quoted.

Examples:

```
PSNM:"SRC 1"
PSNM:"STUDIO \"1\""
RTPE:1
RTPA:"239.192.39.137"
```

Examples of not recommended values:

```
PSNM:STUDIO\ \"1\"
```

6 References

RFC 791 Internet Protocol

RFC 793 Transmission Control Protocol