Wheatstone WheatNet IP
Blade Automation Protocol

Date: November 10, 2017

1 Introduction & Revision History

The Blade automation control protocol for the WheatNet IP system hardware and PC driver Blades is described in this document. These Blades all use a common automation control protocol over an Ethernet TCP/IP connection.

This document describes version 1.13 of the Blade automation control protocol.

Date	Author	Protocol Revision	Description of Change
August 12, 2008	GAJ	1.0	• Created
October 21, 2008	GAJ	1.0	• Added utility mixer commands
October 31, 2008	GAJ	1.1	• Added utility mixer enable query.
November 12, 2008	GAJ	1.0	• Added description of escape characters in strings.
November 12, 2008	GAJ	1.2	• Increased maximum TCP connections from 10 to 20. • Added signal subscription messages. • Added software LIO messages and signal name subscription messages.
November 18, 2008	GAJ	1.2	• Added source and destination DEF parameters to determine when a signal is deleted. • Added SYS IFID command.
February 17, 2009	GAJ	1.3	• Added DEF parameter to SALVO, SALVOSUB and SALVOEVENT messages. • Changed wording in appendix B that implies the fixed position of soft LIOs in the PC driver since we did not implement them that way.
February 17, 2009	GAJ	1.4	• Added UMIX URAMPA, URAMPB, DRAMPA & DRAMPB commands.
April 14, 2009	GAJ	1.4	• Modified model type return strings in SYS query.
July 21, 2009	GAJ	1.5	• Added BLADE command target section.
August 25, 2010	GAJ	1.6	• Fixed broken BLADE LIST query in software. The query always returned invalid channel with version 1.5 of the protocol.
Sept. 14, 2010	GAJ	1.6	• Added UMIX subscriptions and UMIX SIGID query.
May 13, 2011	GAJ	1.6	• Modified section on TCP timeouts and heartbeats to stress the need for applications to implement a periodic heartbeat message.
December 01, 2011	GAJ	1.7	• Added utility mixer INCA, INCB and MINC commands for incrementing (decrementing) utility mixer faders by a specified amount without having to query the fader level before hand.
June 26, 2012	GAJ	1.8	• Added SYS BLID to determine the ID of the Blade you are connected to.
October 09, 2012	GAJ	1.9	• Added utility mixer commands DUCKLVL, DUCKA, DUCKB, BALA & BALB. Added utility mixer subscriptions DUCKA and DUCKB.

August 01, 2013	DAG	1.10	• Added LIO, SURFSPARE, and TIME commands. Added subscriptions for each, as well as subscription for SALVO FIRE events.
July 14, 2014	DAG GAJ	1.10	• Added STRING, STRINGSUB, added GAIN and GAINT to SRC and DST. Added Appendix C – Signal Gain Type Enumerations
Sept. 10, 2014	GAJ	1.10	• Added UMIX subscriptions for fader levels (FDRA, FDRB, MFDR).
Sept. 22, 2015	GAJ	1.12	• Added SYS SUBRATE to modify the subscription message rate for slower devices.
February 10, 2016	DAG	1.13	• Added NAME and LOCATION commands on DST and SRC for setting signal names.
May 26, 2017	DAG	1.14	• Added MIC PPWR to control microphone blades' phantom power setting.
November 10, 2017	JWP	1.15	• Added GUIPORT attribute on BLADE query. Some cleanup of typos.

1 TCP Connection

Each Blade supports up to 20 simultaneous TCP connections (10 on a PC driver) from client PCs or other network devices. The Blade acts as a TCP server for the connection. The PC acts as a TCP client for the connection. The Blade listens for TCP connections on port 55776. All remote computers will make a TCP connection to the Blades with this one TCP port number.

1.1 TCP Connection Timeout

The Blade will timeout and close a TCP connection if it does not have any activity for more than 120 seconds. Therefore, in order to maintain your TCP connection your application **MUST PERIODICALLY SEND A HEARTBEAT** message of some sort.

You could periodically request the version number of the console, which would result in a response being sent to your application or you can send an empty message, if you want to periodically send heartbeats to keep the connection alive without generating a response from the console. An empty message would be of the form:

```
<>
```

You should attempt to send the heartbeat more frequently than the 120 second timeout period. A 30 second heartbeat message would be a good target rate to implement. Each of the up to 20 connections are independent, each requires its own heartbeat activity.

If your application fails to send periodic heartbeats, the Blade will close the TCP interface. There will be a small window of time while your application is attempting to re-establish connection to the Blade during which any commands which your application attempts to send to the Blade will fail. If you are subscribing to events in the Blade you will trigger a burst of subscription event messages from the blade after the reconnect which may have detrimental performance effects on the Blade and/or your application.

2 Message Format

All automation control messages are made up of upper case ASCII character strings. The general format of a message is start character (**<**), message target, query or command character (**?** Or **|**), optional parameters, end character (**>**).

2.1 Message Delimiters

Each message is delimited by a less then (**<**) character at the beginning and a greater than (**>**) character at the end. Any characters following the greater than character at the end of one message and preceding the next less than character starting the next message are ignored. Therefore carriage returns and line feeds and zero terminators between messages will be silently ignored.

Example:

```
<SYS?>
<SALVO:4?NAME>
<SALVO:4|NAME:Salvo 1>
```

The Blade will respond to all properly formatted messages that are sent to it with an answer, an acknowledgment, or an error, depending on the message that was sent.

2.2 Message Target

The message target follows the start character. The target identifies the target of the command. A generic target supported by all Blades is the **SYS** target. The **SYS** target may be queried to find the Blade type and revision, it may also be queried to find the version of this protocol that the mixer supports.

Example:

```
<SYS?>
<SYS?MODEL>
<SYS?VERSION>
```

Other targets are **DST** (destination) signals, **SRC** (source) signals, **SALVO**, etc.
For numbered targets such as **DST** signals the message target also contains the signal number.

Example:

```
<DST:00400001?>
<SRC:00400002?>
<SALVO:3?>
```

2.3 Message Type

The message type follows the message target. The message type may be either a query or a command. Query messages are identified by a question mark character (**?**) and command messages are identified by a pipe character (**|**).

2.4 Optional Parameters

Parameters follow the message type character. When the message contains multiple parameters, the parameters are delimited by comma characters. When a parameter has a value associated, the value will be delimited from the parameter name with a colon character.

Parameters are optional for query messages. For example if you send a **SYS** query with no parameters, you will receive a response string with all of the system parameters and their values in the response.

Example:

```
Query:    <SYS?>
Response: <SYS|NAME:Blade03,BLID:3,MODEL:IP88a,VERSION:1.6.5,AUTO:1.8,UPTIME:0010D23H59M10S,
          TEMP:28.50,UMIX:2,SLIO:64,IFID:192.168.87.103,SUBRATE:10.100>
```

As an alternative you can specify a system parameter on a **SYS** query and the mixer will respond with just the value of that single parameter.

Example:

```
Query:    <SYS?MODEL>
Response: <SYS|MODEL:IP-88a>
Query:    <SYS?VERSION>
Response: <SYS|VERSION:1.0.0>
```

Commands must have at least one parameter but may optionally have multiple parameters. For example you could change a destination's connected source, then lock the destination with two commands each with a single parameter:

Example:

```
Command:  <DST:00400001|SRC:00800003>
Response: <OK>
Command:  <DST:00400001|LOCKED:1>
Response: <OK>
```

Or you could change the connected source on a destination signal lock that destination on with one command with two parameters:

Example:

```
Command:  <DST:00400001|SRC:00800003,LOCKED:1>
Response: <OK>
```

3 Reply Format

Reply messages from the Blade follow the same format as command messages. The Blade will insert a carriage (Hex 0D) and line feed character (Hex 0A) after every reply. Since these characters are outside the message delimiters, the characters should be ignored by a message parser. The characters are included to make string parsing easier for some computer languages.

As you've seen from the examples in the previous section the Blade will respond to query messages with a string that is formatted in the same manner as a command string. The Blade will respond to commands with an acknowledgment string "**<OK>**".

If a query or command message is not properly formatted with a start and end character the Blade will ignore the message. If the message is properly formatted, but contains instructions that are out of bounds or not supported by the particular type of Blade an error message will be returned.

Error String	Description
<NAK|Invalid Message Format>	The message format was invalid.
<NAK|Unsupported Request>	An unsupported command was specified.
<NAK|Invalid Channel>	The target channel number is invalid.
<NAK|Invalid Parameter ID>	The parameter is invalid.
<NAK|Invalid Parameter Value>	The parameter value is invalid.
<NAK|Not All Commands Processed>	Some of the command parameters may have been processed but some had errors. This error will only occur when multiple parameters are specified within a command.

4 Special Characters

You may have noted from the previous sections that there are several characters which have special meaning in the text protocol of the automation interface. These characters are: less than "**<**", greater than "**>**", pipe "**|**", question mark "**?**", comma "**,**", and colon "**:**". And not obvious from the preceding text the forward slash "**/**" is also reserved. These characters are not found in any commands (as of this writing), but may be embedded with a return string. For example if a signal name has one of these characters in it it would make parsing a response string very difficult.

For example let's assume we have a source signal for Joe's microphone named "<mic>Joe". If the Blade returned that string in a response it would make parsing the response string indeterminate.

Example:

```
Query:    <SRC:00400001?NAME>
Response: <SRC:00400001|NAME:<mic>Joe>       Wrong can not parse this !!!
```

To deal with this the Blade will use escape sequences on string parameters in responses. We use the forward slash character "**/**" as our escape character. Whenever you encounter a forward slash character in a response it indicates that the next character following it should be taken literally and not as a token in the response string parser.

So in the previous example the Blade would return:

```
Query:    <SRC:00400001?NAME>
Response: <SRC:00400001|NAME:/<mic/>Joe>
```

Here are some more examples assuming these signals:

```
Source 00400001 = "mic|Joe"
Source 00400002 = "mic:Bob"
Source 00400003 = "A/B/C"
Source 00400004 = "Jeff???"

Response: <SRC:00400001|NAME:mic/|Joe>
Response: <SRC:00400002|NAME:mic/:Bob>
Response: <SRC:00400003|NAME:A//B//C>
Response: <SRC:00400004|NAME:Jeff/?/?/?>
```

5 Parameter Messages

This section breaks down the available message targets and their parameters. Each message target and parameter has a description of the parameter value and a listing of which mixers support the target and parameter.

In the following sections each command has a table showing a quick breakdown of the command components. The table also indicates the initial protocol revision that

5.1 SYS Parameters

System message targets are used to query and set system parameters.

5.1.1 SYS MODEL

Target: SYS
Parameter: MODEL
Channel: N/A
Value: Model number string, see below
Query: Yes
Command: No
Protocol Rev: 1.0

The SYS MODEL parameter may be queried by the automation application on the PC. The reply parameter value will specify the type of Blade (“PC Drvr”, “IP-88a”, “IP-88d”, “IP-88ad” or “IP-88e”, etc.).

Example:

```
Query:    <SYS?MODEL>
Response: <SYS|MODEL:IP-88d>
```

5.1.2 SYS BLID

Target: SYS
Parameter: BLID
Channel: N/A
Value: Blade ID
Query: Yes
Command: No
Protocol Rev: 1.8

The SYS BLID parameter may be queried by the automation application on the PC. The reply parameter value will specify the type of Blade ID.

Example:

```
Query:    <SYS?BLID>
Response: <SYS|BLID:3>
```

5.1.3 SYS VERSION

Target: SYS
Parameter: VERSION
Channel: N/A
Value: Software version string, see below
Query: Yes
Command: No
Protocol Rev: 1.0

The SYS VERSION parameter may be queried by the automation application on the PC. The reply parameter value will specify the Blade software revision number.

Example:

```
Query:    <SYS?VERSION>
Response: <SYS|VERSION:1.0.0>
```

5.1.4 SYS AUTO

Target: SYS
Parameter: AUTO
Channel: N/A
Value: Version number, see below
Query: Yes
Command: No
Protocol Rev: 1.0

The SYS AUTO parameter may be queried by the automation application on the PC. The reply parameter value will specify the version of the automation controller protocol supported.

Example:

```
Query:    <SYS?AUTO>
Response: <SYS|AUTO:1.0>
```

The version number is a two part number delimited by periods.

The first number is the major revision number. This number will only change when a part of the message formats have changed. If you have written an application according to a major revision that does not match the reported major revision of the surface, you will encounter incompatibility problems in some messages. Major version will likely never change (they have not change in 5+ years on any Wheatstone surface).

The second number is the minor revision number. This number will only change when a message is not backward compatible. The minor revision number will typically increment when a new message target and/or parameter is added to the protocol. If you have written an application according to a minor revision that is greater than the reported minor revision of the surface, you may encounter incompatibility problems in some messages. If you have written an application according to a minor revision that is equal to or less then the minor revision of the surface then you will not encounter any compatibility problems.

See Revision History section to find the version of the protocol defined in this document.

5.1.5 SYS IFID

Target: SYS
Parameter: IFID
Channel: N/A
Value: Interface client ID string
Query: Yes
Command: Yes
Protocol Rev: 1.2

The SYS IFID parameter may be queried or set by the automation application on the PC. This parameter is a convenience parameter for helping to identify clients connected to the Blade's automation interface. Your application should write an identification string into this parameter when it firsts connects to the Blade. The parameter has no real effect except as a debug aide. When the TCP connection is first established the IFID string is initialized with your client's IP address.

Example:

```
Command:  <SYS?IFID>
Response: <SYS|IFID:192.168.0.87.10>       Initial value has your IP address.

Command:  <SYS|IFID:Acme Automation PC A>  Changed to a custom string.
Response: <OK>

Command:  <SYS?IFID>                       Later query will return your custom ID string.
Response: <SYS|IFID:Acme Automation PC A>
```

5.1.6 SYS UPTIME

Target: SYS
Parameter: UPTIME
Channel: N/A
Value: Uptime string, see below
Query: Yes
Command: No
Protocol Rev: 1.0

The SYS UPTIME parameter may be queried by the automation application on the PC.

Example:
```
Command:  <SYS?UPTIME>
Response: <SYS|UPTIME:0010D23H59M10S>
```

5.1.7 SYS TEMP

Target:	SYS
Parameter:	TEMP
Channel:	N/A
Value:	Degrees C
Query:	Yes
Command:	No
Protocol Rev:	1.0

The SYS TEMP parameter may be queried by the automation application on the PC to read the internal temperature of the Blade.

Example:
```
Query:    <SYS?TEMP>
Response: <SYS|TEMP:28.50>
```

5.1.8 SYS UMIX

Target:	SYS
Parameter:	UMIX
Channel:	N/A
Value:	0-2
Query:	Yes
Command:	No
Protocol Rev:	1.0

The SYS UMIX parameter may be queried by the automation application on the PC to determine the number of utility mixers on the Blade. The result from this query specifies the number of utility mixers that are available in the hardware configuration for that Blade, it does not indicate the number of virtual mixers enabled. As an example if the optional utility mixer feature was purchased for an IP-88a Blade and both of the utility mixers are enabled in the Blade, the result will be 2. If one or both of the utility mixers are then disabled using the WheatNet IP Navigator GUI, the result from this query will still be 2. You can determine which of the two utility mixers are enabled using the <UMIX:1.0?ENABLED> and <UMIX:2.0?ENABLED> queries.

If a Blade was not setup up at the factory with any optional utility mixers this query will return a 0.

Example:
```
Query:    <SYS?UMIX>
Response: <SYS|UMIX:0>    No utility mixers available
Response: <SYS|UMIX:2>    Two utility mixers available in the hardware
```

5.1.9 SYS SLIO

Target:	SYS
Parameter:	SLIO
Channel:	N/A
Value:	Number of soft LIOs on the Blade.
Query:	Yes
Command:	No
Protocol Rev:	1.2

The SYS SLIO parameter may be queried by the automation application on the PC to determine the number of software LIO pins on the Blade.

The result from this query specifies the number of soft LIO pins that are available in the hardware configuration for that Blade, it does not indicate the number of soft LIO pins bound to signals. Nor is there any means via the automation protocol to determine which soft LIO pins are bound to a signal in the Blade or what their LIO function or direction is. It is up to the system designer to make that determination via the WheatNet IP Navigator GUI.

If a Blade was not setup up at the factory with any software LIO pins this query will return a 0.

See Also: Appendix B which describes the WheatNet IP Blade software LIO pin design.

Example:

```
Query:    <SYS?SLIO>
Response: <SYS|SLIO:0>    No software LIO pins available
Response: <SYS|SLIO:192>  One hundred ninety two software LIO pins available in the hardware
```

5.1.10 SYS LIO

Target: SYS
Parameter: LIO
Channel: N/A
Value: Number of physical LIO pins on the Blade.
Query: Yes
Command: No
Protocol Rev: 1.1

The SYS LIO parameter may be queried by the automation application on the PC to determine the number of LIO pins on the Blade.

The result from this query specifies the number of LIO pins that are available in the hardware configuration for that Blade, it does not indicate the number of LIO pins bound to signals. Nor is there any means via the automation protocol to determine which LIO pins are bound to a signal in the Blade or what their LIO function or direction is. It is up to the system designer to make that determination via the WheatNet IP Navigator GUI.

If a Blade was not setup up at the factory with any LIO pins this query will return a 0.

Example:

```
Query:    <SYS?LIO>
Response: <SYS|LIO:0>    No LIO pins available
Response: <SYS|LIO:12>   LIO pins available in the hardware on card 0 (Circuits 0-11)
```

5.1.11 SYS STRING

Target: SYS
Parameter: STRING
Channel: N/A
Value: Number of String variables available on the Blade.
Query: Yes
Command: No
Protocol Rev: 1.10 (some as the protocol revision in the software was not updated to 1.11 as intended)

The SYS STRING parameter may be queried by the automation application on the PC to determine the number of string variables on the Blade.

Example:

```
Query:    <SYS?STRING>
Response: <SYS|STRING:0>    No string variables available
Response: <SYS|STRING:128>  One hundred twenty eight string variables available (channels 1-128)
```

5.1.12 SYS SUBRATE

Target: SYS
Parameter: SUBRATE
Channel: N/A
Value: Synchronous subscription event message rate token bucket parameters in the format "capacity.fill rate".
Query: Yes
Command: Yes
Protocol Rev: 1.12

The SYS SUBRATE parameter may be queried by the automation application on the PC to determine the synchronous subscription event message rate for the interface. The SYS SUBRATE parameter may be set by the automation application to modify the default subscription event message rate for the interface. The default SUBRATE value is "10.100" this default value will be reset each time the TCP connection is established to the Blade for that ACI connection.

Access to this parameter allows you to slow down (or speed up) the rate at which your application will receive subscription events in immediate response to creating a subscription. When you create a subscription, you will receive a burst of subscription events to synchronize your application with what you are subscribing to. These events may come very fast and for a long amount of time especially in the case of signal subscriptions using wildcards. The default message rate is about 100 messages per second with the default parameter of "10.100". To reduce the rate to 50 per second you might set the parameter to "10.50". And to increase the rate to 200 per second you might change the parameter to "10.200".

There are two numbers in the parameter which are separated by a period. This is not a decimal point in a floating point number, it is a separator between two integers. The first integer is a "capacity" value and the second is a "fill rate". The Blade uses a token bucket algorithm to rate limit the asynchronous subscriptions messages. We will not go into great detail on the token bucket algorithm in this document further reading can be found here: https://en.wikipedia.org/wiki/Token_bucket. For most purposes you could leave the first number (capacity) at 10 and adjust the second parameter (fill rate) to specify the number of messages sent per second.

The capacity parameter will be limited to a range of 1 to 500. The fill rate parameter will be limited to a range of 1 to 1000.

Example:

```
Query:    <SYS?SUBRATE>
Response: <SYS|SUBRATE:10.100>  Token bucket capacity = 10 and fill rate = 100 msgs per second.
Command:  <SYS|SUBRATE:10.50>   Slow down token bucket capacity to 10 and fill rate to 50.
Query:    <SYS?SUBRATE>
Response: <SYS|SUBRATE:10.50>   Token bucket capacity = 10 and fill rate = 50 msgs per second.
```

5.2 BLADE Parameters

Blade targets are used to query basic information about the blades that make up your system.

5.2.1 BLADE LIST

Target:	BLADE:#
Parameter:	NAME
Channel:	0
Value:	List of blade IDs separated by semicolons.
Query:	Yes
Command:	No
Protocol Rev:	1.6

The BLADE LIST parameter may be queried by the automation application on the PC. The parameter value contains a string of blade ID numbers separated by semicolons.

Note: This command was in protocol rev 1.5 but a software bug always returned an invalid channel error.

Example:

```
Query:    <BLADE:0?LIST>
Response: <BLADE:0?LIST:1;2;3;4;5;6;7;8;9;10>  Ten blades numbers 1 through 10

Query:    <BLADE:0?>                           The query can be done with a wildcard query also.
Response: <BLADE:0?LIST:1;2;3;4;5;6;7;8;9;10>  Ten blades numbers 1 through 10
```

5.2.2 BLADE NAME

Target:	BLADE:#
Parameter:	NAME
Channel:	Blade ID (1 - 512)
Value:	Blade Name
Query:	Yes
Command:	No
Protocol Rev:	1.5

The BLADE NAME parameter may be queried by the automation application on the PC. The parameter value indicates the name of the Blade specified in the query. If the specified Blade does not exist or is off line an invalid channel NAK will be returned.

Example:
```
Query:    <BLADE:1?NAME>
Response: <BLADE:1?NAME:BladeOne>
```

5.2.3 BLADE IPADDR

Target: BLADE:#
Parameter: IPADDR
Channel: Blade ID (1 - 512)
Value: Blade IP address
Query: Yes
Command: No
Protocol Rev: 1.5

The BLADE IPADDR parameter may be queried by the automation application on the PC. The parameter value indicates the IP address of the Blade specified in the query. If the specified Blade does not exist or is off line an invalid channel NAK will be returned.

Example:
```
Query:    <BLADE:1?IPADDR>
Response: <BLADE:1?IPADDR:192.168.87.101>
```

5.2.4 BLADE MODEL

Target: BLADE:#
Parameter: MODEL
Channel: Blade ID (1 - 512)
Value: Blade Model type
Query: Yes
Command: No
Protocol Rev: 1.5

The BLADE MODEL parameter may be queried by the automation application on the PC. The reply parameter value will specify the type of Blade ("PC Drvr", "IP-88a", "IP-88d", "IP-88ad" or "IP-88e", etc).

Example:
```
Query:    <BLADE:1?MODEL>
Response: <BLADE:1?MODEL:IP-88ad>
```

5.2.5 BLADE GUIPORT

Target: BLADE:#
Parameter: GUIPORT
Channel: Blade ID (1 - 512)
Value: Port to be used by GUI
Query: Yes
Command: No
Protocol Rev: 1.15

The BLADE GUIPORT parameter may be queried by the automation application on the PC. The reply parameter value will specify the port to be used for connection by the GUI.

Example:
```
Query:    <BLADE:1?GUIPORT>
Response: <BLADE:1?GUIPORT:55001>
```

5.3 DST Parameters

Destination signal targets are used to query or set destination signal parameters and change connections or locks. When an DST target is specified a signal number must be a part of the message target.

5.3.1 DST SRC

Target: DST:#
Parameter: SRC
Channel: Destination signal number
Value: Source signal number
Query: Yes
Command: Yes
Protocol Rev: 1.0

The DST SRC parameter may be queried or set by the automation application on the PC. The parameter value indicates the connection state of the specified destination signal. A source value of '0000FFFF' indicates no connection.

Example:

```
Query:    <DST:00400001?SRC>
Response: <DST:00400001|SRC:00800002>   Destination 00400001 is connected to source 00800002
Command:  <DST:00400001|SRC:00800004>   Connect destination 00400001 to source 00800004
Response: <OK>
Command:  <DST:00400001|SRC:0000FFFF>   Disconnect destination 00400001
Response: <OK>
```

5.3.2 DST LOCKED

Target: DST:#
Parameter: LOCKED
Channel: Destination signal number
Value: 0 - unlocked, 1 = locked
Query: Yes
Command: Yes
Protocol Rev: 1.0

The DST LOCKED parameter may be queried or set by the automation application on the PC. The parameter value indicates the lock state of the destination signal.

Example:

```
Query:    <DST:00400001?LOCKED>
Response: <DST:00400001|LOCKED:0>    Destination 00400001 is not locked
Command:  <DST:00400001|LOCKED:1>    Lock destination 00400001
Response: <OK>
```

5.3.3 DST NAME

Target: DST:#
Parameter: NAME
Channel: Destination signal number
Value: Destination signal name string
Query: Yes
Command: No
Protocol Rev: 1.0

The DST NAME parameter may be queried by the automation application on the PC. The parameter value indicates the name of the specified destination signal.

Example:

```
Query:    <DST:00400001?NAME>
Response: <DST:00400001|NAME:JoesHdpn>
```

5.3.4 DST LOCATION

Target: DST:#
Parameter: LOCATION
Channel: Destination signal number
Value: Destination signal location string
Query: Yes
Command: No
Protocol Rev: 1.0

The DST LOCATION parameter may be queried by the automation application on the PC. The parameter value indicates the name of the specified destination signal.

Example:

```
Query:    <DST:00400001?LOCATION>
Response: <DST:00400001|LOCATION:STUDIO B>
```

5.3.5 DST DEF

Target: DST:#
Parameter: DEF
Channel: Destination signal number
Value: 1 if the specified destination signal is a valid defined signal, 0 if the destination is undefined.
Query: Yes
Command: No
Protocol Rev: 1.2

The DST DEF parameter may be queried by the automation application on the PC. The parameter value indicates whether the specified destination signal is defined or not.

Example:

```
Query:    <DST:00400001?DEF>
Response: <DST:00400001|DEF:0>   Destination 00400001 is not defined.
Response: <DST:00400001|DEF:1>   Destination 00400001 is defined.
```

5.3.6 DST GAINT

Target: DST:#
Parameter: GAINT
Channel: Source signal number
Value: An enumeration for the input output gain type of the specified signal.
Query: Yes
Command: No
Protocol Rev: 1.10 (some as the protocol revision in the software was not updated to 1.11 as intended)

The DST GAINT parameter may be queried by the automation application on the PC. The parameter value indicates the type of input output gain control which the signal provides. See "Appendix C – Signal Gain Type Enumerations" for a table of gain type values.

Example:

```
Query:    <DST:00400001?GAINT>
Response: <DST:00400001|GAINT:1>   The signal has -18.0 to +18.0 adjustable gain range.
```

5.3.7 DST GAIN

Target: DST:#
Parameter: GAIN
Channel: Source signal number
Value: 1 if the specified source signal is a valid defined signal, 0 if the destination is undefined.
Query: Yes
Command: No
Protocol Rev: 1.10 (some as the protocol revision in the software was not updated to 1.11 as intended)

The DST GAIN parameter may be queried by the automation application on the PC. The parameter value indicates the gain value currently active on the destination in tenths of dB (i.e. 180 = +18.0dB). If the GAINT values is 0, then the signal does not have adjustable gain.

Example:

```
Query:    <DST:00400001?GAIN>
Response: <DST:00400001|GAIN:100>   10.0 dB is the current gain setting
```

5.4 SRC Parameters

Source signal targets are used to query or set source signal parameters. When an SRC target is specified a signal number must be a part of the message target.

5.4.1 SRC NAME

Target: SRC:#
Parameter: NAME
Channel: Source signal number
Value: Source signal name string
Query: Yes
Command: No
Protocol Rev: 1.0

The SRC NAME parameter may be queried by the automation application on the PC. The parameter value indicates the name of the specified source signal.

Example:

```
Query:    <SRC:00400001?NAME>
Response: <SRC:00400001|NAME:Joes Mic>
```

5.4.2 SRC LOCATION

Target: SRC:#
Parameter: LOCATION
Channel: Source signal number
Value: Source signal location string
Query: Yes
Command: No
Protocol Rev: 1.0

The SRC LOCATION parameter may be queried by the automation application on the PC. The parameter value indicates the name of the specified source signal.

Example:

```
Query:    <SRC:00400001?LOCATION>
Response: <SRC:00400001|LOCATION:STUDIO B>
```

5.4.3 SRC DEF

Target: SRC:#
Parameter: DEF
Channel: Source signal number
Value: 1 if the specified source signal is a valid defined signal, 0 if the source is undefined.
Query: Yes
Command: No
Protocol Rev: 1.2

The SRC DEF parameter may be queried by the automation application on the PC. The parameter value indicates whether the specified source signal is defined or not.

Example:

```
Query:    <SRC:00400001?DEF>
Response: <SRC:00400001|DEF:0>   Source 00400001 is not defined.
Response: <SRC:00400001|DEF:1>   Source 00400001 is defined.
```

5.4.4 SRC GAINT

Target: SRC:#
Parameter: GAINT
Channel: Source signal number
Value: An enumeration for the input output gain type of the specified signal.
Query: Yes
Command: No
Protocol Rev: 1.10 (some as the protocol revision in the software was not updated to 1.11 as intended)

The SRC GAINT parameter may be queried by the automation application on the PC. The parameter value indicates the type of input output gain control which the signal provides. See "Appendix C – Signal Gain Type Enumerations" for a table of gain type values.

Example:
```
Query:    <SRC:00400001?GAINT>
Response: <SRC:00400001|GAINT:1>   The signal has -18.0 to +18.0 adjustable gain range.
```

5.4.5 SRC GAIN

Target: SRC:#
Parameter: GAIN
Channel: Source signal number
Value: 1 if the specified source signal is a valid defined signal, 0 if the source is undefined.
Query: Yes
Command: No
Protocol Rev: 1.10 (some as the protocol revision in the software was not updated to 1.11 as intended)

The SRC GAIN parameter may be queried by the automation application on the PC. The parameter value indicates the gain value currently active on the source in tenths of dB (i.e. 180 = +18.0dB). If the GAINT values is 0, then the signal does not have adjustable gain.

Example:
```
Query:    <SRC:00400001?GAIN>
Response: <SRC:00400001|GAIN:100>   10.0 dB is the current gain setting
```

5.5 SALVO Parameters

Salvo targets are used to query or set salvo parameters and to fire salvos. When an SALVO target is specified a salvo number must be a part of the message target.

5.5.1 SALVO NAME

Target: SALVO:#
Parameter: NAME
Channel: Salvo number (1 to 256)
Value: Salvo name string
Query: Yes
Command: No
Protocol Rev: 1.0

The SALVO NAME parameter may be queried by the automation application on the PC. The parameter value indicates the name of the specified salvo.

Example:
```
Query:    <SALVO:1?NAME>
Response: <SALVO:1|NAME:Salvo1>
```

5.5.2 SALVO DEF

Target: SALVO:#
Parameter: DEF
Channel: Salvo number (1 to 256)
Value: 1 if the specified salvo is a valid defined salvo, 0 if the salvo is undefined.
Query: Yes
Command: No
Protocol Rev: 1.3 (1.4 see note below)

The SALVO DEF parameter may be queried by the automation application on the PC. The parameter value indicates whether the specified salvo is defined or not.

Note: There is a software implementation error in protocol revision 1.3 Blades in that a query for an undefined salvo returns an invalid channel NAK rather than DEF:0. This was repaired in protocol revisions 1.4 and up. The error has no harmful effect other than the responses to salvo queries being different than stated here.

Example:

```
Query:    <SALVO:1?DEF>
Response: <SALVO:1|DEF:0>   Salvo 1 is not defined.
Response: <SALVO:1|DEF:1>   Salvo 1 is defined.
```

5.5.3 SALVO FIRE

Target: SALVO:#
Parameter: FIRE
Channel: Salvo number (1 to 256)
Value: 1
Query: No
Command: Yes
Protocol Rev: 1.0

The SALVO FIRE command may be sent by the automation application on the PC to fire the specified salvo.
Note: There must be a value of 1 passed in the salvo FIRE parameter.

Example This fires salvo #1:

```
Command:  <SALVO:1|FIRE:1>
Response: <OK>
```

Example This fires salvo #3:

```
Command:  <SALVO:3|FIRE:1>
Response: <OK>
```

5.6 UMIX Parameters

Utility mixer targets are used to query or set utility mixer parameters.

Note: When an UMIX target is specified the utility mixer number and the input or output channel must be a part of the message target. The message target is of the form #.#, where the first # indicates the utility mixer to access (1-2) and the second # indicate the input channel (1-8), or the output bus (A-B), or 0 in the special case of the mixer enabled query.

Note: If the specified utility mixer is not enabled in the Blade, then a <NAK|Invalid Channel> error will be returned.

5.6.1 UMIX ENABLED

Target: UMIX:#.#
Parameter: ENABLED
Channel: Mixer # . 0
Value: 1 = Enabled, 0 = Disabled
Query: Yes
Command: No
Protocol Rev: 1.1

The UMIX ENABLED parameter may be queried by the automation application on the PC. The parameter value indicates the enabled state of the specified utility mixer.

Blade with 2 utility mixers, the first is enabled the second is not:

```
Query:    <SYS?UMIX>
Response: <SYS|UMIX:2>          Two utility mixers available in the hardware
Query:    <UMIX:1.0?ENABLED>
Response: <UMIX:1.0|ENABLED:1>  First is enabled
Query:    <UMIX:2.0?ENABLED>
Response: <UMIX:2.0|ENABLED:0>  Second is not enabled
```

5.6.2 UMIX DUCKLVL

Target: UMIX:#.#
Parameter: DUCKLVL
Channel: Mixer # . 0
Value: -80.0 to 0.0 = dB value
Query: Yes
Command: Yes
Protocol Rev: 1.9

The UMIX DUCKLVL parameter may be queried or set by the automation application on the PC. The parameter value indicates the dB level which the input faders will be ducked when ducked via the UMIX DUCKA and DUCKB commands. One ducking level is set globally across the entire utility mixer. Each utility mixer in a Blade may have a unique ducking level.

Ducking Example:

```
Query:    <UMIX:1.0?DUCKLVL>        Read Utility Mixer 1 Ducking Level
Response: <UMIX:1.0|DUCKLVL:-12.0> It's at -12 dB (the default level)
Command:  <UMIX:1.0|DUCKLVL:-40.0> Set Ducking Level to -40 dB
Response: <OK>
Query:    <UMIX:1.3?DUCKA>          Read Utility Mixer 1 Input 3 A Ducking
Response: <UMIX:1.3|DUCKA:0>        It's at 0 = NOT ducked
Command:  <UMIX:1.3|DUCKA:1>        Duck Utility Mixer 1 Input 3 A (reduce level by 40 dB)
Response: <OK>
Command:  <UMIX:1.3|DUCKA:0>        Un-Duck Utility Mixer 1 Input 3 A (restore level)
Response: <OK>
```

5.6.3 UMIX SIGID

Target:	UMIX:#.#
Parameter:	SIGID
Channel:	Mixer # . Input Channel ~ or ~ Mixer # . Output Bus (see above)
Value:	Signal number
Query:	Yes
Command:	No
Protocol Rev:	1.6

The UMIX SIGID parameter may be queried by the automation application on the PC. The parameter value indicates the destination signal ID for input channels or the source signal ID for output channels. This query will return a signal ID regardless of whether the utility mixer is enabled or not. If the utility mixer is not enabled the signal ID will reflect an undefined signal but that ID will be valid if/when the corresponding utility mixer is enabled.

Input Fader Example:

```
Query:    <UMIX:1.1?SIGID>            Read Utility Mixer 1 Input 1 signal ID
Response: <UMIX:1.1|SIGID:00C00600>
Query:    <UMIX:1.2?SIGID>            Read Utility Mixer 1 Input 2 signal ID
Response: <UMIX:1.2|SIGID:00C00601>
Query:    <UMIX:1.8?SIGID>            Read Utility Mixer 1 Input 8 signal ID
Response: <UMIX:1.8|SIGID:00C00607>
```

Output Fader Example:

```
Query:    <UMIX:1.A?SIGID>            Read Utility Mixer 1 Output A signal ID
Response: <UMIX:1.A|SIGID:00C00600>
Query:    <UMIX:1.B?SIGID>            Read Utility Mixer 1 Output B signal ID
Response: <UMIX:1.B|SIGID:00C00601>
```

5.6.4 UMIX ON

Target:	UMIX:#.#
Parameter:	ON
Channel:	Mixer # . Input Channel ~ or ~ Mixer # . Output Bus (see above)
Value:	1 = ON, 0 = OFF
Query:	Yes
Command:	Yes
Protocol Rev:	1.0

The UMIX ON parameter may be queried or set by the automation application on the PC. The parameter value indicates the state of the specified channel's ON button.

Input Fader Example:

```
Query:    <UMIX:1.2?ON>      Read Utility Mixer 1 Input 2 on button state, it is off
Response: <UMIX:1.2|ON:0>
Command:  <UMIX:1.2|ON:1>    Turn Utility Mixer 1 Input 2 on
Response: <OK>
Query:    <UMIX:1.2?ON>      Read Utility Mixer 1 Input 2 on button state again, it is on
Response: <UMIX:1.2|ON:1>
```

Output Fader Example:

```
Query:    <UMIX:1.A?ON>      Read Utility Mixer 1 Output bus A on button state, it is off
Response: <UMIX:1.A|ON:0>
Command:  <UMIX:1.A|ON:1>    Turn Utility Mixer 1 Output bus A on
```

```
Response: <OK>
```

5.6.5 UMIX FDRA

Target:	UMIX:#.#
Parameter:	FDRA
Channel:	Mixer # . Input Channel (see above)
Value:	-80.0 to 12.0 = dB value
Query:	Yes
Command:	Yes
Protocol Rev:	1.0

The UMIX FDRA parameter may be queried or set by the automation application on the PC. The parameter value indicates the dB level of the specified input channel's "A" bus fader. A value outside the fader range will not generate an error, the Blade will bring the specified level into the valid range and use the modified value (i.e. specifying 18dB will result in 12dB or -100dB will result in -80dB).

Input Fader Example:

```
Query:    <UMIX:1.3?FDRA>          Read Utility Mixer 1 Input 3 A fader level
Response: <UMIX:1.3|FDRA:-12.0>    It's at -12 dB
Command:  <UMIX:1.3|FDRA:0>        Set Utility Mixer 1 Input 3 A fader level to 0dB
Response: <OK>
~ or ~
Command:  <UMIX:1.3|FDRA:0.0>      Set Utility Mixer 1 Input 3 A fader level to 0dB
Response: <OK>
```

5.6.6 UMIX FDRB

Target:	UMIX:#.#
Parameter:	FDRB
Channel:	Mixer # . Input Channel (see above)
Value:	-80.0 to 12.0 = dB value
Query:	Yes
Command:	Yes
Protocol Rev:	1.0

The UMIX FDRB parameter may be queried or set by the automation application on the PC. The parameter value indicates the dB level of the specified input channel's "B" bus fader. A value outside the fader range will not generate an error, the Blade will bring the specified level into the valid range and use the modified value (i.e. specifying 18dB will result in 12dB or -100dB will result in -80dB).

Input Fader Example:

```
Query:    <UMIX:1.3?FDRB>          Read Utility Mixer 1 Input 3 B fader level
Response: <UMIX:1.3|FDRB:-12.0>    It's at -12 dB
Command:  <UMIX:1.3|FDRB:0.0>      Set Utility Mixer 1 Input 3 B fader level to 0dB
Response: <OK>
```

5.6.7 UMIX MFDR

Target:	UMIX:#.#
Parameter:	MFDR
Channel:	Mixer # . Output Bus (see above)
Value:	-80.0 to 0.0 = dB value
Query:	Yes
Command:	Yes
Protocol Rev:	1.0

The UMIX MFDR parameter may be queried or set by the automation application on the PC. The parameter value indicates the dB level of the specified output bus master fader. A value outside the fader range will not generate an error, the Blade will bring the specified level into the valid range and use the modified value (i.e. specifying 6dB will result in 0dB or -100dB will result in -80dB).

Output Fader Example:

```
Query:    <UMIX:1.A?MFDR>          Read Utility Mixer 1 bus A master fader level
Response: <UMIX:1.A|MFDR:-12.0>    It's at -12 dB
Command:  <UMIX:1.A|MFDR:0.0>      Set Utility Mixer 1 bus A master fader level to 0dB
Response: <OK>
```

5.6.8 UMIX INCA

Target: UMIX:#.#
Parameter: INCA
Channel: Mixer # . Input Channel (see above)
Value: Positive or negative dB change value
Query: No
Command: Yes
Protocol Rev: 1.7

The UMIX INCA parameter may be set by the automation application on the PC. The parameter value indicates a change in the dB level of the specified input channel's "A" bus fader. A value outside the fader range will not generate an error, the Blade will bring the specified level into the valid range and use the modified value (i.e. if the fader is at 0dB and specifying +20dB will result in +12dB fader level since +12dB is the top of the range).

Input Fader Example:

```
Query:    <UMIX:1.3?FDRA>           Read Utility Mixer 1 Input 3 A fader level
Response: <UMIX:1.3|FDRA:-12.0>     It's at -12 dB
Command:  <UMIX:1.3|INCA:10.0>      Increment Utility Mixer 1 Input 3 A fader level by 10dB
Response: <OK>
Query:    <UMIX:1.3?FDRA>           Read Utility Mixer 1 Input 3 A fader level
Response: <UMIX:1.3|FDRA:-2.0>      Now it's at -2 dB
Command:  <UMIX:1.3|INCA:-10.0>     Decrement Utility Mixer 1 Input 3 A fader level by 10dB
Response: <OK>
Query:    <UMIX:1.3?FDRA>           Read Utility Mixer 1 Input 3 A fader level
Response: <UMIX:1.3|FDRA:-12.0>     Now it's back at -12 dB
```

5.6.9 UMIX INCB

Target: UMIX:#.#
Parameter: INCB
Channel: Mixer # . Input Channel (see above)
Value: Positive or negative dB change value
Query: No
Command: Yes
Protocol Rev: 1.7

The UMIX INCB parameter may be set by the automation application on the PC. The parameter value indicates a change in the dB level of the specified input channel's "B" bus fader. A value outside the fader range will not generate an error, the Blade will bring the specified level into the valid range and use the modified value (i.e. if the fader is at 0dB and specifying +20dB will result in +12dB fader level since +12dB is the top of the range).

Input Fader Example:

```
Query:    <UMIX:1.3?FDRB>           Read Utility Mixer 1 Input 3 B fader level
Response: <UMIX:1.3|FDRB:-12.0>     It's at -12 dB
Command:  <UMIX:1.3|INCB:10.0>      Increment Utility Mixer 1 Input 3 B fader level by 10dB
Response: <OK>
Query:    <UMIX:1.3?FDRB>           Read Utility Mixer 1 Input 3 B fader level
Response: <UMIX:1.3|FDRB:-2.0>      Now it's at -2 dB
Command:  <UMIX:1.3|INCB:-10.0>     Decrement Utility Mixer 1 Input 3 B fader level by 10dB
Response: <OK>
Query:    <UMIX:1.3?FDRB>           Read Utility Mixer 1 Input 3 B fader level
Response: <UMIX:1.3|FDRB:-12.0>     Now it's back at -12 dB
```

5.6.10 UMIX MINC

Target: UMIX:#.#
Parameter: MINC
Channel: Mixer # . Output Bus (see above)
Value: Positive or negative dB change value
Query: No
Command: Yes
Protocol Rev: 1.7

The UMIX MINC parameter may be set by the automation application on the PC. The parameter value indicates a change in the dB level of the specified output bus master fader. A value outside the fader range will not generate an error, the Blade will

bring the specified level into the valid range and use the modified value (i.e. if the fader is at -12dB and specifying +20dB will result in 0dB fader level since 0dB is the top of the range).

Output Fader Example:

```
Query:     <UMIX:1.A?MFDR>            Read Utility Mixer 1 bus A master fader level
Response: <UMIX:1.A|MFDR:-12.0>     It's at -12 dB
Command:  <UMIX:1.A|MINC:10.0>      Increment Utility Mixer 1 bus A master fader level by 10dB
Response: <OK>
Query:     <UMIX:1.A?MFDR>            Read Utility Mixer 1 bus A master fader level
Response: <UMIX:1.A|MFDR:-2.0>      Now it's at -2 dB
Command:  <UMIX:1.A|MINC:-10.0>     Decrement Utility Mixer 1 bus A master fader level by 10dB
Response: <OK>
Query:     <UMIX:1.A?MFDR>            Read Utility Mixer 1 bus A master fader level
Response: <UMIX:1.A|MFDR:-12.0>     Now it's back at -12 dB
```

5.6.11 UMIX URAMPA

Target:	UMIX:#.#
Parameter:	URAMPA
Channel:	Mixer # . Input Channel (see above)
Value:	0=Off, 1=Fast, 2=Medium, 3=Slow
Query:	Yes
Command:	Yes
Protocol Rev:	1.4

The UMIX URAMPA parameter may be queried or set by the automation application on the PC. The parameter value indicates the fade up ramp speed of the specified input channel's "A" bus fader.

Input Fader Up Ramp Speed Example:

```
Query:     <UMIX:1.3?URAMPA>          Read Utility Mixer 1 Input 3 A fader up ramp rate
Response: <UMIX:1.3|URAMPA:0>       It's set to "Off"
Command:  <UMIX:1.3|URAMPA:2>       Set Utility Mixer 1 Input 3 A fader up ramp speed to "Medium"
Response: <OK>
```

5.6.12 UMIX DRAMPA

Target:	UMIX:#.#
Parameter:	DRAMPA
Channel:	Mixer # . Input Channel (see above)
Value:	0=Off, 1=Fast, 2=Medium, 3=Slow
Query:	Yes
Command:	Yes
Protocol Rev:	1.4

The UMIX DRAMPA parameter may be queried or set by the automation application on the PC. The parameter value indicates the fade down ramp speed of the specified input channel's "A" bus fader.

Input Fader Down Ramp Speed Example:

```
Query:     <UMIX:1.3?DRAMPA>          Read Utility Mixer 1 Input 3 A fader up ramp rate
Response: <UMIX:1.3|DRAMPA:1>       It's set to "Fast"
Command:  <UMIX:1.3|DRAMPA:3>       Set Utility Mixer 1 Input 3 A fader up ramp speed to "Slow"
Response: <OK>
```

5.6.13 UMIX URAMPB

Target:	UMIX:#.#
Parameter:	URAMPB
Channel:	Mixer # . Input Channel (see above)
Value:	0=Off, 1=Fast, 2=Medium, 3=Slow
Query:	Yes
Command:	Yes
Protocol Rev:	1.4

The UMIX URAMPB parameter may be queried or set by the automation application on the PC. The parameter value indicates the fade up ramp speed of the specified input channel's "B" bus fader.

5.6.14 UMIX DRAMPB

Target: UMIX:#.#
Parameter: DRAMPB
Channel: Mixer # . Input Channel (see above)
Value: 0=Off, 1=Fast, 2=Medium, 3=Slow
Query: Yes
Command: Yes
Protocol Rev: 1.4

The UMIX DRAMPB parameter may be queried or set by the automation application on the PC. The parameter value indicates the fade down ramp speed of the specified input channel's "B" bus fader.

5.6.15 UMIX BALA

Target: UMIX:#.#
Parameter: BALA
Channel: Mixer # . Input Channel (see above)
Value: -100 to 100 = percent left (negative) to percent right (positive) or centered (zero).
Query: Yes
Command: Yes
Protocol Rev: 1.9

The UMIX BALA parameter may be queried or set by the automation application on the PC. The parameter value indicates the A bus fader's balance knob position in percent. A value of 0 equals centered, a value of -100 is full left and +100 is full right. Only input faders have balance.

Input Fader Example:

```
Query:    <UMIX:1.3?BALA>          Read Utility Mixer 1 Input 3 A balance
Response: <UMIX:1.3|BALA:0>        It's at 0 = centered
Command:  <UMIX:1.3|BALA:-100>     Set Utility Mixer 1 Input 3 A balance to full left
Response: <OK>
~ or ~
Command:  <UMIX:1.3|BALA:50>       Set Utility Mixer 1 Input 3 A balance half way to full right.
Response: <OK>
```

5.6.16 UMIX BALB

Target: UMIX:#.#
Parameter: BALB
Channel: Mixer # . Input Channel (see above)
Value: -100 to 100 = percent left (negative) to percent right (positive) or centered (zero).
Query: Yes
Command: Yes
Protocol Rev: 1.9

The UMIX BALB parameter may be queried or set by the automation application on the PC. The parameter value indicates the B bus fader's balance knob position in percent. A value of 0 equals centered, a value of -100 is full left and +100 is full right. See the previous section for an example. Only input faders have balance.

5.6.17 UMIX DUCKA

Target: UMIX:#.#
Parameter: DUCKA
Channel: Mixer # . Input Channel (see above)
Value: 1 to duck 0 to un-duck the fader
Query: Yes
Command: Yes
Protocol Rev: 1.9

The UMIX DUCKA parameter may be queried or set by the automation application on the PC. The parameter value indicates whether the A bus fader is ducked or not. The default ducking level is -12dB, but that may be modified using UMIX DUCKLVL command to modify the ducking level for the utility mixer. Only input faders can be ducked.

Ducking Example:

```
Query:    <UMIX:1.0?DUCKLVL>       Read Utility Mixer 1 Ducking Level
Response: <UMIX:1.0|DUCKLVL:-12.0> It's at -12 dB (the default level)
```

```
Command:  <UMIX:1.0|DUCKLVL:-40.0> Set Ducking Level to -40 dB
Response: <OK>
Query:    <UMIX:1.3?DUCKA>         Read Utility Mixer 1 Input 3 A Ducking
Response: <UMIX:1.3|DUCKA:0>       It's at 0 = NOT ducked
Command:  <UMIX:1.3|DUCKA:1>       Duck Utility Mixer 1 Input 3 A (reduce level by 40 dB)
Response: <OK>
Command:  <UMIX:1.3|DUCKA:0>       Un-Duck Utility Mixer 1 Input 3 A (restore level)
Response: <OK>
```

5.6.18 UMIX DUCKB

Target:	UMIX:#.#
Parameter:	DUCKB
Channel:	Mixer # . Input Channel (see above)
Value:	1 to duck 0 to un-duck the fader
Query:	Yes
Command:	Yes
Protocol Rev:	1.9

The UMIX DUCKB parameter may be queried or set by the automation application on the PC. The parameter value indicates whether the B bus fader is ducked or not. The default ducking level is -12dB, but that may be modified using UMIX DUCKLVL command to modify the ducking level for the utility mixer. Only input faders can be ducked. See the previous section for an example.

5.7 SLIO Parameters

Software LIO targets are used to query or set software LIO pin levels.

Note: Use the <SYS?SLIO> query to determine how many software LIO pins are available on the Blade.

See Also: Appendix B which describes the WheatNet IP Blade software LIO pin design.

5.7.1 SLIO LVL

Target:	SLIO:#
Parameter:	LVL
Channel:	Software LIO index (1 - ??)
Value:	1 = Active Logic, 0 = Inactive Logic
Query:	Yes
Command:	Yes
Protocol Rev:	1.2

The SLIO LVL parameter may be queried or set by the automation application on the PC. The parameter value indicates the logic level state of the specified soft LIO pin.

Note: You can not set software LIO pins configured as outputs from the WheatNet IP system, just those that are inputs to the WheatNet IP system. If you set a software LIO pin level for a pin configured as an output or not configured at all, your command will be acknowledged with an <OK> but the level that you specified will quietly be ignored. I want to stress that it's important that you design your command logic with a knowledge of the signal definitions setup in the WheatNet IP Navigator GUI.

```
Query:    <SYS?SLIO>
Response: <SYS|SLIO:20>          Twenty soft LIO pins available in the Blade
Query:    <SLIO:1?LVL>
Response: <SLIO:1|LVL:0>         First set it logic low
Query:    <SLIO:1|LVL:1>         Now it's set logic high
```

5.8 LIO Parameters

LIO targets are used to query or set LIO pin levels.

Note: Use the <SYS?LIO> query to determine how many physical LIO pins are available on the Blade on card 0. Note that this count does not include the LIOs provided via 3rd Party expansion devices which appear on other card numbers starting at card 49 for device 1. For access to this information use WNIP Navigator.

5.8.1 LIO LVL

Target: LIO:#
Parameter: LVL
Channel: LIO index (0 - ??) on card 0 or use CARD.CIRCUIT (ie. 49.0 for card 49 circuit 0) for access to other cards.
Value: 1 = Active Logic, 0 = Inactive Logic
Query: Yes
Command: Yes
Protocol Rev: 1.10

The LIO LVL parameter may be queried or set by the automation application on the PC. The parameter value indicates the logic level state of the specified LIO pin.

Note: You can not actually set physical LIO pins configured as inputs to the WheatNet IP system, just those that are outputs from the WheatNet IP system. If you set a LIO pin level for a pin configured as an input or not configured at all, it will generate an LIO event indicating transition of the LIO as if the input pin transitioned. However, the physical input pin level on the Blade or device can still transition and override the automation-set level if it changes from it's previous state. So, if the pin is to be software-controlled as an input it should be disconnected to avoid conflict, and you should consider using Software LIOs as these will not consume physical pin resources.

```
Query:    <SYS?LIO>
Response: <SYS|LIO:12>          Twelve hard LIO pins available in the Blade (card 0)
Query:    <LIO:0.1?LVL>         First read card 0, circuit 1
Response: <LIO:0.1|LVL:0>       it's logic low
Command:  <LIO:0.1|LVL:1>       Now set logic high
Response: <OK>
OR
Query:    <LIO:1?LVL>          Query card 0, circuit 1
Response: <LIO:0.1|LVL:1>      LIO card 0, circuit 1 is logic high
```

5.9 SURFSPARE Parameters

SURFSPARE targets are used to query the spare buttons state on a surface connected to the Blade.

Note: For this query to not return a NAK, it must be performed on a surface-capable Blade. There must be a surface attached and connected to this Blade for the spare buttons to reflect actual values, otherwise they will default to level 0.

5.9.1 SURFSPARE LVL

Target: SURFSPARE:#
Parameter: LVL
Channel: Spare Button Number (0 - ??) on Surface
Value: 1 = Active Logic, 0 = Inactive Logic
Query: Yes
Command: Yes
Protocol Rev: 1.10

```
Query:    <SURFSPARE:0?LVL>     Read Surface Spare Button 1
Response: <SURFSPARE:0|LVL:0>   It's logic low
```

5.10 TIME Parameters

TIME targets are used to query for system time.

5.10.1 TIME TIME

Target: TIME
Parameter: TIME

Value: Time/Date in MM DD YY hh.mm.ss.xxx where M:month D:day Y:year h:hour m:minute s:second x: milliseconds
Query: Yes
Command: Yes
Protocol Rev: 1.10

```
Query:    <TIME?TIME>                          Query Blade time
Response: <TIME|TIME:08 01 13 13.23.51.605>    Time received.
                                                 Parsing is trivial, the value here is
                                                 August 1st, 2013 13:23:51 and 605 milliseconds
```

5.11 STRING Parameters

STRING targets are used to query and set string variables. These variables are stored on each blade, they are not distributed. Each blade has its own bank of string variables. The variables will not persist through a blade reboot.

5.11.1 STRING VAL

Target:	STRING
Parameter:	VAL
Channel:	String number (1 - ??) on Blade (Get max number using <SYS?STRING>)
Value:	String variable. Maximum 255 characters. Note that no special characters that are used for parsing are allowed. [? < > | :]
Query:	Yes
Command:	Yes
Protocol Rev:	1.10 (some as the protocol revision in the software was not updated to 1.11 as intended)

```
Command:  <STRING:1|VAL:Hello World>           Command String
Response: <OK>

Query:    <STRING:1?VAL>                       Query String
Response: <STRING:1|VAL:Hello World>
```

5.12 MIC Parameters

MIC targets are used to query and set Microphone source parameters for M4/M4USB/M8 blade types.

5.12.1 MIC PPWR

Target:	MIC
Parameter:	PPWR
Channel:	Source ID in either Hexadecimal or dotted notation
Value:	0 – Disable Phantom Power 1 – Enable Phantom Power
Query:	Yes
Command:	Yes
Protocol Rev:	1.14

```
Command:  <MIC:06000C00|PPWR:1>                Command String
Response: <OK>

Query:    <MIC:24.0.6.0?PPWR>                  Query String
Response: <MIC:24.0.6.0|PPWR:1>
```

6 Event Subscription Messages

Your application can subscribe to certain asynchronous events which may occur due to system activity. This section describes how your application can subscribe to events related to source and destination signals.

To receive notification of asynchronous events your application must register a subscription to the event with the Blade. Once your application has subscribed to one or more events, your application will receive asynchronous messages over the TCP socket. The format of these event messages is the same as any other message type. Since they are asynchronous, you may receive event notifications when you don't expect them. For example your application might send a command to the surface and expect to see an <OK> acknowledgment. There is always a slim chance that at that very instance a cross point connection is changed on a destination that you have subscribed to, you may receive the destination connection event before

you receive the <OK> for a command that you just sent. So please think asynchronously in your message handling when you use subscriptions. You will not ever receive nested messages (i.e. <DST_E<OK>VENT|SRC:04000001> will not occur).

When you first establish a TCP connection to the Blade there will be no subscriptions. Therefore if your application disconnects and reconnects to the Blade, you must re-subscribe to any events you wish to monitor.

Each of the up to 20 connections are independent, each has it's own unique subscription list.

6.1 SRCSUB

The SRCSUB message targets allow you to subscribe to events on source signal properties.

When you subscribe to a source attribute you will specify the source signal ID that you want to monitor. You may subscribe to all sources by specifying a source signal of "**FFFFFFFF**" or "***.*.*.***".

6.1.1 SRCSUB NAME

Target: SRCSUB:#
Parameter: NAME
Channel: Source signal number
Value: 1 to subscribe, 0 to unsubscribe
Query: Yes
Command: Yes
Protocol Rev: 1.2

The SRCSUB NAME parameter may be queried or set by the automation application on the PC. The parameter value indicates the state of your subscription to NAME change events on a particular source signal. A value of '0' indicates no subscription, a value of '1' indicates a subscription.

Example:

```
Query:    <SRCSUB:00400001?NAME>
Response: <SRCSUB:00400001|NAME:0>           The name property is not subscribed to on this signal

Command:  <SRCSUB:00400001|NAME:1>           Subscribe to the name property on this signal
Response: <OK>
Event:    <SRCEVENT:00400001|NAME:Joes Mic>  You immediately receive an event for this subscription
Event:    <SRCEVENT:00400001|NAME:Bobs Mic>  Later on the Navigator GUI the signal name is changed
```

6.1.2 SRCSUB LOCATION

Target: SRCSUB:#
Parameter: LOCATION
Channel: Source signal number
Value: 1 to subscribe, 0 to unsubscribe
Query: Yes
Command: Yes
Protocol Rev: 1.2

The SRCSUB LOCATION parameter may be queried or set by the automation application on the PC. The parameter value indicates the state of your subscription to LOCATION change events on a particular source signal. A value of '0' indicates no subscription, a value of '1' indicates a subscription.

Example:

```
Query:    <SRCSUB:00400001?LOCATION>
Response: <SRCSUB:00400001|LOCATION:0>    The location property is not subscribed to on this signal

Command:  <SRCSUB:00400001|LOCATION:1>           Subscribe to the location property on this signal
Response: <OK>
Event:    <SRCEVENT:00400001|LOCATION:Studio A>  You immediately receive an event for this subscription
Event:    <SRCEVENT:00400001|LOCATION:Studio B>  Later the signal location string is changed
```

6.1.3 SRCSUB DEF

Target: SRCSUB:#
Parameter: DEF
Channel: Source signal number
Value: 1 to subscribe, 0 to unsubscribe
Query: Yes
Command: Yes
Protocol Rev: 1.2

The SRCSUB DEF parameter may be queried or set by the automation application on the PC. The parameter value indicates the state of your subscription to DEF change events on a particular source signal. A value of '0' indicates no subscription, a value of '1' indicates a subscription.

Example:

```
Command:  <SRCSUB:*.*.*.*|DEF:1,NAME:1>     Subscribe to defined and name properties on all sources
Response: <OK>
Event:    <SRCEVENT:00400001|DEF:1,NAME:Joes Mic> You immediately receive events for this subscription
Event:    <SRCEVENT:00400002|DEF:1,NAME:Bobs Mic> ... there will be several due to the wild card.
...
...
Event:    <SRCEVENT:00400002|DEF:0>               Later on source 00400002 is deleted.
...
Event:    <SRCEVENT:00400003|DEF:1,NAME:Sues Mic> Later on source 00400003 is created.
```

6.1.4 SRCSUB GAINT

Target: SRCSUB:#
Parameter: GAINT
Channel: Source signal number
Value: 1 to subscribe, 0 to unsubscribe
Query: Yes
Command: Yes
Protocol Rev: 1.10 (some as the protocol revision in the software was not updated to 1.11 as intended)

The SRCSUB GAINT parameter may be queried or set by the automation application on the PC. The parameter value indicates the state of your subscription to GAINT value change events on a particular source signal. A value of '0' indicates no subscription, a value of '1' indicates a subscription. Typically signal gain types do not change so typically you would get these events only when you first subscribe to them and never receive any more. You will get GAINT events when a signal is created if you have subscribed to a wildcard range of signals. See "Appendix C – Signal Gain Type Enumerations" for a table of gain type values.

6.1.5 SRCSUB GAIN

Target: SRCSUB:#
Parameter: GAIN
Channel: Source signal number
Value: 1 to subscribe, 0 to unsubscribe
Query: Yes
Command: Yes
Protocol Rev: 1.10 (some as the protocol revision in the software was not updated to 1.11 as intended)

The SRCSUB GAIN parameter may be queried or set by the automation application on the PC. The parameter value indicates the state of your subscription to GAIN level change events on a particular source signal. A value of '0' indicates no subscription, a value of '1' indicates a subscription.

Example:

```
Query:    <SRCSUB:00400001?GAIN>
Response: <SRCSUB:00400001|GAIN:0>            The gain level property is not subscribed to on this
                                              signal

Command:  <SRCSUB:00400001|GAIN:1>            Subscribe to the gain level property on this signal
Response: <OK>
Event:    <SRCEVENT:00400001|GAIN:100>        You immediately receive an event for this subscription
Event:    <SRCEVENT:00400001|GAIN:105>        Later the gain level is changed to 10.5dB
```

6.2 DSTSUB

The DSTSUB message targets allow you to subscribe to events on destination signal properties.

When you subscribe to a destination attribute you will specify the destination signal ID that you want to monitor. You may subscribe to all destinations by specifying a destination signal of "**FFFFFFFF**" or "***.*.*.***".

6.2.1 DSTSUB NAME

Target: DSTSUB:#
Parameter: NAME
Channel: Destination signal number
Value: 1 to subscribe, 0 to unsubscribe
Query: Yes
Command: Yes
Protocol Rev: 1.2

The DSTSUB NAME parameter may be queried or set by the automation application on the PC. The parameter value indicates the state of your subscription to NAME change events on a particular destination signal. A value of '0' indicates no subscription, a value of '1' indicates a subscription.

Example:

```
Query:    <DSTSUB:00400001?NAME>
Response: <DSTSUB:00400001|NAME:0>           The name property is not subscribed to on this signal

Command:  <DSTSUB:00400001|NAME:1>           Subscribe to the name property on this signal
Response: <OK>
Event:    <DSTEVENT:00400001|NAME:JoesHdpn>  You immediately receive an event for this subscription
Event:    <DSTEVENT:00400001|NAME:BobsHdpn>  Later on the Navigator GUI the signal name is changed
```

6.2.2 DSTSUB LOCATION

Target: DSTSUB:#
Parameter: LOCATION
Channel: Destination signal number
Value: 1 to subscribe, 0 to unsubscribe
Query: Yes
Command: Yes
Protocol Rev: 1.2

The DSTSUB LOCATION parameter may be queried or set by the automation application on the PC. The parameter value indicates the state of your subscription to LOCATION change events on a particular destination signal. A value of '0' indicates no subscription, a value of '1' indicates a subscription.

Example:

```
Query:    <DSTSUB:00400001?LOCATION>
Response: <DSTSUB:00400001|LOCATION:0>           The location property is not subscribed on this signal

Command:  <DSTSUB:00400001|LOCATION:1>           Subscribe to the location property on this signal
Response: <OK>
Event:    <DSTEVENT:00400001|LOCATION:Studio A>  You immediately receive an event for this subscription
Event:    <DSTEVENT:00400001|LOCATION:Studio B>  Later the signal location string is changed
```

6.2.3 DSTSUB DEF

Target: DSTSUB:#
Parameter: DEF
Channel: Destination signal number
Value: 1 to subscribe, 0 to unsubscribe
Query: Yes
Command: Yes
Protocol Rev: 1.2

The DSTSUB DEF parameter may be queried or set by the automation application on the PC. The parameter value indicates the state of your subscription to DEF change events on a particular source signal. A value of '0' indicates no subscription, a value of '1' indicates a subscription.

Example:

```
Command:   <DSTSUB:*.*.*.*|DEF:1,NAME:1>  Subscribe to defined and name properties on all destinations.
Response:  <OK>
Event:     <DSTEVENT:00400001|DEF:1,NAME:JoesHdpn> You immediately receive events for this subscription
Event:     <DSTEVENT:00400002|DEF:1,NAME:BobsHdpn> ... there will be several due to the wild card.
...
...
Event:     <DSTEVENT:00400002|DEF:0>               Later on destination 00400002 is deleted.
...
Event:     <DSTEVENT:00400003|DEF:1,NAME:SuesHdpn> Later on destination 00400003 is created.
```

6.2.4 DSTSUB SRC

Target: DSTSUB:#
Parameter: SRC
Channel: Destination signal number
Value: 1 to subscribe, 0 to unsubscribe
Query: Yes
Command: Yes
Protocol Rev: 1.2

The DSTSUB SRC parameter may be queried or set by the automation application on the PC. The parameter value indicates the state of your subscription to connected SRC change events on a particular destination signal. A value of '0' indicates no subscription, a value of '1' indicates a subscription.

Example:

```
Query:     <DSTSUB:00400001?SRC>
Response:  <DSTSUB:00400001|SRC:0>            The source property is not subscribed to on this signal

Command:   <DSTSUB:00400001|SRC:1>            Subscribe to the source property on this signal
Response:  <OK>
Event:     <DSTEVENT:00400001|SRC:00800002>   You immediately receive an event for this subscription
Event:     <DSTEVENT:00400001|SRC:00800004>   Later cross point to this destination is changed
```

6.2.5 DSTSUB LOCKED

Target: DSTSUB:#
Parameter: LOCKED
Channel: Destination signal number
Value: 1 to subscribe, 0 to unsubscribe
Query: Yes
Command: Yes
Protocol Rev: 1.2

The DSTSUB LOCKED parameter may be queried or set by the automation application on the PC. The parameter value indicates the state of your subscription to LOCKED state change events on a particular destination signal. A value of '0' indicates no subscription, a value of '1' indicates a subscription.

Example:

```
Query:     <DSTSUB:00400001?LOCKED>
Response:  <DSTSUB:00400001|LOCKED:0>            The lock state property is not subscribed to on this
                                                 signal

Command:   <DSTSUB:00400001|LOCKED:1>           Subscribe to the lock state property on this signal
Response:  <OK>
Event:     <DSTEVENT:00400001|LOCKED:0>          You immediately receive an event for this subscription
Event:     <DSTEVENT:00400001|LOCKED:1>          Later the lock state is changed
```

6.2.6 DSTSUB GAINT

Target: DSTSUB:#
Parameter: GAINT
Channel: Destination signal number
Value: 1 to subscribe, 0 to unsubscribe
Query: Yes
Command: Yes
Protocol Rev: 1.10 (some as the protocol revision in the software was not updated to 1.11 as intended)

The DSTSUB GAINT parameter may be queried or set by the automation application on the PC. The parameter value indicates the state of your subscription to GAINT value change events on a particular destination signal. A value of '0' indicates no subscription, a value of '1' indicates a subscription. Typically signal gain types do not change so typically you would get these events only when you first subscribe to them and never receive any more. You will get GAINT events when a signal is created if you have subscribed to a wildcard range of signals. See "Appendix C – Signal Gain Type Enumerations" for a table of gain type values.

6.2.7 DSTSUB GAIN

Target: DSTSUB:#
Parameter: GAIN
Channel: Destination signal number
Value: 1 to subscribe, 0 to unsubscribe
Query: Yes
Command: Yes
Protocol Rev: 1.10 (some as the protocol revision in the software was not updated to 1.11 as intended)

The DSTSUB GAIN parameter may be queried or set by the automation application on the PC. The parameter value indicates the state of your subscription to GAIN level change events on a particular destination signal. A value of '0' indicates no subscription, a value of '1' indicates a subscription.

Example:

```
Query:    <DSTSUB:00400001?GAIN>
Response: <DSTSUB:00400001|GAIN:0>          The gain level property is not subscribed to on this
                                            signal

Command:  <DSTSUB:00400001|GAIN:1>          Subscribe to the gain level property on this signal
Response: <OK>
Event:    <DSTEVENT:00400001|GAIN:100>      You immediately receive an event for this subscription
Event:    <DSTEVENT:00400001|GAIN:105>      Later the gain level is changed to 10.5dB
```

6.3 SALVOSUB

The SALVOSUB message targets allow you to subscribe to events on salvo signal properties.

When you subscribe to a salvo attribute you will specify the salvo ID that you want to monitor. You may subscribe to all salvos by specifying a salvo ID of "*****".

6.3.1 SALVOSUB NAME

Target: SALVOSUB:#
Parameter: NAME
Channel: Salvo ID number (1 to 256)
Value: 1 to subscribe, 0 to unsubscribe
Query: Yes
Command: Yes
Protocol Rev: 1.2

The SALVOSUB NAME parameter may be queried or set by the automation application on the PC. The parameter value indicates the state of your subscription to NAME change events on a particular salvo. A value of '0' indicates no subscription, a value of '1' indicates a subscription.

6.3.2 SALVOSUB DEF

Target: SALVOSUB:#
Parameter: DEF
Channel: Salvo ID number (1 to 256)
Value: 1 to subscribe, 0 to unsubscribe
Query: Yes
Command: Yes
Protocol Rev: 1.3

The SALVOSUB DEF parameter may be queried or set by the automation application on the PC. The parameter value indicates the state of your subscription to DEF change events on a particular salvo. A value of '0' indicates no subscription, a value of '1' indicates a subscription.

6.3.3 SALVOSUB FIRE

Target: SALVOSUB:#
Parameter: FIRE
Channel: Salvo ID number (1 to 256)
Value: 1 to subscribe, 0 to unsubscribe
Query: Yes
Command: Yes
Protocol Rev: 1.10

The SALVOSUB FIRE parameter may be queried or set by the automation application on the PC. The parameter value indicates the state of your subscription to FIRE events on a particular salvo. A value of '0' indicates no subscription, a value of '1' indicates a subscription.

Example:

```
Command:  <SALVOSUB:*|DEF:1,NAME:1,FIRE:1> Subscribe to defined, name and fire properties on all salvos
Response: <OK>
Event:    <SALVOEVENT:1|DEF:1,NAME:Salvo1> You immediately receive events for this subscription
Event:    <SALVOEVENT:2|DEF:2,NAME:Salvo2> ... there may be up to 256 events due to the wild card.
Event:    <SALVOEVENT:3|DEF:0>
...
...
Event:    <SALVOEVENT:2|DEF:0>             Later on salvo 2 is deleted.
...
Event:    <SALVOEVENT:3|DEF:2,NAME:Salvo3> Later on salvo 3 is created.
Event:    <SALVOEVENT:1|FIRE:1>            Later on salvo 1 is fired.
```

6.4 UMIXSUB

The UMIXSUB message targets allow you to subscribe to events on the blade's utility mixers.

Note: When an UMIX target is specified the utility mixer number and the input or output channel must be a part of the message target. The message target is of the form #.#, where the first # indicates the utility mixer to access (1-2) and the second # indicate the input channel (1-8), or the output bus (A-B), or 0 in the special case of the mixer enabled query.

6.4.1 UMIXSUB ON

Target: UMIXSUB:#.#
Parameter: ON
Channel: Mixer # . Input Channel ~ or ~ Mixer # . Output Bus (see above)
Value: 1 to subscribe, 0 to unsubscribe
Query: Yes
Command: Yes
Protocol Rev: 1.6

The UMIXSUB ON parameter may be queried or set by the automation application on the PC. The parameter value indicates the state of your subscription to channel ON change events on a particular utility mixer channel. A value of '0' indicates no subscription, a value of '1' indicates a subscription.

Example:

```
Command:  <UMIXSUB:1.2|ON:1>        Subscribe changes on Utility Mixer 1 Input 2 on button state
Response: <OK>
Event:    <UMIXEVENT:1.2|ON:0>      You immediately receive events for this subscription
                                    In this example the channel 2 is off.
...
...
Event:    <UMIXEVENT:1.2|ON:1>      Later Utility Mixer 1 Input 2 is turned on.
...
Event:    <UMIXEVENT:1.2|ON:0>      Later Utility Mixer 1 Input 2 is turned back off.
```

6.4.2 UMIXSUB FDRA

Target: UMIXSUB:#.#
Parameter: FDRA
Channel: Mixer # . Input Channel
Value: 1 to subscribe, 0 to unsubscribe
Query: Yes
Command: Yes
Protocol Rev: 1.10 (some as the protocol revision in the software was not updated to 1.11 as intended)

The UMIXSUB FDRA parameter may be queried or set by the automation application on the PC. The parameter value indicates the state of your subscription to the specified utility mixer input channel's A fader level. A value of '0' indicates no subscription, a value of '1' indicates a subscription.

6.4.3 UMIXSUB FDRB

Target: UMIXSUB:#.#
Parameter: FDRB
Channel: Mixer # . Input Channel
Value: 1 to subscribe, 0 to unsubscribe
Query: Yes
Command: Yes
Protocol Rev: 1.10 (some as the protocol revision in the software was not updated to 1.11 as intended)

The UMIXSUB FDRB parameter may be queried or set by the automation application on the PC. The parameter value indicates the state of your subscription to the specified utility mixer input channel's B fader level. A value of '0' indicates no subscription, a value of '1' indicates a subscription.

6.4.4 UMIXSUB MFDR

Target: UMIXSUB:#.#
Parameter: MFDR
Channel: Mixer # . Output Bus
Value: 1 to subscribe, 0 to unsubscribe
Query: Yes
Command: Yes
Protocol Rev: 1.10 (some as the protocol revision in the software was not updated to 1.11 as intended)

The UMIXSUB MFDR parameter may be queried or set by the automation application on the PC. The parameter value indicates the state of your subscription to the specified utility mixer output bus' fader level. A value of '0' indicates no subscription, a value of '1' indicates a subscription.

6.4.5 UMIXSUB DUCKA

Target: UMIXSUB:#.#
Parameter: DUCKA
Channel: Mixer # . Input Channel
Value: 1 to subscribe, 0 to unsubscribe
Query: Yes
Command: Yes
Protocol Rev: 1.9

The UMIXSUB DUCKA parameter may be queried or set by the automation application on the PC. The parameter value indicates the state of your subscription to the specified utility mixer channel's A bus ducking control. A value of '0' indicates no subscription, a value of '1' indicates a subscription.

6.4.6 UMIXSUB DUCKB

Target: UMIXSUB:#.#
Parameter: DUCKB
Channel: Mixer # . Input Channel
Value: 1 to subscribe, 0 to unsubscribe
Query: Yes
Command: Yes
Protocol Rev: 1.9

The UMIXSUB DUCKB parameter may be queried or set by the automation application on the PC. The parameter value indicates the state of your subscription to the specified utility mixer channel's B bus ducking control. A value of '0' indicates no subscription, a value of '1' indicates a subscription.

6.5 SLIOSUB

The SLIOSUB message targets allow you to subscribe to events on software LIO pins.

See Also: Appendix B which describes the WheatNet IP Blade software LIO pin design.

6.5.1 SLIOSUB LVL

Target:	SLIOSUB:#
Parameter:	LVL
Channel:	Software LIO pin number
Value:	1 to subscribe, 0 to unsubscribe
Query:	Yes
Command:	Yes
Protocol Rev:	1.2

The SLIOSUB LVL parameter may be queried or set by the automation application on the PC. The parameter value indicates the state of your subscription to LVL change events on a particular software LIO pin. A value of '0' indicates no subscription, a value of '1' indicates a subscription.

Note: You will not receive SLIOEVENT messages for software LIO pins configured as inputs to the system, just those that are outputs from the system. If you subscribe to a software LIO pin configured as an input to the system or not configured at all, your command will be acknowledged with an <OK>, but you will deafly never receive any events for that pin. I want to stress again that it's important that you design your command logic with a knowledge of the signal definitions setup in the WheatNet IP Navigator GUI.

Example:

```
Query:    <SLIOSUB:1?LVL>
Response: <SLIOSUB:1|LVL:0>    The logic level is not subscribed to on this soft LIO pin

Command:  <SLIOSUB:1|LVL:1>    Subscribe to the logic level on this soft LIO pin
Response: <OK>
Event:    <SLIOEVENT:1|LVL:0>  You immediately receive an event for this subscription
Event:    <SLIOEVENT:1|LVL:1>  Later on the software LIO output pin logic level has changed
```

6.6 LIOSUB

The LIOSUB message targets allow you to subscribe to events on LIO pins.

6.6.1 LIOSUB LVL

Target:	LIOSUB:#
Parameter:	LVL
Channel:	LIO card and pin number (CARD.PIN ie. 0.1)
Value:	1 to subscribe, 0 to unsubscribe
Query:	Yes
Command:	Yes
Protocol Rev:	1.10

The LIOSUB LVL parameter may be queried or set by the automation application on the PC. The parameter value indicates the state of your subscription to LVL change events on a particular LIO pin. A value of '0' indicates no subscription, a value of '1' indicates a subscription.

Example:

```
Query:    <LIOSUB:0.1?LVL>
Response: <LIOSUB:0.1|LVL:0>    The logic level is not subscribed to on this LIO pin

Command:  <LIOSUB:0.1|LVL:1>    Subscribe to the logic level on this LIO pin
Response: <OK>
Event:    <LIOEVENT:0.1|LVL:0>  You immediately receive an event for this subscription
Event:    <LIOEVENT:0.1|LVL:1>  Later on the LIO output pin logic level has changed
```

6.7 SURFSPARESUB

The SURFSPARESUB message targets allow you to subscribe to events on surface spare buttons.

6.7.1 SURFSPARESUB LVL

Target: SURFSPARESUB:#
Parameter: LVL
Channel: Surface spare button number (0-N)
Value: 1 to subscribe, 0 to unsubscribe
Query: Yes
Command: Yes
Protocol Rev: 1.10

The SURFSPARESUB LVL parameter may be queried or set by the automation application on the PC. The parameter value indicates the state of your subscription to LVL change events on a particular surface spare button pin. A value of '0' indicates no subscription, a value of '1' indicates a subscription.

Example:

```
Query:    <SURFSPARESUB:0?LVL>
Response: <SURFSPARESUB:0|LVL:0>    The logic level is not subscribed to on this surface spare button

Command:  <SURFSPARESUB:0|LVL:1>    Subscribe to the logic level on this surface spare button
Response: <OK>
Event:    <SURFSPAREEVENT:0|LVL:0>  You immediately receive an event for this subscription
Event:    <SURFSPAREEVENT:0|LVL:1>  Later on the button logic level has changed
```

6.8 TIMESUB

The TIMESUB message targets allow you to subscribe a periodic current blade time broadcast (~once per second).

6.8.1 TIMESUB TIME

Target: TIMESUB
Parameter: TIME

Value: 1 to subscribe, 0 to unsubscribe
Query: Yes
Command: Yes
Protocol Rev: 1.10

The TIMESUB TIME parameter may be queried or set by the automation application on the PC. The parameter value indicates the state of your subscription to TIME periodic events. A value of '0' indicates no subscription, a value of '1' indicates a subscription.

Example:

```
Query:    <TIMESUB|TIME:1>
Response: <OK>
Response: <TIMEEVENT|TIME:08 01 13 13.23.51.605>   Time Event is received within a second
                                                   Parsing is trivial, the value here is August 1st,
                                                   2013 13:23:51 and 605 milliseconds

Response: <TIMEEVENT|TIME:08 01 13 13.23.52.671>   Events continue approximately once per second
```

6.9 STRINGSUB

The STRINGSUB message targets allow you to subscribe a STRING variable change.

6.9.1 STRINGSUB VAL

Target: STRINGSUB
Parameter: VAL

Value: 1 to subscribe, 0 to unsubscribe
Query: Yes
Command: Yes
Protocol Rev: 1.10 (some as the protocol revision in the software was not updated to 1.11 as intended)

The STRINGSUB VAL parameter may be queried or set by the automation application on the PC. The parameter value indicates the state of your subscription to STRING variable change events. A value of '0' indicates no subscription, a value of '1' indicates a subscription.

Example:

```
Query:    <STRINGSUB|VAL:1>
Response: <OK>
Response: <STRINGEVENT|VAL:Hello World>   on String change
```

Appendix A – Automation Enabled Blades

At the time of this writing all Blade or Blade like devices except the PC audio driver have the automation interfaces described in this document built into them.

The intention of placing the automation interface into the PC audio driver is so that an automation application running on a PC with a driver may interface to the WheatNet IP network for making cross point changes or firing salvos by simply using it's loop back IP address (127.0.0.1). That same automation interface may also access software LIO pins bound to the PC audio driver's signals using that TCP interface.

Appendix B – Blade Software LIOs

The WheatNet IP system typically uses hardware I/O pins on the back panel of a Blade for bringing LIO logic levels into and out of the WheatNet IP network. This works great for LIOs that end up being hard wired to physical pieces of hardware, but in the modern day of more and more computer based software devices it's not such a great way to get data into and out of these software based devices. So a software solution is needed.

On the back panel of an IP-88a Blade there are 12 bidirectional LIO pins. These 12 physical pins go into a pool of LIO pins in that Blade which may be assigned to any source or destination signal created in that Blade using the WheatNet IP Navigator GUI. We've taken a similar approach to software controlled LIO pins by creating a pool of virtual LIO pins for each Blade. These virtual software LIO pins also go into the LIO pin pool and may be assigned to any user signals on that Blade[1].

These software LIO pins are only addressable via the automation interfaces. Therefore they may be used as LIO entry points for automation PCs or for peripheral devices such as Wheatstone GP16P button panels or other network based LIO controllers.

To access a software LIO pin you must make a TCP connection to the Blade that owns that software LIO pin.

To determine how many software LIO pins are available in a Blade the automation device may use the <SYS?SLIO> command. To drive the software LIO input pins the automation device may use the <SLIO:#|LVL:1> or <SLIO:#|LVL:0> commands. To read the software LIO output pins the automation device may poll them using the <SLIO:#?LVL> query or the automation device may subscribe to software LIO output pin events by subscribing to an output pin using the <SLIOSUB#|LVL:1> command. Using subscriptions for receiving LIO output events is the preferred mechanism since it requires much less network traffic than polling would and will guarantee delivery of short duration logic level pulses.

The software LIO pins in a Blade are free resources for an end user to setup in any manner he/she so desires. As an automation programmer you will need to know the purpose of each software LIO that you access. It's up to you and the end customer to determine how you do that. You may want to specify for example on a PC audio driver which soft LIO pins he

[1] Blades with automation protocol revisions prior to revision 1.2 do not have any software LIO pins. Blades at automation protocol 1.2 will have some number of pins. Also software pins (just like physical pins) may be assigned to signals owned by another Blade device via the remote LIO mechanism.

should set up to perform such functions as machine start, machine stop, etc., or optionally allow them to configure your software with the soft LIO pins that they expect you to use. Either way should work fine. The following table shows how a typical 2 channel PC audio driver may be setup, the shaded columns show which attributes are configurable for each PC audio driver signal.

PC Audio Driver Software LIO Pins

Signal	*Soft LIO ID*	*Wire*	*Enable*	*Invert*	*Direction*	*Function*
Source 1	1	Soft LIO 1	Yes	No	Out	Machine Start
Source 1	2	Soft LIO 2	Yes	No	Out	Machine Stop
Source 1	3	Soft LIO 3	Yes	No	In	Machine Ready
Source 2	4	Soft LIO 4	Yes	No	Out	Machine Start
Source 2	5	Soft LIO 5	Yes	No	Out	Machine Stop
Source 2	6	Soft LIO 6	Yes	No	In	Machine Ready
Destination 1	7	Soft LIO 7	Yes	No	Out	Record
Destination 2	8	Soft LIO 8	Yes	No	Out	Record
~ unused ~	9	Soft LIO 9	No	No	n/a	none
~ unused ~	…	…	…	…	…	…
~ unused ~	48	Soft LIO 48	No	No	n/a	none

Appendix C – Signal Gain Type Enumerations

The following table shows the gain type enumerations which may be reported for various source and destination signals when their GAINT parameter is queried or subscribed to.

Value	*Range dB*	*Range ACI*	*Typical Use*
0	No Adjustment	"0"	Any signal which does not have integrated I/O gain adjustment.
1	-18dB to +18dB	"-180" to "180"	Blade rear panel audio I/O.
2	0dB to +80dB	"0" to "800"	Mic Blade rear panel audio inputs.
3	-60dB to 0dB	"-600" to "0"	Blade headphone jack.