

D I G I T A L S Y S T E M S G R O U P

BASIS

THE NETWORK AUDIO SOLUTION



Managing QSC's BASIS Products via Third-Party Control Systems (Crestron, AMX, Cue, etc.)

Published by:
QSC Audio Products, LLC
1675 MacArthur Blvd.
Costs Mesa, Ca. 92626

Document version: 3.2.0
Software release: 3.XX.XX
Date: 3 Mar 2009

Abstract

This document provides a detailed description of QSC's third party control interface for the BASIS platform of products. The control interface presented in this document is accessible over a local area network by establishing a remote TCP/IP connection with a BASIS product. This document does not address BASIS management through third party serial control mechanisms or through the OMNI control ports that are available on some BASIS models.

This document is intended for programmers and designers developing custom applications for managing QSC's BASIS products. The developer may be creating their application for deployment on a common third party control system or they may be developing their application for deployment on a Windows PC, a wireless handheld device or even a proprietary piece of hardware. In each of these applications, the developer must know the language and format used to communicate instructions to a BASIS product so that these instructions are interpreted and processed appropriately. QSC's third party control interface makes use of the extensible markup language (XML) to represent a unique set of instructions and a set of managed object identifiers (OIDs) that form the core of our proprietary protocol. This document describes these instructions, the set of managed objects that are accessible in BASIS and the means for assembling them in a manner that is both meaningful to BASIS and appropriate for the application.

Table of Contents

Table of Contents.....	2
Revision History:.....	4
Overview	4
BASIS Architecture.....	5
Configurations	5
Snapshots	5
Original Loaded Settings	6
Snapshot Behavior	6
Device Parameters	8
BASIS Communication	8
Restrictions	8
Control Gating	9
Polling.....	9
Control Registration.....	10
Forcing Control Values.....	10
XML Command Overview	11
Commands and Controls	14
Mute Commands.....	14
Mute GET Command.....	15
Mute GET Command Example	15
Mute SET Command	15
Mute SET Command Example	16
Gain Commands	17
Gain GET Command	18
Gain GET Command Example	18
Gain SET Command.....	19
Gain SET Command Example.....	19
Config/Snapshot Commands	20
Config/Snapshot GET Command	20
Config/Snapshot GET Command Example	21
Config/Snapshot SET Command.....	21
Config/Snapshot SET Command Example.....	23
Sensing and Controlling	24
Relay / Contact Closure Commands	24
Relay Output GET Command.....	24
Relay Output GET Command Example.....	25
Relay Output SET Command	25
Relay Output SET Command Example	26
OMNI Input Commands	26
OMNI Input GET Command	26
OMNI Input GET Command Example.....	27
Logic Output Commands.....	28
Logic Output GET Command.....	28
Logic Output GET Command Example.....	29
Logic Output SET Command	29
Logic Output SET Command Example	30
Amplifiers.....	31
Amplifier Standby Commands	31
Amplifier Standby GET Command	32
Amplifier Standby GET Command Example	32
Amplifier Standby SET Command	32
Amplifier Standby SET Command Example.....	33
Amplifier Power Status Commands.....	34

Managing BASIS via 3rd Party Control Systems

Amplifier Power Status GET Command	35
Amplifier Power Status GET Command Example	35
Amplifier Clip Status Commands	36
Amplifier Clip Status GET Command	37
Amplifier Clip Status GET Command Example	37
Amplifier Protect Status Commands	38
Amplifier Protect Status GET Command	38
Amplifier Protect Status GET Command Example	39
Amplifier Temperature Commands	40
Amplifier Temperature GET Command	40
Amplifier Temperature GET Command Example	41
Monitor Chain Commands	42
Monitor Tap Selection	42
Monitor Tap Select GET Command	42
Monitor Tap Select GET Command Example	43
Monitor Tap Select SET Command Example	43
Monitor Gain Commands	44
Monitor Gain GET Command	45
Monitor Gain GET Command Example	45
Monitor Gain SET Command	46
Monitor Gain SET Command Example	47
DSP Chain Commands	48
Dynamic DSP Overview	48
Dynamic DSP Cautions	48
Dynamic OID Calculation	49
Parameter Codes by Processor	50
OID Calculation for DSP Objects except Crossover and Mixer	55
OID Calculation for Crossover	56
OID Calculation for Mixer	58
Dynamic DSP Command Example	59
Dynamic DSP Control Troubleshooting	60
Keep Alive Command	61
Control Registration Command	62
Troubleshooting	64
Appendix A – Configuration Contents	67
Appendix B – Snapshot Parameters	68
Appendix C – DSP Processors	69
Appendix D – XML Overview	70
XML Structure	70
XML Formatting	70
Appendix E - Using a HyperTerm telnet session to send/receive Basis data	72
Appendix F - Managing Global Control Objects	73
Appendix G - BASIS / RAVE CobraNet OIDs	90

Revision History:

Rev	Change Description
2.1	First Draft. Incorporates material from older document.
3.0.1	Supports BASIS firmware versions 3.00.x, update to general information, various corrections
3.2.0	Edits and corrections to XML code snippets and examples

Overview

This document will describe how to manage (configure, control and monitor) most of the features of the BASIS family of products from a third-party management device. A management device may be as simple as an embedded microcontroller that includes network accessibility or as complex as a multi-core PC running Windows.

A BASIS can be managed by sending messages to the BASIS that instruct the product to manipulate specific objects or to return status or meter data to the managing device. In many applications the managing device happens to be a common third party control system such as a network enabled touch panel. A third party control system must be able to assemble all instructions directed at BASIS products in a manner that complies with QSC's XML based message format, and must be able to establish a network connection to the BASIS devices (as opposed to a serial port connection).

Direct support for Ethernet is not a requirement. It is perfectly acceptable to employ a managing device that communicates via USB, Blue Tooth, or even ATM, so long as the managing device is able to establish a TCP connection on port 4446 and so long as the third party control system provides a bridge to the BASIS Ethernet LAN.

The BASIS third party control interface makes use of a specific TCP port (port 4446) for all communications between a BASIS and its managing device. All XML messages directed at the BASIS product are therefore delivered through this TCP port. Likewise, the BASIS will use TCP port 4446 when directing XML messages towards the managing device. Therefore, two basic requirements of any third party control system is that it is capable of establishing a TCP connection on port 4446 and that port 4446 is not in use for any other service other than managing BASIS products.

QSC's use of TCP/IP in the control protocol allows BASIS products to be managed by any device that is capable of establishing a TCP/IP connection so long as the managing device can perform basic string manipulation or at least has the ability to source pre-assembled XML strings.

All XML messages sourced from a managing device towards a BASIS product are sent "directly" to the BASIS. Therefore, no additional BASIS-related hardware or computers are required. If the designer prefers to deploy computers on the network with one of the QSCControl.net applications running, any modifications to the system that are made through the third party control system will be reflected in these applications. As a result, QSC's Venue Manager, QSCreator, QSCAD and Notify applications will display and report any system changes, status or faults invoked by a third party control system.

We believe the BASIS third party control interface is one of the most flexible solutions for managing BASIS with third party systems. However, our solution does require the use of XML and it will almost certainly require some amount of programming for the third party control system application and/or for a host PC solution. If you are new to third party control systems or XML, we have provided some information resources that can help get you started. These resources are located in the appendices at the end of this document.

Also check out the Basis Term.exe application in the QSCControl.net "Third Party" installation folder which aids in creating the proper xml messages.

BASIS Architecture

In order to decide what objects to control and when to control them, it may be helpful to briefly discuss the BASIS hardware and software architecture and the operation of the BASIS family of products.

The BASIS products are all built on a common hardware and software “platform” architecture. While many features are model-specific, there is a core feature set that is common to all models. If the designer of a third party control solution were to concentrate on the elements that are common to all BASIS models, the designer could create a control solution that works with the entire BASIS family of products.

However, if management of model-specific features are required, the command and argument structure of our XML based protocol are organized in way that makes it easy to add model-specific modules to an otherwise common block of third party management commands that works with all models. And if the target application requires management of objects within the DSP chain, this can also be added to the base (or common) command set in a modular way.

Configurations

A Configuration is an overall software setup for a BASIS product. A Configuration includes settings to manage CobraNet attributes, settings to manage the properties of the analog inputs, settings to manage the properties of the analog outputs, settings to manage the content of the DSP chain, and settings to manage virtually any other software-programmable elements in the product where it is reasonable to expect to change their Configuration values in varying use cases. A table showing a list of the types of data included in a Configuration is included in Appendix A at the end of this document. A Configuration does not contain information that relates to the identity of the device such as its name or its IP address. These things are not expected to change once a product is deployed in a venue.

All BASIS products currently in production include eight separate Configurations. Of these eight Configurations, one is designated as the active Configuration. The active Configuration represents is the operating Configuration of the product. In other words, the active Configuration represents the signal flow that is being heard (of course you won't actually hear anything unless the Configuration is designed to pass audio). The remaining seven Configurations are stored for later recall into the active role. Recalling of Configurations can be performed through the BASIS front panel LCD, from QSC's Venue Manager software application, from an XML message sourced from a third party control device, or from a change on the BASIS product's rear panel OMNI inputs (this requires that the design associate a Configuration recall event with an OMNI input setting).

The Configuration/Snapshot commands can be used to control which of the eight Configurations is in the active role. The eight Configurations can be (and often are) very different, so that changing the active Configuration number can effect major change in the overall structure of the signal chain within a BASIS. That is because the entire signal chain (among other things) can be different in each Configuration that is stored within the device.

Switching between Configurations in a BASIS is accompanied by a muting of the audio while the device loads the new Configuration. This is a protection mechanism to prevent damage to amplifiers, speakers (or human ears) due to a discontinuity or remapping of audio in the DSP chain or in the CobraNet Configuration. A Configuration change can be thought of as being similar to reconnecting a bunch of analog gear to do another task – the potential for major change in the signal path exists, and is even likely.

Snapshots

BASIS products have many internal controls. Many of these controls can be changed at runtime. For example, there are controls to manage the audio inputs, the audio outputs, the processing elements in the DSP chain, the CobraNet attributes, and many other properties. Of these controls, some can be manipulated at runtime and some cannot. The controls that can be manipulated at runtime are referred to as 'Snapshot-capable' controls – a term that is used to indicate that a particular control can be set up in Venue Manager as participating in Snapshots.

A Snapshot is a set of control data “values” for any number of runtime-modifiable and Snapshot-capable controls for a particular Configuration. In its default state, Venue Manager loads every Configuration with a special immutable Snapshot entitled 'original loaded settings'. This Snapshot, which is designated Snapshot number 0, holds a

Managing BASIS via 3rd Party Control Systems

complete set of control values for the entire Configuration and acts as a baseline setting if the user should want to return the device to a known state (as originally programmed).

Each Configuration can have an arbitrary number of Snapshots associated with it. The number is limited only by the size of the Snapshots and the available Snapshot memory in the BASIS for a given Configuration. The number of Snapshots per Configuration that can be stored is dependent upon the number of Snapshot-capable controls in each Snapshot, but could potentially number in the tens or hundreds. A complete list of Snapshot-capable objects is included in Appendix B at the end of this document.

All Configurations have at least one Snapshot and they may have as many as are supported based on the limits discussed above. Therefore, an active Configuration could have any number of sets of predefined control values for any Snapshot-capable control(s) in the BASIS.

The Config/Snapshot recall commands can be used to manage which of the available Snapshots for the active Configuration is in the “active Snapshot” role. Note that Snapshots can only vary in certain control values so as to maintain the continuity of the signal chain. This restriction is managed during the setup process through the notion of Snapshot-capable controls and is mostly transparent to the user. By enforcing this restriction, there is a guarantee of complete signal chain continuity between all Snapshots associated with the active Configuration.

Switching between Snapshots in a BASIS occurs without muting of the audio. That’s the whole idea... to be able to make changes (possibly many changes) to live device with a single command without affecting the audio signal chain. When switching between Snapshots, the control values are smoothed during the transition from one Snapshot to the other so that even large changes in values occur with a minimum of audio distortion. A Snapshot change can be thought of, as being similar to tweaking the knobs on a lot of analog gear while the wiring remains the same. See the Venue Manager documentation for a complete discussion of Snapshots.

Original Loaded Settings

The Snapshot entitled ‘Original Loaded Settings’ has special meaning to a BASIS device. The “Original Loaded Settings” Snapshot contains ‘starting’ values for all of the Snapshot-capable controls and it guarantees that the device can initialize and perform in an expected manner.

It is important to understand the ‘Original Loaded Settings’ are NOT “set in stone” – they will be updated ANYTIME a Config is copied (“flashed” or Sync’d) from Design to Live. At that instant, ‘Original Loaded Settings’ are now the most recent settings as defined by/during the Design mode. Any further adjustments to the Design settings and subsequent Syncs will further update ‘Original Loaded Settings’. DO NOT RELY on Original Loaded Settings to always be a set of default values. The user is advised to create/define a Snapshot which contains ALL of the possible parameters, and storing it with the name “Default Settings” or “Master Reset” or “As Designed” or similar.

The Original Loaded Settings Snapshot (Snapshot 0) is the active Snapshot when the BASIS contains no user-defined Snapshots or if Snapshot 0 is manually recalled by a third party control system. Manipulation of Snapshot 0 is not permitted through applications in the QSControl.net software suite... this includes Venue Manager. However, it is possible to manipulate Snapshot 0 via commands issued from a third party control system. We recommend avoiding this. Should any or all Snapshot-capable controls require manipulation through a third party control system, we recommend that new user-defined Snapshots be created for this purpose.

It is important that the designer grasp the concept of these “Original Loaded Settings”. It is up to the designer to either create appropriate user-defined Snapshots for their application or to detect a BASIS reset event and reload any Snapshot-capable control values that differ from the defaults in Snapshot 0 if no user-defined Snapshots are created.

Snapshot Behavior

A Snapshot is a “set” of related control values that are applied to the BASIS when the Snapshot is recalled. This simply means that “all” of the control values that are contained in the Snapshot are used as the new values for the associated controls and they are applied to the BASIS all at once.

Managing BASIS via 3rd Party Control Systems

Only one Snapshot may be applied to a BASIS (being active in the device) at one time. The specific control values in each Snapshot are manipulated to match the value provided by the Snapshot when the Snapshot is recalled. All other controls in a BASIS (all other Configuration objects in the active Configuration) that are not part of the Snapshot retain their present value – the value they had before the Snapshot was recalled. This includes any casual control changes made through Venue Manager or through a third party control system. So if one were to make changes to a gain setting that was not part of the Snapshot being recalled, the gain setting would not be altered. However, if the gain setting were part of the Snapshot then the gain setting’s prior value would be replaced with the value specified in the Snapshot.

This is a departure from the original behavior of Snapshots in BASIS. With BASIS firmware versions 1.x.x, all Snapshots include all of the controls in a Configuration. Therefore, a Snapshot recall in these older versions affects all controls. This is an inefficient design and it severely limits the quantity of Snapshots that can be stored in the device. In order to make room for additional Snapshots, and to provide more granular control of Snapshot content, BASIS 2.x.x and 3.x.x firmware releases allow the user to define which controls participate in each Snapshot. Of course if the user includes all Snapshot-capable controls in a Snapshot then the behavior is exactly the same as it is in the 1.x.x firmware versions.

The ability to independently determine which controls are stored in a Snapshot can give the user a lot of flexibility when setting up a venue. For instance, observe the behavior that occurs when a series of Snapshots are applied in one sequential order versus another.

State/Action	Example of Controls in BASIS (gray denotes no value change in Snapshot)					
	Mute 1	Level 1	Mute 2	Level 2	Delay 1	Delay 2
Original Values	Mute	Mute	Mute	-12dB	10ms	10ms
Apply SS#1	Unmute		Unmute	0dB	100ms	
Apply SS#2		Unmute	Mute		10ms	
Apply SS#3	Mute	Mute	Mute	-24dB	50ms	100ms
Apply SS#4		Unmute				25ms
Resulting Values	Mute	Unmute	Mute	-24dB	50ms	25ms

The sequence above of a Snapshot recall and the resulting control value changes are quite different than the sequence (and results) shown below. Since the Snapshots do not contain the same set of controls, the final results of the Snapshot recall are order-dependent, even though all the same Snapshots are being used.

State/Action	Example of Controls in BASIS (gray denotes no value change in Snapshot)					
	Mute 1	Level 1	Mute 2	Level 2	Delay 1	Delay 2
Original Values	Mute	Mute	Mute	-12dB	10ms	10ms
Apply SS#3	Mute	Mute	Mute	-24dB	50ms	100ms
Apply SS#1	Unmute		Unmute	0dB	100ms	
Apply SS#2		Unmute	Mute		10ms	
Apply SS#4		Unmute				25ms
Resulting Values	Unmute	Unmute	Mute	0dB	10ms	25ms

What is illustrated is that unless all Snapshots are set up to cover all Snapshot-capable control values, the device could achieve an end state that is different simply depending upon the order of the Snapshots that are recalled. Clearly, if the user intended that a recall of Snapshot 4 should always result in the first result (for instance, that of ‘Mute 1’ being in the mute state) then Snapshot 4 must include ‘Mute 1’ (and lose any casual changes made to ‘Mute 1’), and thus return the device to a defined state – that of Snapshot 4.

The real upside of this model is that it is possible to craft subsets that act independently of any casual changes made to the device. Thus, it is possible to have Snapshots specifically target a group of controls, say the mute or level of a microphone input, without affecting anything else. This can be quite useful, and economic of device storage and processing time during Snapshot recall.

The other benefit is that Snapshots can be constructed to specifically exclude controls that are to be manipulated via third party controllers, thus giving complete precedence to the controller’s settings. In the example below, ‘Delay 2’ is not included in any Snapshots (other than the ‘original loaded settings’ Snapshot) so that its’ value can be determined solely by the value sent by the control device.

Managing BASIS via 3rd Party Control Systems

The only times when the BASIS will set the control values independently of the control device in this scenario are a) upon BASIS reboot or b) manual recall of the ‘original loaded settings’ Snapshot.

State/Action	Example of Controls in BASIS (gray denotes no value change in Snapshot)					
	Mute 1	Level 1	Mute 2	Level 2	Delay 1	Delay 2
Original Values	Mute	Mute	Mute	-24dB	10ms	10ms
Apply SS#3	Mute	Mute	Mute	0dB	50ms	
Control Device						50ms
Apply SS#1	Unmute	Mute	Unmute		100ms	
Apply SS#2		Unmute	Mute		10ms	
Control Device						100ms
Apply SS#4		Unmute				
Resulting Values	Unmute	Unmute	Mute	0dB	10ms	100ms

In the above, the ‘Delay 2’ value is not in any of the Snapshots, is set by the external controller, and thus retains the last setting that the control device sent. The one Snapshot that will override the controller setting is the ‘original loaded settings’ Snapshot, which in this case will revert the ‘Delay 2’ time to 10ms.

Device Parameters

All BASIS devices have device setup information that remains the same regardless of the Configuration and Snapshot that are active in the device. Some examples are the device name and IP address – items that shouldn’t change during device use.

Some device parameter changes, such as changing an IP address of a BASIS, may require a reset of the device to take place. It is not expected that device parameter changes would occur outside of system setup.

BASIS Communication

The BASIS communicates with third-party control devices using TCP/IP. A dedicated port is provided for third party communications - Port 4446 (decimal). To communicate with the BASIS device, a client TCP/IP connection must be established on this port. The port is kept open for the entire duration of the control session, as long as data is sent from the control device to the BASIS every 2 seconds or less. The TCP connection will be closed by the BASIS if it is not kept open by the control device’s activity.

In the event that the controller has no activity to transmit to the BASIS, a ‘keep-alive’ message is provided so that the controller can keep the connection to the BASIS open.

The payload to this port is an XML string that will may: a) request the current value of a device parameter, b) set the parameter with a new value, c) register the parameter for automatic update notification, or d) enable metering. Each of the message types is covered in detail later in the document.

To establish a session with the BASIS, it is necessary to supply a password. The password that controls all third party control devices is the **Front Panel Password**, which is accessed through the General tab of the BASIS setup screen in Venue Manager. This is also known as the Level One Password (L1PW).

Each message sent to the BASIS must contain a valid password to be processed. That password must be the Front Panel Password that is accessible through the main device panel for the BASIS within Venue Manager.

There are three other places where a password can be entered within Venue Manager, and they do not affect the communications with third party control devices. Only the front panel password affects the third party control interface to the BASIS.

Restrictions

The control device, so as to keep its UI up-to-date, must do one of the following:

- Poll the control values in the BASIS periodically,
- Register for automatic control value updates from the BASIS, or
- Force the control values into the BASIS periodically.

Managing BASIS via 3rd Party Control Systems

The primary reason for selecting one method over the other is that some control devices have limited processing ability.

Polling, depending upon implementation, can be CPU-intensive, but it allows the device to self-regulate the amount of traffic to and from itself. For control devices with limited resources, this can be a good way to make sure the CPU and/or limited memory resources do not get overrun.

Registration allows the device to get unsolicited control value change notifications, but can result in high peak data rates that some controllers may not be able to handle. For control devices with a lot of available CPU and memory, this is a good option. The key-determining factor is whether the controller can service the messages during a peak load fast enough to avoid a message overrun.

Forcing the control values to the BASIS (open loop) can be very simple to implement. The least capable controllers can use this method to avoid a lot of incoming message processing. However, this method is based upon the assumption that the control value being sent are actually applicable to the BASIS. It is important to make sure that the BASIS will always be in a state where the control values that are blindly being sent will not cause a problem. Having all Configurations and Snapshots in a compatible/same signal flow state would be a very reasonable start to assuring that the assumption holds true.

In addition, the TCP/IP connection must adhere to the Maximum Segment Size (MSS) returned by the BASIS. This ensures that the commands will function properly, regardless of media type.

Control Gating

In addition to limiting poll rates and in general being a good network citizen, QSC strongly recommends that all third party control devices transmit control updates at no faster than a 50ms period. This 'gating' is to avoid a 'data flood' – a condition where a physical control can generate many small incremental changes back-to-back at a rate that is less than the turn-time of a control update.

As an example, take a control surface with a slider, which is set up to send control value changes to a pair of output level controls for a stereo pair. The control surface should be set up so that a data change at the slider triggers an event to occur no faster than every 50ms while change is detected at the slider. This is in contrast to the simple approach of forwarding all slider changes to the device. The reason for this is that a slider can easily generate several hundred messages in a second, well above the supported data rate.

By limiting continuous control changes with a 50ms gate, the third party controller can ensure that the BASIS can service the requests in a timely manner and make sure there is only one control change pending at any given time. The result is even, predictable response from the BASIS, and a 'tight' connection between the controller and the BASIS. For reference, this is exactly what QSCControl.NET (Venue Manager) does.

Failure to gate control changes can result in more changes being sent per unit time than can be serviced by the BASIS, resulting in a backlog of change requests. This results in a 'loose', or 'spongy' connection between the controller and the BASIS and can eventually lead to a message buffer overrun in the BASIS.

Over a period of time, flooding the BASIS with non-gated control changes can cause the BASIS to reset due to a TCP input buffer overrun.

Polling

When polling the BASIS for data, the polling rate should not exceed what is minimally required by the UI to maintain a reasonable visible consistency with what is being heard in the audio chain. QSC recommends that control polls be issued at no faster than a 5ms per control rate, and a 100ms per UI page rate. The reason for this is that if the device polls continuously, it will saturate the network with traffic, possibly adversely affecting any other Ethernet devices that are connected to the network.

Note that with large numbers of controls on a UI, it may not be possible to achieve the 100ms per page rate. A page with 50 controls can't achieve a 100ms per page polling rate because doing so would violate the 5ms per control

Managing BASIS via 3rd Party Control Systems

limit, or would poll the page at a rate that was less than the time it takes to service all of the controls (250ms minimum). In this case, a better solution would be control registration.

Polling can be an ideal solution for those devices that have limited processing capability, and/or can't service interrupts in a timely manner. The amount of data to be processed can be limited by the controlling device's polling rate.

Control Registration

An alternative to control polling is control registration. Control registration frees the controller from having to maintain lists of controls to poll, and reduces the average processing overhead involved in keeping the UI up-to-date. However, the peak processing can be quite great, depending upon the number of controls that are registered.

Control registration works like this: The third party controller sends a registration message to the BASIS, specifying the control(s) to be registered. Later, when a registered control is changed – from whatever source - the BASIS notifies all registered clients of the changing control of the value change.

The BASIS continues providing control notification until the active Configuration running in the BASIS is changed to another Configuration or the TCP connection is broken. In both of these instances, the controls must be re-registered by each client that requires control updates.

Control registration is ideal for high-bandwidth controllers like embedded or panel PC's and many of the currently available dedicated touch panel controllers. The main criterion in determining if control registration is the best for a particular situation is the controller's peak processing capability and how it manages potentially large bursts of high data-rate information. This becomes even more important when multiple BASIS units are being managed from one controller and the number of registered controls is high.

Forcing Control Values

A last alternative available for a controller is to simply force the same data to a BASIS control on a periodic BASIS. The expectation in this case is that synchronization between the UI and the BASIS will be assured by repetition.

Casual control changes made through another interface – Venue Manager or another controller – would be overridden. This can lead to behavior at the other UI's that would not be obvious upon first glance. However, this approach is easy to implement and lightweight.

XML Command Overview

In the following examples, the XML is broken by line breaks for readability. The XML that is sent to the BASIS cannot have carriage returns or linefeeds embedded in it. The BASIS will not process XML with embedded carriage returns and linefeeds.

There are two basic command syntax discussed in this document. The 'simple get' (usually just called the 'GET') command is used to retrieve data from a control in a BASIS, and the 'simple set' (usually just called the 'SET') command is used to change a value in a control in a BASIS. These commands are directed towards objects that, in the BASIS, are implemented as simple control objects.

There is a variation of the normal 'GET' and 'SET' commands which is called a 'complex get' or 'complex set', which is a get or set which is directed towards a compound or complex object in the BASIS. The Config/Snapshot commands are complex get/set commands because the Config/Snapshot manager in the BASIS is a complex control object.

The important thing to remember about simple and complex objects is that they aren't the same, and each object type requires a command tailored to its object type. The specific documentation for each command will provide everything a programmer will need to be able to construct a suitable command of the correct command type.

An example BASIS XML 'simple get' message is shown below. Carriage returns and linefeeds are shown for readability, but are not in the actual message.

```
<?xml version="1.0"?>
<!--0000109-->
<GET_S RT="C00" ID="0" L1PW="password">
<EGS O="0x01120006" M="0"/>
</GET_S>
```

A breakdown of the above XML message shows the elements that are common to all messages sent to a BASIS and those that will vary, based upon the command being issued by the control device.

Managing BASIS via 3rd Party Control Systems

XML Message: Example 'Get'	
<i>Description:</i> This is a breakdown of a typical XML 'GET' message that is sent to a BASIS. Note that the elements that are common to all commands are marked as such.	
Text	Description
<?xml version="1.0"?>	XML header string. Constant for all messages.
<!--0000108-->	Message length comment field. The number appearing between the "--" delimiters must be set to the decimal length of the entire XML message, and must use leading zeros for a total of 7 numerals (ie: a length of 32 is '<!--000032-->'). Used for all commands.
<GET_S	The 'GET' header. This will vary by type of command. The preceding bracket indicates the start of an XML tag.
RT="C00"	Message routing parameter. Constant for all BASIS commands. This is an attribute of the 'GET_S' tag.
ID="0"	Message ID parameter. Can be any string up to 20 characters in length. Not required, and can be left as a zero. If left as a zero, the IP address is used by the BASIS for message ID. This is an attribute of the 'GET_S' tag.
L1PW="password">	The level one password for the BASIS. This must match the password programmed into the BASIS through the ' Front Panel Password ' option in the "General" tab in Venue Manager. This password is used to confirm command legitimacy. Used for all commands. This is an attribute of the 'GETACK_S' tag. The trailing bracket closes the 'GET_S' tag.
<EGS	The 'GET' command for a parameter in the BASIS. This will vary by type of command. The preceding bracket indicates the start of the 'EGS' tag. Since no ending tag for 'GET_S' has occurred, the 'EGS' tag is a child of the 'GET_S' tag.
O="0x01120006" '—this is the letter O, not a zero	The 'OID' number for the parameter being retrieved from the BASIS, in HEX format. This will vary, based upon the parameter being queried. This is an attribute of the 'EGS' tag.
M="0"/> '—this is a zero, not the letter O	Method parameter. Constant for all commands. The trailing bracket closes the 'EGS' tag.
</GET_S>	The 'end of get' trailer. This will vary by type of command. The closing tag for 'GET_S'.

Next is an example of the XML message the BASIS would send in response to the above 'GET' message. This is termed a 'get acknowledgement' from the BASIS. Carriage returns and linefeeds are shown for readability, but are not in the actual message.

```
<?xml version="1.0"?>
<!--0000114-->
<GETACK_S RT="C00" ID="0">
<EGS O="0x01120006" R="0" M="0" B="true"/>
</GETACK_S>
```

A breakdown of the above XML message shows the elements that are common to all acknowledgement messages sent from a BASIS and those that will vary, based upon the command being issued by the control device.

Managing BASIS via 3rd Party Control Systems

XML Message: Example 'Get Acknowledgement'	
<i>Description:</i> This is a breakdown of a typical XML 'get acknowledgement' message that is sent from a BASIS in response to a 'GET' message from the control device. Note that the elements that are common to all commands are marked as such.	
Text	Description
<?xml version="1.0"?>	XML header string. Constant for all messages.
<!--0000131-->	Message length comment field.
<GETACK_S	The 'get acknowledgement' header. This will vary by type of command. The preceding bracket indicates the start of an XML tag.
RT="C00"	Message routing parameter. Constant for all BASIS commands. This is an attribute of the 'GETACK_S' tag.
ID="0"	Message ID parameter. Same as the 'GET' that caused the acknowledgement. This is an attribute of the 'GETACK_S' tag.
L1PW="password">	The level one password from the 'GET' message. This is an attribute of the 'GETACK_S' tag. The trailing bracket closes the opening 'GETACK_S' tag.
<EGS	The 'GET' command for a parameter in the BASIS. This will vary by type of command. The preceding bracket indicates the start of the 'EGS' tag. Since no ending tag for 'GET_S' has occurred, the 'EGS' tag is a child of the 'GET_S' tag.
O="0x01120006" '—this is the letter O, not a zero	The 'OID' number for the parameter being retrieved from the BASIS, in HEX format. This will vary, based upon the parameter being queried. This is an attribute of the 'EGS' tag.
R="0" '—this is a zero, not the letter O	Method response parameter. If the message succeeded, the response is set to 0. Any other value indicates message failure. This is an attribute of the 'EGS' tag.
M="0" '—this is a zero, not the letter O	Method parameter. Constant for all commands. This is an attribute of the 'EGS' tag.
B="true"/>	Value parameter. The data type for the OID requested was Boolean ('B'), and the value of the OID was "true". This will vary, based upon the OID requested. This is an attribute of the 'EGS' tag. The trailing bracket closes the 'EGS' tag.
</GETACK_S>	The 'end of get acknowledgement' trailer. This will vary by type of command. The trailing bracket closes the 'GETACK_S' tag.

The above pair of examples shows both sides of the conversation between a BASIS and a control device for a 'GET'. Other XML commands are handled similarly, in that every command issued by the control device will generate a response from the BASIS.

Until a response is returned from the BASIS, no assumptions can be made as to the success of a message. If the controlling device will show a value in its UI, it should show the value returned from the BASIS' acknowledgement of the message, not the presumption of a messages' success. Always check return values for a value of 0 (zero).

"R" Parameter	NAME	MEANING
0	OK	It worked !
1	FAIL	Ooops..
2	OBJECT NON-EXISTANT	Object being addressed does not exist
3	SETPOINT_TOO_SMALL	Given setpoint value too small. No action taken
4	SETPOINT_TOO_BIG	Given setpoint value too large. No action taken.
5	REQUEST_INVALID	Request invalid in current context. No action taken.
6	READ_ONLY	Attempt to change value of read-only property.
7	OPERATION_ABORTED	Operation in progress aborted by external request.
8	OBJECT_BUSY	Object is doing something and cannot fulfill request..
9	OBJECT_LOCKED	Object is locked by some controller and is unavailable.
10	OBJECT_DISABLED	Object has been disabled by device and cannot honor requests.
11	PROCESSING_ERROR	A processing error has occurred within the addressed object.
12	DSP_ALLOCATION_EXCEEDED	Requested DSP signal flow exceeds DSP processing capability
13	DSP_FEEDBACK_LOOP	A feedback loop has been found in requested signal flow. Not allowed
14	DSP_UNCONNECTED_INPUT	An unconnected input has been found in requested signal flow. Not allowed
15	DSP_UNCONNECTED_OUTPUT	An unconnected output has been found in requested signal flow. Not allowed
16	DSP_DUPLICATE_OUTPUT	More than 1 DSP output is driving 1 input. Not allowed
17	DSP_ROUTE_LIMIT_EXCEEDED	Signal flow routing not implementable by DSP hardware
18	DSP_INVALID_PARAMETER	DSP processor parameter is out of range.

Commands and Controls

To make it easier for the third party developer, the following commands are discussed in the context of the controls that are to be affected. In this manner, exact messages can be shown for reference.

Mute Commands

The mute controls use the ‘GET’ and ‘SET’ command structure. The mute controls are the same controls that reside on the “DSP Out” and “DP Out” tabs of Venue Manager, and are located at the end of the DSP chain, but before the analog or CobraNet outputs. The OIDs used in the commands to get and set the mute status of the outputs of a BASIS are shown below (note the OIDs are “zero based indexed”):

Physical Port Number	Port Type	OID
Output 1	Dataport A – Channel 1	0x01120000
Output 2	Dataport A – Channel 2	0x01120001
Output 3	Dataport B – Channel 1	0x01120002
Output 4	Dataport B – Channel 2	0x01120003
Output 5	Dataport C – Channel 1	0x01120004
Output 6	Dataport C – Channel 2	0x01120005
Output 7	Dataport D – Channel 1	0x01120006
Output 8	Dataport D – Channel 2	0x01120007
Logical (DSP) Port Number	Port Type	
Output 9	Digital Output to CobraNet	0x01120008
Output 10	Digital Output to CobraNet	0x01120009
Output 11	Digital Output to CobraNet	0x0112000A
Output 12	Digital Output to CobraNet	0x0112000B
Output 13	Digital Output to CobraNet	0x0112000C
Output 14	Digital Output to CobraNet	0x0112000D
Output 15	Digital Output to CobraNet	0x0112000E
Output 16	Digital Output to CobraNet	0x0112000F
Output 17	Digital Output to CobraNet	0x01120010
Output 18	Digital Output to CobraNet	0x01120011
Output 19	Digital Output to CobraNet	0x01120012
Output 20	Digital Output to CobraNet	0x01120013
Output 21	Digital Output to CobraNet	0x01120014
Output 22	Digital Output to CobraNet	0x01120015
Output 23	Digital Output to CobraNet	0x01120016
Output 24	Digital Output to CobraNet	0x01120017

On BASIS922az devices, the first eight output ports in the DSP chain are the Dataport outputs, or euro (analog) outputs on the back of the BASIS522aa device, which can also be routed to CobraNet. The remaining DSP outputs can ONLY be routed to CobraNet.

The number of analog, Dataport and/or CobraNet input and output channels supported by a particular BASIS device will vary by model. *For specific details on the number of Dataport and CobraNet outputs, please consult the product Hardware Manual for the specific BASIS model being used.*

Mute GET Command

To get the value of a mute button for a particular output port, the GET command is issued. An example get for a mute follows. Carriage returns and linefeeds are shown for readability, but are not in the actual message.

```
<?xml version="1.0"?>
<!--nnnnnnnn-->
<GET_S RT="C00" ID="0" L1PW="password">
<EGS O="0x011200cc" M="0"/>
</GET_S>
```

There are three portions of the string that must be dynamically created:

- The first (represented by “*nnnnnnnn*”) is the length of the entire message.
- The second (represented by “*password*”) is the Level 1 (Front Panel) password for the BASIS unit.
- The third (represented by “*cc*”) is the channel number (in hex format) that is being adjusted. The table on the previous page lists all possible output mute OIDs.

The response will have both the response code (as indicated by the “R” value) and a Boolean return (as indicated by the “B” value) that indicates the current value of the Mute setting. Only one EGS (Element Get Simple) node is allowed per message.

Mute GET Command Example

Carriage returns and linefeeds are shown for readability, but are not in the actual message.

Get command example for output 6:

```
<?xml version="1.0"?>
<!--0000109-->
<GET_S RT="C00" ID="0" L1PW="password">
<EGS O="0x01120005" M="0"/>
</GET_S>
```

Get command response for above message (assuming the output is currently muted):

```
<?xml version="1.0"?>
<!--0000114-->
<GETACK_S RT="C00" ID="0">
<EGS O="0x01120005" R="0" M="0" B="true"/>
</GETACK_S>
```

Mute SET Command

To set the value of a mute button for a particular output port, the SET command is issued. An example set for a mute follows. Carriage returns and linefeeds are shown for readability, but are not in the actual message.

```
<?xml version="1.0"?>
<!--nnnnnnnn-->
<SET_S RT="C00" ID="0" L1PW="password">
<ESS O="0x011200cc" M="0" B="value"/>
</SET_S>
```

There are four portions of the string that must be dynamically created:

- The first (represented by “*nnnnnnnn*”) is the length of the entire message.

Managing BASIS via 3rd Party Control Systems

- The second (represented by “*password*”) is the Level 1 (Front Panel) password for the BASIS unit.
- The third (represented by “*cc*”) is the channel number (in hex format) that is being adjusted. The previous table lists all possible output mute OIDs.
- The fourth (represented by “*value*”) is the value for the new setting. In the case of a Mute, this will be either “true” or “false”.

The response will have both the response code (as indicated by the “R” value – see following example) and a Boolean return (as indicated by the “B” value) that indicates the new value of the Mute setting. Only one ESS (Element Set Simple) node is allowed per message.

Mute SET Command Example

Carriage returns and linefeeds are shown for readability, but are not in the actual message.

Set command example for channel 10:

```
<?xml version="1.0"?>
<!--0000119-->
<SET_S RT="C00" ID="0" L1PW="password">
<ESS O="0x01120009" M="0" B="true"/>
</SET_S>
```

Set command response for above message:

```
<?xml version="1.0"?>
<!--0000114-->
<SETACK_S RT="C00" ID="0">
<ESS O="0x01120009" R="0" M="0" B="true"/>
</SETACK_S>
```

Managing BASIS via 3rd Party Control Systems

Gain Commands

The gain level of an amplifier attached to a BASIS Dataport can be changed remotely. The gain controls are the same controls that reside on the “Levels” tab of Venue Manager, and are located at the end of the DSP chain, but before the analog or CobraNet outputs.

The gain controls use the ‘GET’ and ‘SET’ command structure. The OIDs used in the commands to get and set the gain of the outputs of a BASIS are shown below (note the OIDs are “zero based indexed”):

Physical Port Number	Port Type	OID
Output 1	Dataport A – Channel 1	0x01110000
Output 2	Dataport A – Channel 2	0x01110001
Output 3	Dataport B – Channel 1	0x01110002
Output 4	Dataport B – Channel 2	0x01110003
Output 5	Dataport C – Channel 1	0x01110004
Output 6	Dataport C – Channel 2	0x01110005
Output 7	Dataport D – Channel 1	0x01110006
Output 8	Dataport D – Channel 2	0x01110007
Logical (DSP) Port Number	Port Type	
Output 9	Digital Output to CobraNet	0x01110008
Output 10	Digital Output to CobraNet	0x01110009
Output 11	Digital Output to CobraNet	0x0111000A
Output 12	Digital Output to CobraNet	0x0111000B
Output 13	Digital Output to CobraNet	0x0111000C
Output 14	Digital Output to CobraNet	0x0111000D
Output 15	Digital Output to CobraNet	0x0111000E
Output 16	Digital Output to CobraNet	0x0111000F
Output 17	Digital Output to CobraNet	0x01110010
Output 18	Digital Output to CobraNet	0x01110011
Output 19	Digital Output to CobraNet	0x01110012
Output 20	Digital Output to CobraNet	0x01110013
Output 21	Digital Output to CobraNet	0x01110014
Output 22	Digital Output to CobraNet	0x01110015
Output 23	Digital Output to CobraNet	0x01110016
Output 24	Digital Output to CobraNet	0x01110017

On BASIS922az devices, the first eight output ports in the DSP chain are designed to correspond to the Dataport outputs, or euro (analog) outputs on the back of the BASIS522aa device, which can also be routed to CobraNet. The remaining DSP outputs are designed to correspond to the outputs that can route to CobraNet.

The number of analog, Dataport and/or CobraNet input and output channels supported by a particular BASIS device will vary by model. *For specific details on the number of Dataport and CobraNet outputs, please consult the product Hardware Manual for the specific BASIS model being used.*

Gain GET Command

To get the value of a gain setting for a particular output port, the GET command is issued. An example get for a gain setting follows. Carriage returns and linefeeds are shown for readability, but are not in the actual message.

```
<?xml version="1.0"?>
<!--0000000-->
<GET_S RT="C00" ID="0" L1PW="password">
<EGS O="0x011100cc" M="0"/>
</GET_S>
```

There are three portions of the string that must be dynamically created:

- The first (represented by “0000000”) is the length of the entire message.
- The second (represented by “password”) is the Level 1 (Front Panel) password for the BASIS unit.
- The third (represented by “cc”) is the channel number (in hex format) that is being adjusted. The table above lists all possible output mute OIDs.

The response will have both the response code (as indicated by the “R” value) and a floating point return (as indicated by the “F” value) that indicates the current value of the Gain setting. Only one EGS (Element Get Simple) node is allowed per message.

Gain GET Command Example

Carriage returns and linefeeds are shown for readability, but are not in the actual message.

Get command example for channel 3:

```
<?xml version="1.0"?>
<!--0000109-->
<GET_S RT="C00" ID="0" L1PW="password">
<EGS O="0x01110002" M="0"/>
</GET_S>
```

Get command response for above message:

```
<?xml version="1.0"?>
<!--0000115-->
<GETACK_S RT="C00" ID="0">
<EGS O="0x01110002" R="0" M="0" F="-80.5"/>
</GETACK_S>
```

Gain SET Command

To set the value of a gain setting for a particular output port, the SET command is issued. An example set for a gain setting follows. Carriage returns and linefeeds are shown for readability, but are not in the actual message.

```
<?xml version="1.0"?>
<!--0000000-->
<SET_S RT="C00" ID="0" L1PW="password">
<ESS O="0x011100cc" M="0" F="value"/>
</SET_S>
```

There are four portions of the string that must be dynamically created:

- The first (represented by “0000000”) is the length of the entire message.
- The second (represented by “password”) is the Level 1 (Front Panel) password for the BASIS unit.
- The third (represented by “cc”) is the channel number (in hex format) that is being adjusted.
- The fourth (represented by “value”) is the value for the new setting. In the case of a Gain, this will be values from “12.0” to “-100.0” in 0.1db increments.

The response will have both the response code (as indicated by the “R” value) and a float return (as indicated by the “F” value) that indicates the new value of the Gain setting. Only one ESS (Element Set Simple) node is allowed per message. A response code of 4 (R=“4”) indicates the Set value sent is out of range for this OID and no change in previous value occurred.

Gain SET Command Example

Carriage returns and linefeeds are shown for readability, but are not in the actual message.

Set command example for channel 11 to change to a value of -12.5:

```
<?xml version="1.0"?>
<!--0000119-->
<SET_S RT="C00" ID="0" L1PW="password">
<ESS O="0x0111000a" M="0" F="-12.5"/>
</SET_S>
```

Set command response for above message:

```
<?xml version="1.0"?>
<!--0000115-->
<SETACK_S RT="C00" ID="0">
<ESS O="0x0111000a" R="0" M="0" F="-12.5"/>
</SETACK_S>
```

Config/Snapshot Commands

The Config/Snapshot commands allow the third-party controller to change the BASIS unit's active Configuration and/or Snapshot by recalling a different Config and/or Snapshot within a Config.

Recalling a different Configuration can change the device dramatically, and will cause a muting of the audio because there are no assurances that the old and new Configurations will be anything like one another, signal path-wise. Changing the Configuration is best thought of as rewiring the signal flow – essentially reprogramming the BASIS. Nothing is guaranteed to be the same except the IP address and other data that relates to the identity of the unit.

Recalling a different Snapshot can change the audio in the BASIS less dramatically, since every Snapshot within a Configuration is based upon the Configuration itself. As such, there is a guarantee that the signal chain in the device is substantially the same between all Snapshots for a particular Configuration. Recalling a Snapshot can be thought of as a grouped change of parameters to be applied to the active Configuration. There is no muting between Snapshot changes unless, of course, the recalled Snapshot includes a muted parameter.

Control	OID
Active Configuration Number	0x00070000
Active Snapshot Number	0x00070100
Active Configuration Name	0x00070200
Active Snapshot Name	0x00070300

All BASIS devices have only one active Configuration at a time and one active Snapshot at a time within that Configuration. All BASIS devices have eight Configurations that can be brought to the active/audible state through the Config/Snapshot commands. Each Configuration can have as few or as many Snapshots as the Configuration's flash memory allocation will allow. Each Snapshot for the active Configuration can be made active through the Config/Snapshot commands.

Config/Snapshot GET Command

To get the value of the active Configuration and Snapshot controls, the 'get complex' command is issued. An example get command for active Configuration and Snapshot follows. Carriage returns and linefeeds are shown for readability, but are not in the actual message.

```
<?xml version="1.0"?>
<!--000000-->
<GET_C RT="C00" ID="0" L1PW="password">
<EGC O="0x00070000" M="0"/>
<EGCP O="0x00070000"/>
<EGCP O="0x00070100"/>
<EGCP O="0x00070200"/>
<EGCP O="0x00070300"/>
</GET_C>
```

This command will return the current Configuration and Snapshot settings. There are two portions of the string that must be dynamically created:

- The first (represented by "000000") is the length of the entire message.
- The second (represented by "password") is the Level 1 (Front Panel) password for the BASIS unit.

The response will have both the response code (as indicated by the "R" value) and a set of values that give the current Configuration and Snapshot numbers (as indicated by the "S" values) as well as their names (as indicated by the "ST" values). The Configuration numbers and Snapshot numbers are all zero relative. The Snapshot numbered 0 is the 'original loaded settings' Snapshot, and Snapshots numbered 1 to N are the user-defined Snapshots.

Managing BASIS via 3rd Party Control Systems

The EGC tag (Element Get Complex) is used to identify a group of related settings. Each control that is requested from the group needs its own EGCP (Element Get Complex Parameter) tag to request the data. The below example shows all requests for all of the available controls from this group, however, a subset of the controls can be requested if only one or two control values are required.

Config/Snapshot GET Command Example

Carriage returns and linefeeds are shown for readability, but are not in the actual message.

Get command example for Config/Snapshot numbers and names:

```
<?xml version="1.0"?>
<!--0000197-->
<GET_C RT="C00" ID="0" "password">
<EGC O="0x00070000" M="0"/>
<EGCP O="0x00070000"/>
<EGCP O="0x00070100"/>
<EGCP O="0x00070200"/>
<EGCP O="0x00070300"/>
</GET_C>
```

Get command response for above message:

```
<?xml version="1.0"?>
<!--0000259-->
<GETACK_C RT="C00" ID="0">
<EGC O="0x00070000" R="0" M="0"/>
<EGCP O="0x00070000" R="0" S="1"/>
<EGCP O="0x00070100" R="0" S="0"/>
<EGCP O="0x00070200" R="0" ST="BigGain"/>
<EGCP O="0x00070300" R="0" ST="Snapshot #1"/>
</GETACK_C>
```

Config/Snapshot SET Command

To set the value of the active Configuration and Snapshot numbers, the ‘Element Set Complex’ (ESC) command is issued. An example SET for active Configuration and Snapshot follows. Note that ESC is not the "escape" character!

The Configuration and Snapshot number may be set at the same time, within one complex message. Carriage returns and linefeeds are shown for readability, but are not in the actual message.

```
<?xml version="1.0"?>
<!--0000000-->
<SET_C RT="C00" ID="0" L1PW="password">
<ESC O="0x00070000" M="0"/>
<ESCP O="0x00070000" S="Config"/>
<ESCP O="0x00070100" S="Snapshot"/>
</SET_C>
```

This command will change the active Configuration and Snapshot controls to the values that are sent. There are four portions of the string that must be dynamically created:

- The first (represented by “0000000”) is the length of the entire message.
- The second (represented by “password”) is the Level 1 (Front Panel) password for the BASIS unit.
- The third (represented by “Config”) is the new Configuration number. Configurations are numbered 0 to 7 in BASIS.

Managing BASIS via 3rd Party Control Systems

- The fourth (represented by “*Snapshot*”) is the new Snapshot number. Snapshots are also numbered from 0 to N in BASIS, where 0 is the number of the ‘original loaded settings’ Snapshot and 1 to N are the user-defined Snapshot numbers for the Configuration.

The response will have a response code (as indicated by the “R” value) and a set of values that give the current Configuration and Snapshot numbers (as indicated by the "S" values) as well as their names (as indicated by the "ST" values).

The ESC tag (**E**lement **S**et **C**omplex) is used to identify a group of related settings. Each control that is to be set in the group needs its own ESCP (**E**lement **S**et **C**omplex **P**arameter) tag to set the value of the control. The below example demonstrates the correct method for changing a Configuration and Snapshot.

Both of these controls (Configuration/Snapshot) need not be passed together. It is possible to set just the Snapshot number but only for the currently Active Config. You may also set (recall) a Config without specifying any particular Snapshot. This is the same as choosing “None” in the Venue Manager application, and essentially recalls the Config with ALL of its most recent parameter values, whether part of the most recently recalled Snapshot or not.

NOTE: The Configuration and Snapshot name values are **Read-Only** and cannot be changed using these commands.

Config/Snapshot SET Command Example

Carriage returns and linefeeds are shown for readability, but are not in the actual message.

Set command example for Configuration 3 and 'original loaded settings' Snapshot 0:

```
<?xml version="1.0"?>
<!--0000168-->
<SET_C RT="C00" ID="0" L1PW="password">
<ESC O="0x00070000" M="0"/>
<ESCP O="0x00070000" S="2"/>
<ESCP O="0x00070100" S="0"/>
</SET_C>
```

Set command response for above message:

```
<?xml version="1.0"?>
<!--0000175-->
<SETACK_C RT="C00" ID="0">
<ESC O="0x00070000" R="0" M="0"/>
<ESCP O="0x00070000" R="0" UC="0"/>
<ESCP O="0x00070100" R="0" S="-1"/>
</SETACK_C>
```

Set command example for user-defined Snapshot 12 only:

```
<?xml version="1.0"?>
<!--0000138-->
<SET_C RT="C00" ID="0" L1PW="password">
<ESC O="0x00070000" M="0"/>
<ESCP O="0x00070100" S="12"/>
</SET_C>
```

Set command response for above message:

```
<?xml version="1.0"?>
<!--0000140-->
<SETACK_C RT="C00" ID="0">
<ESC O="0x00070000" R="0" M="0"/>
<ESCP O="0x00070100" R="0" S="12"/>
</SETACK_C>
```

Sensing and Controlling

The BASIS family of devices are equipped with an array of hardware that enables the remote sensing and control of devices that are connected to them. The following is a list of sensing and control options on most BASIS devices (the exact number of I/O features vary by model):

- Two relay outputs with N.O. and N.C. outputs
- 4 logic outputs
- 6 logic inputs

Since the BASIS family of devices are all Ethernet-based, this gives the installer a powerful tool for putting logic onto the Ethernet for remote control and sensing. Some examples are:

- Remote control of a motorized projection screen
- Remote control of lighting
- Remote sensing of a door's open/closed state
- Remote actuation of a relay
- Monitoring of equipment for loss prevention

The following are the commands that are provided to give access to the remote control and sensing hardware in the BASIS. *For specific details on the operational parameters of the contact closures, including electrical considerations, please refer to the product Hardware Manual for the specific BASIS model being used.*

Relay / Contact Closure Commands

The relay output / contact closure outputs are located on the back of the BASIS, and are used to effect external control of devices through the BASIS. The state of the relay outputs can be read and written, thus giving a way to transmit local (to the control surface) state change information over the Ethernet to a remote (to the control surface) contact controlled device that is proximate to the BASIS.

The relay output / contact closure controls use the 'GET' and 'SET' command structure. The OIDs that are used in the commands to get and set the closure status of the relay outputs of a BASIS are shown below. Note the OIDs are zero-base indexed.

For specific details on the operational parameters of the contact closures, including electrical considerations, please refer to the product Hardware Manual for the specific BASIS model being used:

Physical Port Number	Port Type	OID
Relay Output 1	Contact Closure	0x01180000
Relay Output 2	Contact Closure	0x01180001

On most BASIS devices there are two relay outputs. One model, the BASIS 914lz, has NO relay outputs.

Relay Output GET Command

To get the value of a particular relay output, the GET command is issued. An example GET for a relay output follows. Carriage returns and linefeeds are shown for readability, but are not in the actual message.

```
<?xml version="1.0"?>
<!--000000-->
<GET_S RT="C00" ID="0" L1PW="password">
<EGS O="0x011800cc" M="0"/>
</GET_S>
```

There are three portions of the string that must be dynamically created:

Managing BASIS via 3rd Party Control Systems

- The first (represented by “000000”) is the length of the entire message.
- The second (represented by “*password*”) is the Level 1 (Front Panel) password for the BASIS unit.
- The third (represented by “*cc*”) is the relay output number that is being adjusted. The table above lists all possible relay output OIDs.

The response will have both the response code (as indicated by the “R” value) and an unsigned character return value (as indicated by the “UC” value, limited to 0 for deenergized, or 1 for energized) that indicates the current value of the relay output. Only one EGS (Element Get Simple) node is allowed per message.

Keep in mind that the relay output can be wired as normally open or normally closed, so the return value does not necessarily reflect the electrical state of the controlled device. The return value reflects the energized/deenergized (on/off) state of the relay coil itself. It is the responsibility of the user/programmer to determine which state provides the correct open/closed state of the respective contacts.

Relay Output GET Command Example

Carriage returns and linefeeds are shown for readability, but are not in the actual message.

Get command example for relay output 1:

```
<?xml version="1.0"?>
<!--0000109-->
<GET_S RT="C00" ID="0" L1PW="password">
<EGS O="0x01180001" M="0"/>
</GET_S>
```

Get command response for above message, assuming the relay is ENERGIZED:

```
<?xml version="1.0"?>
<!--0000112-->
<GETACK_S RT="C00" ID="0">
<EGS O="0x01180001" R="0" M="0" UC="1"/>
</GETACK_S>
```

Relay Output SET Command

To set the state of a relay output, the SET command is issued. An example set for a relay output follows. Carriage returns and linefeeds are shown for readability, but are not in the actual message.

```
<?xml version="1.0"?>
<!--000000-->
<SET_S RT="C00" ID="0" L1PW="password">
<ESS O="0x011800cc" M="0" UC="value"/>
</SET_S>
```

There are four portions of the string that must be dynamically created:

- The first (represented by “000000”) is the length of the entire message.
- The second (represented by “*password*”) is the Level 1 (Front Panel) password for the BASIS unit.
- The third (represented by “*cc*”) is the channel number (in hex format) that is being adjusted.
- The fourth (represented by “*value*”) is the value for the new setting. In the case of a relay output, this will be either “1” (relay energized) or “0” (relay deenergized).

The response will have both the response code (as indicated by the “R” value) and an unsigned character return value (limited to 0 for deenergized, or 1 for energized) that indicates the new value of the relay output setting. Only one ESS (Element Set Simple) node is allowed per message.

Managing BASIS via 3rd Party Control Systems

Relay Output SET Command Example

Carriage returns and linefeeds are shown for readability, but are not in the actual message.

Set command example for relay output 2 to force it to the energized state:

```
<?xml version="1.0"?>
<!--0000116-->
<SET_S RT="C00" ID="0" L1PW="password">
<ESS O="0x01180001" M="0" UC="1"/>
</SET_S>
```

Set command response for above message:

```
<?xml version="1.0"?>
<!--0000112-->
<SETACK_S RT="C00" ID="0">
<ESS O="0x01180001" R="0" M="0" UC="1"/>
</SETACK_S>
```

OMNI Input Commands

The OMNI inputs are located on the back of the BASIS. The OMNI inputs can receive (sense) trigger signals from external devices and equipment. The OMNI inputs are driving an analog-to-digital convertor circuit (ADC), one circuit per input. The signals to be sensed can be:

- dry contact closures (open/closed contacts)
- TTL-compatible logic signals (0 vdc “off”, 5 vdc “on”)
- variable resistance (0 to 10 k linear pot)
- variable voltage (0 to 5 vdc range)

The state or value of the OMNI inputs can be read, thus giving a way to transmit local (to the BASIS) state information over the Ethernet to a remote (to the BASIS) control surface.

The OMNI input controls use only the ‘GET’ command structure, since a ‘SET’ can’t be used on an input. The OIDs that are used in the commands to GET the status of the OMNI inputs of a BASIS are shown below. Note the OIDs are zero-base indexed.

For specific details on the operational parameters of the OMNI inputs, including electrical considerations, please contact the product Hardware Manual for the specific BASIS model being used:

Physical Port Number	Port Type	OID
OMNI Input 1	ADC / Logic Input 1	0x01170000
OMNI Input 2	ADC / Logic Input 2	0x01170001
OMNI Input 3	ADC / Logic Input 3	0x01170002
OMNI Input 4	ADC / Logic Input 4	0x01170003
OMNI Input 5	ADC / Logic Input 5	0x01170004
OMNI Input 6	ADC / Logic Input 6	0x01170005

On most BASIS devices there are six OMNI inputs. The BASIS 914lz has two OMNI inputs.

OMNI Input GET Command

To get the value of a particular OMNI input, the GET command is issued. An example get for an OMNI input follows. Carriage returns and linefeeds are shown for readability, but are not in the actual message.

```
<?xml version="1.0"?>
<!--0000000-->
<GET_S RT="C00" ID="0" L1PW="password">
<EGS O="0x011700cc" M="0"/>
```

Managing BASIS via 3rd Party Control Systems

```
</GET_S>
```

There are three portions of the string that must be dynamically created:

- The first (represented by “000000”) is the length of the entire message.
- The second (represented by “password”) is the Level 1 (Front Panel) password for the BASIS unit.
- The third (represented by “cc”) is the number of the OMNI input number that is being requested. The table above lists all possible OMNI input OIDs.

The response will have both the response code (as indicated by the “R” value) and an unsigned character return value (as indicated by the “UC” value of 0 – 255) that indicates the current value of the OMNI input. Only one EGS (Element Get Simple) node is allowed per message.

Keep in mind that the OMNI input can be wired to either active high or active low logic, so the return value does not necessarily reflect the state of the UI on the device sending the logic signal.

OMNI Input GET Command Example

Carriage returns and linefeeds are shown for readability, but are not in the actual message.

Get command example for OMNI input 1 2:

```
<?xml version="1.0"?>
<!--0000109-->
<GET_S RT="C00" ID="0" L1PW="password">
<EGS O="0x01170001" M="0"/>
</GET_S>
```

Get command response for above message, assuming the OMNI is receiving a mid-range resistance (approx. 5k ohm) or voltage (approx. 2.5Vdc):

```
<?xml version="1.0"?>
<!--0000114-->
<GETACK_S RT="C00" ID="0">
<EGS O="0x01170001" R="0" M="0" UC="128"/>
</GETACK_S>
```

Note that a shorted input (or 0 vdc signal) would yield a UC="0" (or very close to zero), while an open input (or 5 vdc) would yield a UC="255". It is up to the user/programmer to determine the proper range(s) of returned value(s).

Managing BASIS via 3rd Party Control Systems

Logic Output Commands

The logic output controls use the ‘GET’ and ‘SET’ command structure. The OIDs that are used in the commands to get and set the output level (HIGH or LOW) of the logic outputs of a BASIS are shown below. Note the OIDs are zero-base indexed.

For specific details on the operational parameters of the logic outputs, including electrical considerations such as TTL/CMOS compatibility and drive specs, please contact the product Hardware Manual for the specific BASIS model being used:

Physical Port Number	Port Type	OID
Logic Output 1	Logic Level Output 1	0x01190000
Logic Output 2	Logic Level Output 2	0x01190001
Logic Output 3	Logic Level Output 3	0x01190002
Logic Output 4	Logic Level Output 4	0x01190003

On most BASIS devices there are four logic outputs. The BASIS 914lz has no logic outputs.

Logic Output GET Command

To get the value of a particular logic output, the GET command is issued. An example get for a logic output follows. Carriage returns and linefeeds are shown for readability, but are not in the actual message.

```
<?xml version="1.0"?>
<!--0000000-->
<GET_S RT="C00" ID="0" L1PW="password">
<EGS O="0x011900cc" M="0"/>
</GET_S>
```

There are three portions of the string that must be dynamically created:

- The first (represented by “0000000”) is the length of the entire message.
- The second (represented by “**password**”) is the Level 1 (Front Panel) password for the BASIS unit.
- The third (represented by “**cc**”) is the number of the logic output that is being adjusted. The table above lists all possible logic output OIDs.

The response will have both the response code (as indicated by the “R” value) and an unsigned character return value (as indicated by the “UC” value, limited to 0 or 1) that indicates the current value of the logic output. Only one EGS (Element Get Simple) node is allowed per message.

Keep in mind that the logic output can be wired to active high or active low logic, so the return value does not necessarily reflect the electrical state of the controlled device.

Managing BASIS via 3rd Party Control Systems

Logic Output GET Command Example

Carriage returns and linefeeds are shown for readability, but are not in the actual message.

Get command example for logic output 2:

```
<?xml version="1.0"?>
<!--0000109-->
<GET_S RT="C00" ID="0" L1PW="password">
<EGS O="0x01190001" M="0"/>
</GET_S>
```

Get command response for above message, assuming logic output is activated:

```
<?xml version="1.0"?>
<!--0000112-->
<GETACK_S RT="C00" ID="0">
<EGS O="0x01190001" R="0" M="0" UC="1"/>
</GETACK_S>
```

Logic Output SET Command

To set the state of a logic output, the SET command is issued. An example set for a logic output follows. Carriage returns and linefeeds are shown for readability, but are not in the actual message.

```
<?xml version="1.0"?>
<!--0000000-->
<SET_S RT="C00" ID="0" L1PW="password">
<ESS O="0x011900cc" M="0" UC="value"/>
</SET_S>
```

There are four portions of the string that must be dynamically created:

- The first (represented by “0000000”) is the length of the entire message.
- The second (represented by “password”) is the Level 1 (Front Panel) password for the BASIS unit.
- The third (represented by “cc”) is the channel number (in hex format) that is being adjusted.
- The fourth (represented by “value”) is the value for the new setting. In the case of a logic output, this will be either “1” (logic high/on) or “0” (logic low/off).

The response will have both the response code (as indicated by the “R” value) and an unsigned character return value (as indicated by the “UC” value, limited to 0 or 1) that indicates the new value of the logic output setting. Only one ESS (Element Set Simple) node is allowed per message.

Managing BASIS via 3rd Party Control Systems

Logic Output SET Command Example

Carriage returns and linefeeds are shown for readability, but are not in the actual message.

Set command example for logic output 4 to a value of 1 (on/high):

```
<?xml version="1.0"?>
<!--0000115-->
<SET_S RT="C00" ID="0" L1PW="password">
<ESS O="0x01190003" M="0" UC="1"/>
</SET_S>
```

Set command response for above message:

```
<?xml version="1.0"?>
<!--0000112-->
<SETACK_S RT="C00" ID="0">
<ESS O="0x01190003" R="0" M="0" UC="1"/>
</SETACK_S>
```

Amplifiers

Almost all QSC amplifier models that come equipped with a Dataport and are connected to the Dataport on a BASIS device through a DB-15 cable can be remotely controlled and queried. The only exception is ISA series amplifiers – while these can be connected to a BASIS via a Dataport, and receive processed audio signals from the BASIS, this series of amps cannot be remotely controlled or queried.

This section outlines some of the remote control functions that are available for all Dataport amps other than the ISA series amps. For purposes of this document, any further reference to Dataport amps does not include the ISA series.

Amplifier Standby Commands

An amplifier can be placed into a standby condition to mute audio and save power.

NOTE: Amplifier Standby is a BASIS Dataport function! It is NOT the actual status of the connected amplifier, if any – for the status of the connected amplifier you will need to query (GET) the "Amplifier Power Status", which is explained following the Amplifier Standby section.

The Amplifier Standby controls use the 'GET' and 'SET' command structure. The OIDs are used in the commands to get and set the standby status of the BASIS Dataports. Amplifiers connected to BASIS Dataports can be controlled through these commands, but an amplifier does NOT need to be connected to the BASIS to respond to a GET or SET command. Note the OIDs are zero-base indexed. Also, using any of the OIDs which ultimately communicate to the same amp, in the case of 4- or 8-channel amps, will yield the same results.

For specific details on the operational parameters of the Dataports, including electrical considerations and number of Dataports, please consult the manual for the specific BASIS model being used:

Physical Port Number	Port Type	OID
Amplifier Standby 1	Dataport A	0x03050000
Amplifier Standby 2	Dataport B	0x03050001
Amplifier Standby 3	Dataport C	0x03050002
Amplifier Standby 4	Dataport D	0x03050003
Amplifier Standby 5	Dataport E	0x03050004
Amplifier Standby 6	Dataport F	0x03050005
Amplifier Standby 7	Dataport G	0x03050006
Amplifier Standby 8	Dataport H	0x03050007

BASIS devices vary on the number of Dataports, typically four or eight Dataports. A command sent to a nonexistent Dataport will be ignored by the BASIS.

Amplifier Standby GET Command

To get the value of a particular amplifier standby control, the GET command is issued. An example GET for an amplifier standby follows. Carriage returns and linefeeds are shown for readability, but are not in the actual message.

```
<?xml version="1.0"?>
<!--0000000-->
<GET_S RT="C00" ID="0" L1PW="password">
<EGS O="0x030500cc" M="0"/>
</GET_S>
```

There are three portions of the string that must be dynamically created:

- The first (represented by “0000000”) is the length of the entire message.
- The second (represented by “password”) is the Level 1 (Front Panel) password for the BASIS unit.
- The third (represented by “cc”) is the number of the amplifier standby (Dataport) that is being queried. The table above lists all possible amplifier standby OIDs.

The response will have both the response code (as indicated by the “R” value) and a short return value (as indicated by the “S” value, limited to 0 for “standby”, or 1 for “on”) that indicates the current value of the amplifier standby. Only one EGS (Element Get Simple) node is allowed per message.

Amplifier Standby GET Command Example

Carriage returns and linefeeds are shown for readability, but are not in the actual message.

Get command example for amplifier standby 1 (Dataport A):

```
<?xml version="1.0"?>
<!--0000109-->
<GET_S RT="C00" ID="0" L1PW="password">
<EGS O="0x03050000" M="0"/>
</GET_S>
```

Get command response for above message, assuming the BASIS Amp Standby is “on”:

```
<?xml version="1.0"?>
<!--0000111-->
<GETACK_S RT="C00" ID="0">
<EGS O="0x03050000" R="0" M="0" S="1"/>
</GETACK_S>
```

Amplifier Standby SET Command

To set the state of an amplifier standby control, the SET command is issued. An example SET for an amplifier standby follows. Carriage returns and linefeeds are shown for readability, but are not in the actual message.

```
<?xml version="1.0"?>
<!--0000000-->
<SET_S RT="C00" ID="0" L1PW="password">
<ESS O="0x030500cc" M="0" S="value"/>
</SET_S>
```

There are four portions of the string that must be dynamically created:

- The first (represented by “0000000”) is the length of the entire message.

Managing BASIS via 3rd Party Control Systems

- The second (represented by “*password*”) is the Level 1 (Front Panel) password for the BASIS unit.
- The third (represented by “*cc*”) is the amplifier number (Dataport number) (in hex format) that is being adjusted.
- The fourth (represented by “*value*”) is the value for the new setting. In the case of an amplifier standby output, this will be either “1” (set the amp to "on" state) or “0” (set the amp to standby), assuming there IS an amp connected. Remember, this is a Dataport command, not really an amp function.

The response will have both the response code (as indicated by the “R” value) and a short return value (as indicated by the “S” value, limited to 0 for "standby", or 1 for "on") that indicates the new value of the amplifier standby setting. Only one ESS (Element Set Simple) node is allowed per message.

Amplifier Standby SET Command Example

Carriage returns and linefeeds are shown for readability, but are not in the actual message.

Set command example for amplifier 4 (Dataport D):

```
<?xml version="1.0"?>
<!--0000115-->
<SET_S RT="C00" ID="0" L1PW="password">
<ESS O="0x03050003" M="0" S="1"/>
</SET_S>
```

Set command response for above message:

```
<?xml version="1.0"?>
<!--0000111-->
<SETACK_S RT="C00" ID="0">
<ESS O="0x03050003" R="0" M="0" S="1"/>
</SETACK_S>
```

Managing BASIS via 3rd Party Control Systems

Amplifier Power Status Commands

An amplifier's power status can be remotely queried through the BASIS.

NOTE: It is this command which returns the true amp status value with regards to On/Standby/Off for a connected amp, including the fact that an amp is indeed connected to the Dataport of the BASIS or not, all based on the returned response code. Merely sending the "Amplifier Standby Set" command as described previously will not guarantee the amp is indeed in the state desired. It is imperative to issue an "Amplifier Power Status" command to ascertain the true status of an amplifier.

The amplifier power status controls use the 'GET' command structure. The OIDs are used in the commands to get the power status of the amplifiers that are connected to the BASIS Dataports. Only amplifiers that are connected to BASIS Dataports can be queried through this command. Note the OIDs are zero-base indexed.

For specific details on the operational parameters of the Dataports, including electrical considerations and number of Dataports, please consult the manual for the specific BASIS model being used:

Physical Port Number	Port Type	OID
Amplifier Power Status 1	Dataport A	0x03060000
Amplifier Power Status 2	Dataport B	0x03060001
Amplifier Power Status 3	Dataport C	0x03060002
Amplifier Power Status 4	Dataport D	0x03060003
Amplifier Power Status 5	Dataport E	0x03060004
Amplifier Power Status 6	Dataport F	0x03060005
Amplifier Power Status 7	Dataport G	0x03060006
Amplifier Power Status 8	Dataport H	0x03060007

BASIS devices vary on the number of Dataports, typically four or eight Dataports. A command sent to a nonexistent Dataport will be ignored by the BASIS.

Amplifier Power Status GET Command

To get the value of a particular amplifier power status control, the GET command is issued. An example GET for an amplifier power status follows. Carriage returns and linefeeds are shown for readability, but are not in the actual message.

```
<?xml version="1.0"?>
<!--0000000-->
<GET_S RT="C00" ID="0" L1PW="password">
<EGS O="0x030600cc" M="0"/>
</GET_S>
```

There are three portions of the string that must be dynamically created:

- The first (represented by “0000000”) is the length of the entire message.
- The second (represented by “password”) is the Level 1 (Front Panel) password for the BASIS unit.
- The third (represented by “cc”) is the number of the amplifier power status (ie: Dataport) that is being queried. The table above lists all possible amplifier power status OIDs.

The response will have both the response code (as indicated by the “R” value) and a short return value (as indicated by the “S” value) that indicates the current value of the amplifier power status. There are four possible values for the short return value:

S="0"	Amp is in standby
S="1"	Amp is on
S="2"	Amp is off, with no AC power (amp switch is physically turned off, or the mains power is removed)
S="255"	Dataport is disconnected, no amp present (also, R="10" instead of the expected R="0")

Only one EGS (Element Get Simple) node is allowed per message.

Amplifier Power Status GET Command Example

Carriage returns and linefeeds are shown for readability, but are not in the actual message.

Get command example for amplifier power status on Dataport 2 (Dataport B):

```
<?xml version="1.0"?>
<!--0000109-->
<GET_S RT="C00" ID="0" L1PW="password">
<EGS O="0x03060001" M="0"/>
</GET_S>
```

GET command response for above message when Amplifier is "on":

```
<?xml version="1.0"?>
<!--0000111-->
<GETACK_S RT="C00" ID="0">
<EGS O="0x03060001" R="0" M="0" S="1"/>
</GETACK_S>
```

GET command response for above message when amplifier is in "Standby":

```
<?xml version="1.0"?>
<!--0000111-->
<GETACK_S RT="C00" ID="0">
<EGS O="0x03060001" R="0" M="0" S="0"/>
```

Managing BASIS via 3rd Party Control Systems

```
</GETACK_S>
```

GET command response for above message when Amplifier is "off" (via front-panel switch or by removing AC mains power):

```
<?xml version="1.0"?>
<!--0000111-->
<GETACK_S RT="C00" ID="0">
<EGS O="0x03060001" R="0" M="0" S="2"/>
</GETACK_S>
```

GET command response for above message when Amplifier Dataport is "disconnected" (no amp detected at BASIS Dataport):

```
<?xml version="1.0"?>
<!--0000114-->
<GETACK_S RT="C00" ID="0">
<EGS O="0x03060001" R="10" M="0" S="255"/>
</GETACK_S>
```

Amplifier Clip Status Commands

An amplifier's clip status can be remotely queried through the BASIS.

The amplifier clip status controls use the 'GET' command structure. These OIDs are used in the commands to get the clip status of the amplifier channels that are connected to the BASIS Dataports. Only Dataport-equipped amplifiers that are connected to BASIS Dataports can be queried through this command. Note the OIDs are zero-base indexed.

For specific details on the operational parameters of the Dataports, including electrical considerations and number of Dataports, please consult the manual for the specific BASIS model being used:

Physical Port Number	Port Type	OID
Amplifier Clip Status 1	Dataport A, Channel 1	0x03090000
Amplifier Clip Status 2	Dataport A, Channel 2	0x03090001
Amplifier Clip Status 3	Dataport B, Channel 1	0x03090002
Amplifier Clip Status 4	Dataport B, Channel 2	0x03090003
Amplifier Clip Status 5	Dataport C, Channel 1	0x03090004
Amplifier Clip Status 6	Dataport C, Channel 2	0x03090005
Amplifier Clip Status 7	Dataport D, Channel 1	0x03090006
Amplifier Clip Status 8	Dataport D, Channel 2	0x03090007
Amplifier Clip Status 9	Dataport E, Channel 1	0x03090008
Amplifier Clip Status 10	Dataport E, Channel 2	0x03090009
Amplifier Clip Status 11	Dataport F, Channel 1	0x0309000A
Amplifier Clip Status 12	Dataport F, Channel 2	0x0309000B
Amplifier Clip Status 13	Dataport G, Channel 1	0x0309000C
Amplifier Clip Status 14	Dataport G, Channel 2	0x0309000D
Amplifier Clip Status 15	Dataport H, Channel 1	0x0309000E
Amplifier Clip Status 16	Dataport H, Channel 2	0x0309000F

BASIS devices vary on the number of Dataports, typically four or eight Dataports. A command sent to a nonexistent Dataport will be ignored by the BASIS.

Amplifier Clip Status GET Command

To get the value of a particular amplifier clip status control, the GET command is issued. An example get for an amplifier clip status follows. Carriage returns and linefeeds are shown for readability, but are not in the actual message.

```
<?xml version="1.0"?>
<!--0000000-->
<GET_S RT="C00" ID="0" L1PW="password">
<EGS O="0x030900cc" M="0"/>
</GET_S>
```

There are three portions of the string that must be dynamically created:

- The first (represented by “0000000”) is the length of the entire message.
- The second (represented by “password”) is the Level 1 (Front Panel) password for the BASIS unit.
- The third (represented by “cc”) is the number of the amplifier clip status (ie: Dataport) that is being queried. The table above lists all possible amplifier clip status OIDs.

The response will have both the response code (as indicated by the “R” value) and a short return value (as indicated by the “S” value, limited to 0 for "not clipping", or 1 for "clipping") that indicates the current value of the amplifier clip status. Only one EGS (Element Get Simple) node is allowed per message.

Amplifier Clip Status GET Command Example

Carriage returns and linefeeds are shown for readability, but are not in the actual message.

Get command example for amplifier clip status on Channel 2 of Dataport 1 (Dataport A):

```
<?xml version="1.0"?>
<!--0000109-->
<GET_S RT="C00" ID="0" L1PW="password">
<EGS O="0x03090001" M="0"/>
</GET_S>
```

Get command response for above message, assuming the channel is clipping:

```
<?xml version="1.0"?>
<!--0000111-->
<GETACK_S RT="C00" ID="0">
<EGS O="0x03090001" R="0" M="0" S="1"/>
</GETACK_S>
```

Amplifier Protect Status Commands

An amplifier's protect status can be remotely queried through the BASIS.

The amplifier protect status controls use the 'GET' command structure. These OIDs are used in the commands to get the protect status of the amplifiers that are connected to the BASIS Dataports. Only dataport-equipped amplifiers that are connected to BASIS Dataports can be queried through this command. Note the OIDs are zero-based indexed.

Most Dataport-based QSC amplifiers will report as being in Protect mode if *either* channel of that amplifier has the "fault condition" which causes the Protect. In other words, querying either Channel 1 or Channel 2 of a Dataport-connected amplifier will yield the same result.

For specific details on the operational parameters of the Dataports, including electrical considerations and number of Dataports, please consult the manual for the specific BASIS model being used:

Physical Port Number	Port Type	OID
Amplifier Protect Status 1	Dataport A, Channel 1	0x030A0000
Amplifier Protect Status 2	Dataport A, Channel 2	0x030A0001
Amplifier Protect Status 3	Dataport B, Channel 1	0x030A0002
Amplifier Protect Status 4	Dataport B, Channel 2	0x030A0003
Amplifier Protect Status 5	Dataport C, Channel 1	0x030A0004
Amplifier Protect Status 6	Dataport C, Channel 2	0x030A0005
Amplifier Protect Status 7	Dataport D, Channel 1	0x030A0006
Amplifier Protect Status 8	Dataport D, Channel 2	0x030A0007
Amplifier Protect Status 9	Dataport E, Channel 1	0x030A0008
Amplifier Protect Status 10	Dataport E, Channel 2	0x030A0009
Amplifier Protect Status 11	Dataport F, Channel 1	0x030A000A
Amplifier Protect Status 12	Dataport F, Channel 2	0x030A000B
Amplifier Protect Status 13	Dataport G, Channel 1	0x030A000C
Amplifier Protect Status 14	Dataport G, Channel 2	0x030A000D
Amplifier Protect Status 15	Dataport H, Channel 1	0x030A000E
Amplifier Protect Status 16	Dataport H, Channel 2	0x030A000F

BASIS devices vary on the number of Dataports, typically four or eight Dataports. A command sent to a nonexistent Dataport will be ignored by the BASIS.

Amplifier Protect Status GET Command

To get the value of a particular amplifier protect status control, the GET command is issued. An example get for an amplifier protect status follows. Carriage returns and linefeeds are shown for readability, but are not in the actual message.

```
<?xml version="1.0"?>
<!--000000-->
<GET_S RT="C00" ID="0" L1PW="password">
<EGS O="0x030A00cc" M="0"/>
</GET_S>
```

There are three portions of the string that must be dynamically created:

- The first (represented by "000000") is the length of the entire message.
- The second (represented by "password") is the Level 1 (Front Panel) password for the BASIS unit.

Managing BASIS via 3rd Party Control Systems

- The third (represented by “*cc*”) is the number of the amplifier protect status (ie: Dataport) that is being queried. The table above lists all possible amplifier protect status OIDs.

The response will have both the response code (as indicated by the “R” value) and a short return value (as indicated by the “S” value, limited to 0 for “not in protect mode”, or 1 for “in protect mode”) that indicates the current value of the amplifier protect status. Only one EGS (Element Get Simple) node is allowed per message.

Amplifier Protect Status GET Command Example

Carriage returns and linefeeds are shown for readability, but are not in the actual message.

Get command example for amplifier protect status on Dataport 1(A), Channel 2:

```
<?xml version="1.0"?>
<!--0000109-->
<GET_S RT="C00" ID="0" L1PW="password">
<EGS O="0x030A0001" M="0"/>
</GET_S>
```

Get command response for above message, assuming the amp is in protect mode:

```
<?xml version="1.0"?>
<!--0000111-->
<GETACK_S RT="C00" ID="0">
<EGS O="0x030A0001" R="0" M="0" S="1"/>
</GETACK_S>
```

Amplifier Temperature Commands

An amplifier's temperature can be remotely queried through the Basis. The temperature controls use the 'get' command structure. These OIDs are used in the commands to get the temperature of the amplifiers that are connected to the Basis Dataports. Only dataport-equipped amplifiers that are connected to Basis Dataports can be queried through this command.

For specific details on the operational parameters of the Dataports, including electrical considerations and number of Dataports, please consult the manual for the specific Basis model being used:

Physical Port Number	Port Type	OID
Amplifier Temperature 1	Dataport A, Ch1	0x030C0000
Amplifier Temperature 2	Dataport A, Ch2	0x030C0001
Amplifier Temperature 3	Dataport B, Ch1	0x030C0002
Amplifier Temperature 4	Dataport B, Ch2	0x030C0003
Amplifier Temperature 5	Dataport C, Ch1	0x030C0004
Amplifier Temperature 6	Dataport C, Ch2	0x030C0005
Amplifier Temperature 7	Dataport D, Ch1	0x030C0006
Amplifier Temperature 8	Dataport D, Ch2	0x030C0007
Amplifier Temperature 9	Dataport E, Ch1	0x030C0008
Amplifier Temperature 10	Dataport E, Ch2	0x030C0009
Amplifier Temperature 11	Dataport F, Ch1	0x030C000A
Amplifier Temperature 12	Dataport F, Ch2	0x030C000B
Amplifier Temperature 13	Dataport G, Ch1	0x030C000C
Amplifier Temperature 14	Dataport G, Ch2	0x030C000D
Amplifier Temperature 15	Dataport H, Ch1	0x030C000E
Amplifier Temperature 16	Dataport H, Ch2	0x030C000F

Basis devices vary on the number of Dataports. A command sent to a nonexistent Dataport will be ignored by the Basis.

Amplifier Temperature GET Command

To get the value of a particular amplifier's temperature control, the get command is issued. An example get follows. Carriage returns and linefeeds are shown for readability, but are not in the actual message.

```
<?xml version="1.0"?>
<!--0000000-->
<GET_S RT="C00" ID="0" L1PW="password">
<EGS O="0x030C00cc" M="0"/>
</GET_S>
```

There are three portions of the string that must be dynamically created:

- The first (represented by "0000000") is the length of the entire message.
- The second (represented by "password") is the Level 1 (Front Panel) password for the BASIS unit.
- The third (represented by "cc") is the number of the amplifier temperature status (ie: Dataport Output Channel) that is being queried. The table above lists all possible amplifier temperature status OIDs.

The response will have both the response code (as indicated by the "R" value) and a float return value (as indicated by the "F" value) that indicates the current value of the amplifier temperature control, in degrees Celsius. Only one EGS (Element Get Simple) node is allowed per message.

Amplifier Temperature GET Command Example

Carriage returns and linefeeds are shown for readability, but are not in the actual message.

Get command example for amplifier temperature on Dataport A, Channel 2:

```
<?xml version="1.0"?>
<!--0000109-->
<GET_S RT="C00" ID="0" L1PW="password">
<EGS O="0x030C0001" M="0"/>
</GET_S>
```

Get command response for above message:

```
<?xml version="1.0"?>
<!--0000114-->
<GETACK_S RT="C00" ID="0">
<EGS O="0x030C0001" R="0" M="0" F="23.3"/>
</GETACK_S>
```

Monitor Chain Commands

The monitor chain in a BASIS can be controlled remotely. To set up the monitor chain, a few things need to be done to assure that the results are as expected.

First, remember that there are only four monitor tap points that are available at any one time from any one BASIS device. Selecting a fifth monitor tap will automatically disable the first one that was selected, maintaining the limit of four active taps at any one time.

Also note that when a tap selection has been made, there are individual gain controls for each tap selection type. The gain for the selected monitor tap needs to be set to provide adequate signal from the monitor output. The values for the gain of disabled taps are retained for later recall/tap selection.

Monitor Tap Selection

The monitor tap select control uses the 'SET' and 'GET' command structures. These OIDs are used in the commands to get and set the monitor tap selection for the channels that connect to the outputs. The monitor tap select controls are the same controls that reside in the "Monitor" dialog of Venue Manager.

For specific details on the operational parameters of the Monitor Chain, including meaning of settings, please consult the manual for the specific BASIS model being used:

Physical Port Number	Port Type	OID
Monitor Tap Select 1	Dataport A, Channel 1	0x01130000
Monitor Tap Select 2	Dataport A, Channel 2	0x01130001
Monitor Tap Select 3	Dataport B, Channel 1	0x01130002
Monitor Tap Select 4	Dataport B, Channel 2	0x01130003
Monitor Tap Select 5	Dataport C, Channel 1	0x01130004
Monitor Tap Select 6	Dataport C, Channel 2	0x01130005
Monitor Tap Select 7	Dataport D, Channel 1	0x01130006
Monitor Tap Select 8	Dataport D, Channel 2	0x01130007
Monitor Tap Select 9	Dataport E, Channel 1	0x01130008
Monitor Tap Select 10	Dataport E, Channel 2	0x01130009
Monitor Tap Select 11	Dataport F, Channel 1	0x0113000A
Monitor Tap Select 12	Dataport F, Channel 2	0x0113000B
Monitor Tap Select 13	Dataport G, Channel 1	0x0113000C
Monitor Tap Select 14	Dataport G, Channel 2	0x0113000D
Monitor Tap Select 15	Dataport H, Channel 1	0x0113000E
Monitor Tap Select 16	Dataport H, Channel 2	0x0113000F

BASIS devices vary on the number of Dataports. A command sent to a nonexistent Dataport will be ignored by the BASIS.

Monitor Tap Select GET Command

To get the value of a particular monitor tap select control, the GET command is issued. An example get for a monitor tap select follows. Carriage returns and linefeeds are shown for readability, but are not in the actual message.

```
<?xml version="1.0"?>
<!--000000-->
<GET_S RT="C00" ID="0" L1PW="password">
<EGS O="0x011300cc" M="0"/>
</GET_S>
```

Managing BASIS via 3rd Party Control Systems

There are three portions of the string that must be dynamically created:

- The first (represented by “000000”) is the length of the entire message.
- The second (represented by “password”) is the Level 1 (Front Panel) password for the BASIS unit.
- The third (represented by “cc”) is the monitor tap (ie: Dataport channel) that is being queried. The table above lists all possible monitor tap selection OIDs.

The response will have both the response code (as indicated by the “R” value) and a short return value (as indicated by the “S” value, limited to 0 to 3) that indicates the current monitor tap point selection:

0 = Off

1 = Pre-Basis tap

2 = Post-Basis / DataPort output tap

3 = Post amplifier output tap

Only one EGS (Element Get Simple) node is allowed per message.

Monitor Tap Select GET Command Example

Carriage returns and linefeeds are shown for readability, but are not in the actual message.

Get command example for monitor tap select on Dataport A, channel 2:

```
<?xml version="1.0"?>
<!--0000109-->
<GET_S RT="C00" ID="0" L1PW="password">
<EGS O="0x01130001" M="0"/>
</GET_S>
```

Get command response for above message:

```
<?xml version="1.0"?>
<!--0000111-->
<GETACK_S RT="C00" ID="0">
<EGS O="0x01130001" R="0" M="0" S="0"/>
</GETACK_S>
```

Monitor Tap Select SET Command Example

Carriage returns and linefeeds are shown for readability, but are not in the actual message.

Set command example for tap select of ‘post’ on channel 4 (Dataport B, channel 2):

```
<?xml version="1.0"?>
<!--0000115-->
<SET_S RT="C00" ID="0" L1PW="password">
<ESS O="0x01130003" M="0" S="2"/>
</SET_S>
```

Set command response for above message:

```
<?xml version="1.0"?>
<!--0000113-->
<SETACK_S RT="C00" ID="0">
<ESS O="0x01130003" R="0" M="0" S="2"/>
</SETACK_S>
```

Managing BASIS via 3rd Party Control Systems

Monitor Gain Commands

The gain level of the various taps for the monitor chain can be changed remotely. The gain controls are the same controls that reside in the “Monitor” dialog of Venue Manager. There is a gain control for each of the tap points – pre, post, and amplifier output.

All of the monitor tap gain controls use the ‘GET’ and ‘SET’ command structure. The OIDs that are used in the commands to get and set the gain of the monitor outputs of a BASIS are shown below:

Monitor Pre Gain Setting	Port Type	OID
Monitor Pre Gain 1	Dataport A, Channel 1	0x01140000
Monitor Pre Gain 2	Dataport A, Channel 2	0x01140001
Monitor Pre Gain 3	Dataport B, Channel 1	0x01140002
Monitor Pre Gain 4	Dataport B, Channel 2	0x01140003
Monitor Pre Gain 5	Dataport C, Channel 1	0x01140004
Monitor Pre Gain 6	Dataport C, Channel 2	0x01140005
Monitor Pre Gain 7	Dataport D, Channel 1	0x01140006
Monitor Pre Gain 8	Dataport D, Channel 2	0x01140007
Monitor Pre Gain 9	Dataport E, Channel 1	0x01140008
Monitor Pre Gain 10	Dataport E, Channel 2	0x01140009
Monitor Pre Gain 11	Dataport F, Channel 1	0x0114000A
Monitor Pre Gain 12	Dataport F, Channel 2	0x0114000B
Monitor Pre Gain 13	Dataport G, Channel 1	0x0114000C
Monitor Pre Gain 14	Dataport G, Channel 2	0x0114000D
Monitor Pre Gain 15	Dataport H, Channel 1	0x0114000E
Monitor Pre Gain 16	Dataport H, Channel 2	0x0114000F
Monitor Post Gain Setting	Port Type	OID
Monitor Post Gain 1	Dataport A, Channel 1	0x01150000
Monitor Post Gain 2	Dataport A, Channel 2	0x01150001
Monitor Post Gain 3	Dataport B, Channel 1	0x01150002
Monitor Post Gain 4	Dataport B, Channel 2	0x01150003
Monitor Post Gain 5	Dataport C, Channel 1	0x01150004
Monitor Post Gain 6	Dataport C, Channel 2	0x01150005
Monitor Post Gain 7	Dataport D, Channel 1	0x01150006
Monitor Post Gain 8	Dataport D, Channel 2	0x01150007
Monitor Post Gain 9	Dataport E, Channel 1	0x01150008
Monitor Post Gain 10	Dataport E, Channel 2	0x01150009
Monitor Post Gain 11	Dataport F, Channel 1	0x0115000A
Monitor Post Gain 12	Dataport F, Channel 2	0x0115000B
Monitor Post Gain 13	Dataport G, Channel 1	0x0115000C
Monitor Post Gain 14	Dataport G, Channel 2	0x0115000D
Monitor Post Gain 15	Dataport H, Channel 1	0x0115000E
Monitor Post Gain 16	Dataport H, Channel 2	0x0115000F
Monitor Speaker Gain Setting	Port Type	OID
Monitor Speaker Gain 1	Dataport A, Channel 1	0x01160000
Monitor Speaker Gain 2	Dataport A, Channel 2	0x01160001
Monitor Speaker Gain 3	Dataport B, Channel 1	0x01160002
Monitor Speaker Gain 4	Dataport B, Channel 2	0x01160003
Monitor Speaker Gain 5	Dataport C, Channel 1	0x01160004
Monitor Speaker Gain 6	Dataport C, Channel 2	0x01160005
Monitor Speaker Gain 7	Dataport D, Channel 1	0x01160006
Monitor Speaker Gain 8	Dataport D, Channel 2	0x01160007
Monitor Speaker Gain 9	Dataport E, Channel 1	0x01160008
Monitor Speaker Gain 10	Dataport E, Channel 2	0x01160009
Monitor Speaker Gain 11	Dataport F, Channel 1	0x0116000A
Monitor Speaker Gain 12	Dataport F, Channel 2	0x0116000B

Managing BASIS via 3rd Party Control Systems

Monitor Speaker Gain 13	Dataport G, Channel 1	0x0116000C
Monitor Speaker Gain 14	Dataport G, Channel 2	0x0116000D
Monitor Speaker Gain 15	Dataport H, Channel 1	0x0116000E
Monitor Speaker Gain 16	Dataport H, Channel 2	0x0116000F

On BASIS devices, the first eight output ports in the DSP chain are designed to correspond to the Dataport or euro (analog) outputs on the back of the device, which can also be routed to CobraNet. The remaining DSP outputs are designed to correspond to the outputs that can route to CobraNet.

The number of Dataport and CobraNet channels supported by a particular BASIS device will vary by model. *For specific details on the number of Dataport and CobraNet outputs, please consult the product Hardware Manual for the specific BASIS model being used.*

Monitor Gain GET Command

To get the value of a gain setting for a particular monitor tap, the GET command is issued. An example get command for a gain setting follows. Carriage returns and linefeeds are shown for readability, but are not in the actual message.

Pre gain:

```
<?xml version="1.0"?>
<!--0000000-->
<GET_S RT="C00" ID="0" L1PW="password">
<EGS O="0x011400cc" M="0"/>
</GET_S>
```

Post gain:

```
<?xml version="1.0"?>
<!--0000000-->
<GET_S RT="C00" ID="0" L1PW="password">
<EGS O="0x011500cc" M="0"/>
</GET_S>
```

Speaker gain:

```
<?xml version="1.0"?>
<!--0000000-->
<GET_S RT="C00" ID="0" L1PW="password">
<EGS O="0x011600cc" M="0"/>
</GET_S>
```

There are three portions of the string that must be dynamically created:

- The first (represented by “0000000”) is the length of the entire message.
- The second (represented by “password”) is the Level 1 (Front Panel) password for the BASIS unit.
- The third (represented by “cc”) is the channel number (in hex format) that is being adjusted. The table above lists all possible monitor gain OIDs.

The response will have both the response code (as indicated by the “R” value) and a floating point return that indicates the current value of the Gain setting. Only one EGS (Element Get Simple) node is allowed per message.

Monitor Gain GET Command Example

Carriage returns and linefeeds are shown for readability, but are not in the actual message.

Get command example for channel 3, monitor pre gain:

```
<?xml version="1.0"?>
```

Managing BASIS via 3rd Party Control Systems

```
<!--0000109-->
<GET_S RT="C00" ID="0" L1PW="password">
<EGS O="0x01140002" M="0"/>
</GET_S>
```

Get command response for above message, assuming Ch 3, Monitor Pre Gain is set for -12.0:

```
<?xml version="1.0"?>
<!--0000115-->
<GETACK_S RT="C00" ID="0" L1PW="password">
<EGS O="0x01140002" R="0" M="0" F="-12.0"/>
</GETACK_S>
```

Monitor Gain SET Command

To set the value of a gain setting for a particular monitor tap, the SET command is issued. An example set for a monitor gain setting follows. Carriage returns and linefeeds are shown for readability, but are not in the actual message.

Pre gain:

```
<?xml version="1.0"?>
<!--0000000-->
<SET_S RT="C00" ID="0" L1PW="password">
<ESS O="0x011400cc" M="0" F="value"/>
</SET_S>
```

Post gain:

```
<?xml version="1.0"?>
<!--0000000-->
<SET_S RT="C00" ID="0" L1PW="password">
<ESS O="0x011500cc" M="0" F="value"/>
</SET_S>
```

Speaker gain:

```
<?xml version="1.0"?>
<!--0000000-->
<SET_S RT="C00" ID="0" L1PW="password">
<ESS O="0x011600cc" M="0" F="value"/>
</SET_S>
```

There are four portions of the string that must be dynamically created:

- The first (represented by "0000000") is the length of the entire message.
- The second (represented by "password") is the Level I (Front Panel) password for the BASIS unit.
- The third (represented by "cc") is the channel number (in hex format) that is being adjusted.
- The fourth (represented by "value") is the value for the new setting. In the case of a Monitor Gain, this will be in the range of "0.0" and "-95.5" in .5db increments.

The response will have both the response code (as indicated by the "R" value) and a float return (as indicated by the "F" value) that indicates the new value of the Gain setting. Only one ESS (Element Set Simple) node is allowed per message.

Monitor Gain SET Command Example

Carriage returns and linefeeds are shown for readability, but are not in the actual message.

Set command example for channel 10, monitor post gain:

```
<?xml version="1.0"?>
<!--0000119-->
<SET_S RT="C00" ID="0" L1PW="password">
<ESS O="0x0115000a" M="0" F="-24.0"/>
</SET_S>
```

Set command response for above message:

```
<?xml version="1.0"?>
<!--0000114-->
<SETACK_S RT="C00" ID="0">
<ESS O="0x0115000a" R="0" M="0" F="-24.0"/>
</SETACK_S>
```

DSP Chain Commands

A control device can send commands to all of the Snapshot-capable controls of all of the processors in the DSP chain of the active Configuration.

This section will tell you how to connect to any of the Snapshot-capable controls in the active DSP chain, regardless of how the DSP chain is configured. It will also inform you of a number of potential problems that can arise if the control architecture is not thoroughly thought-out, and how to avoid that.

Dynamic DSP Overview

All of the other sections of this document have been geared towards controls that are always present in a BASIS device, no matter how it is programmed. A BASIS922az, for instance, always has 8 analog input channels, and 8 analog output channels. The controls that are always present in a BASIS are referred to as static controls, or static object IDs (OID). They are called static because they are always there for a particular BASIS model.

Controls in the DSP chain are, by virtue of the user-defined nature of the DSP chain, not always present in a BASIS. In fact, since every one of the eight Configurations in a BASIS can have a different DSP chain, a control in the DSP chain may or may not be accessible simply given which Configuration is presently active. The controls in the DSP chain are called dynamic controls, or dynamic object IDs (OID). They are called dynamic because they are dynamically allocated in response to user programming.

A dynamic control / OID is no different to the BASIS than a static control / OID, with regard to control commands. A dynamic OID responds to the same 'set simple' (SET_S) / 'get simple' (GET_S) syntax as the static controls do. It is simply a matter of determining the proper OID value for the message, and getting the data type from the processor parameter table. From that, a message can be built to set/get the dynamic control value.

The following sections will tell you everything you need to know to be able to formulate the control messages for all of the dynamic OIDs that are available.

Dynamic DSP Cautions

The content of the DSP chain portion of every flashed Configuration that is stored in a BASIS is completely under the control of the end user. This includes signal path, processing, and the values of the processor parameters. As such, it is not possible for the BASIS firmware to assume very much about the content of any of the flashed Configurations. It is possible that each flashed Configuration in a BASIS is dramatically different – in both structure and processing.

This flexibility, that of the BASIS to have many different (potentially incompatible) flashed Configurations ready for recall, is a concern when third party control is being used. The reason for concern is that the object IDs (OID) in one Configuration may overlap the OIDs of another, incompatible Configuration. This is a built-in limitation of the object-addressing scheme.

Briefly speaking, the OID contains: a) the processor ID, b) the processor parameter, and c) the channel index. Among these three pieces of data, it is not possible for a command destined for the first channel gain setting of a gain block called 'Block 1' in one Configuration to be differentiated from the delay time of a delay block called 'Block 1' in another Configuration. Both object IDs are exactly the same (0x40010400), even though their purposes in the two signal chains are obviously not the same. If a third party control device were sending commands to OID 0x40010400 and the active Configuration changed from the first Configuration to the second, the resulting (likely unintentional) changes made to the delay time would likely not be the desired result.

Since the flashed Configurations can be so different, and since the OIDs in one Configuration can overlap OIDs in another Configuration, it is recommended that when using a third party control device that you set up all of the flashed Configurations in the BASIS to be the same (identical) Configuration. This way, when the active Configuration is changed (intentionally or not), the OIDs all remain the same and retain their same meaning between Configuration changes.

Managing BASIS via 3rd Party Control Systems

If it is not possible for the same Configuration to be put into all of the flashed Configurations, it is important that the control device reads the Configuration and Snapshot values from the BASIS periodically so that the control device can make sure that the OIDs that are being controlled are, in fact, the correct ones. In essence, this check makes the active Configuration number a qualifier to the OID, assuring a unique address for a control within all flashed Configurations for the device.

NOTE: While one could certainly calculate all the various OIDs for the signal flow of a particular Config in a Basis by reading the remainder of this section, it is HIGHLY recommended you use the special application “BasisTerm.exe” to generate your desired OIDs and parameters. BasisTerm will save you literally HOURS of calculations. In addition, if you are connected to a live QSControl network, you can use BasisTerm to test and verify your desired XML strings. BasisTerm is included, along with this document, in the default installation directory for QSControl.Net software. Usually, you will find this directory at:

C:\Program Files\QSC Audio\QSControl.net\Third Party Control

Dynamic OID Calculation

To send commands to a control in the DSP chain, it is necessary to calculate the value of the particular OID to be controlled. This is easily done, as long as the following is known:

- The type of DSP object
- The processor ID number.
- The parameter code.
- The parameter index (indices).

The type of processor should be self-evident in the DSP chain; it is simply the type of processor as shown in the Venue Manager DSP programming user interface. The type of processor will be used to look into this document to a) locate the proper parameter code in the following table, and b) locate the appropriate calculation to use for the OID.

The processor ID can be obtained in two ways. If the processor has not been renamed (the name of the processor is still of the format ‘Block n’), then the processor number is the number shown in the name. If the processor has been renamed (the processor name is not of the format ‘Block n’), then the existing name can be temporarily erased in the DSP editor and the default name will be automatically shown.

The parameter code can be obtained from the table in the next section. Note that only Snapshot-capable controls are in the table, so if a control is missing it is probably not Snapshot-capable.

The parameter index is the zero-based index of the control within the processor. For example, if the first gain level in a multi-channel gain block is to be controlled, then the index would be 0. If the second gain level is to be controlled, then the index would be 1, and so on. Controls that are visually grouped together are usually of the same index value. So, for instance, the associated mute controls for the preceding gain block example would carry the index values of 0 and 1, respectively, for the channel 1 and 2 mutes.

In the case of crossovers and mixers, the calculation is a little different, and requires a little more data. In the crossover, there are two indices. One is for the band, and the second is for the filter number within the band. In the mixer, there are also two indices. One index for the input channel, one index for the output channel, and the two are used together for accessing the matrix.

Parameter Codes by Processor

The following table lists the parameter codes and ranges for each of the processors that are capable of being controlled via external controllers. Any special considerations are also noted for each processor.

The table contains the following:

- Processor Name
This is the name of the processor when viewed in Venue Manager.
- Parameter Name
This is the name of the parameter within the processor.
- Parameter Range
This is the usable range and expected units of the parameter.
- Data Type
This is the data type code that the BASIS expects for the parameter.
- Parameter Code
This is the decimal representation of the parameter code.

The BASIS requires that a control value that is supplied to the device is in the proper format for the type code that is specified. The data types and their associated formats are:

- **F** Data Type
Float data type. A float must include at least one digit to the left of the decimal point. For values less than 1.0, this will require a leading zero.
- **S** Data Type
Short data type. A short must not include any characters other than zero through nine. A short should not include a leading zero.
- **ST** Data Type
String data type. A string must not include any characters other than 0-9 and a-z (upper or lower case).

Managing BASIS via 3rd Party Control Systems

Processor	Parameter	Range	Type	Code
AGC	Processor Name	16 characters max.	ST	0
	Target Level	-80dB to 0dB	F	5
	Threshold	-80dB to Target Level	F	7
	Ratio	1:1 to 5:1	F	8
	Release Time	50ms to 60s	F	9
	Side Chain HPF	1.0Hz to 1kHz	F	10
	Bypass	0=normal, 1=bypass	S	11

Processor	Parameter	Range	Type	Code
Automixer	Processor Name	16 characters max.	ST	0
	Input Gain	-100dB to +12dB	F	4
	Input Mute	0=normal, 1=mute	S	5
	Input Priority	0=normal, 1=priority	S	6
	Program Gain	-100dB to +12dB	F	7
	Program Off	0=on, 1=off	S	8
	Mix Gain	-100dB to +12dB	F	9
	Mix Mute	0=normal, 1=mute	S	10
	Hold Time	0s to 60s	F	11
	Response Time	100ms to 60s	F	12
	Upper Threshold	-80dB to 0dB	F	13
	Lower Threshold	-80dB to 0dB	F	14

Processor	Parameter	Range	Type	Code
Compressor	Processor Name	16 characters max.	ST	0
	Threshold	-150dB to bottom of soft knee	F	5
	Ratio	1:1 to over 100000:1	F	6
	Soft Knee Width	0.8dB to 20dB	F	7
	Attack Time	100us to 60s (must be < Release Time)	F	9
	Release Time	1ms to 60s (must be > Attack Time)	F	10
	Delay Time	0ms to 50ms	F	11
	Delay Mode	0=no delay, 1=use Delay Time, 2=optimum, 3=minimum overshoot	S	12
	Output Gain	-88dB to +12dB	F	13
	Polarity	0=normal, 1=invert	S	14
	Mute	0=normal, 1=mute	S	15
Bypass	0=normal, 1=bypass	S	16	

Processor	Parameter	Range	Type	Code
Crossover	Processor Name	16 characters max.	ST	0
	Filter Frequency	20Hz to 20kHz	F	6

The crossover uses a different OID calculation than most processors. Refer to the crossover OID calculation section.

Frequency synchronization between adjacent sections of the crossover will be maintained by the BASIS.

Managing BASIS via 3rd Party Control Systems

Processor	Parameter	Range	Type	Code
Delay	Processor Name	16 characters max.	ST	0
	Delay Time	0s to 0.910s (must be < Delay Max)	F	4
	Bypass	0=normal, 1=bypass	S	5
	Peak Meter	0 to -127 dB	F	6

The delay time must be less than the maximum delay time of the processor, as set up in Venue Manager. The maximum delay time can not be changed during runtime.

Processor	Parameter	Range	Type	Code
Ducker	Processor Name	16 characters max.	ST	0
	Threshold	-100dB to 0dB	F	4
	Attenuation	-100dB to 0dB	F	5
	Fade Out Time	100us to 60s	F	6
	Hold Time	0s to 60s	F	7
	Fade In Time	100us to 60s	F	8
	Priority Gain	-100dB to +12dB	F	9
	Priority Mute	0=normal, 1=mute	S	10
	Bypass	0=normal, 1=mute	S	11
	Peak Meter	0 to -127dB	F	12

Processor	Parameter	Range	Type	Code
Dynamics	Processor Name	16 characters max.	ST	0
	Compressor Threshold	-100dB to 0dB	F	5
	Compressor Ratio	1:1 to over 100,000:1	F	6
	Expander Threshold	-150dB to 0dB	F	7
	Expander Ratio	0.0588:1 to 1:1	F	8
	Noise Gate Threshold	-150dB to 0dB	F	9
	Soft Knee Width	0.753dB to 20dB	F	10
	RMS Avg. Time	0s to 60s	F	11
	Attack Time	20us to 60s (must be < Release Time)	F	12
	Release Time	100us to 60s (must be > Attack Time)	F	13
	Delay Time	0ms to 200ms	F	14
	Delay Mode	0=none, 1=use Delay Time, 2=optimum, 3=minimum overshoot	S	15
	Output Gain	-88dB to +12dB	F	16
	Polarity	0=normal, 1=invert	S	17
	Mute	0=normal, 1=mute	S	18
Bypass	0=normal, 1=bypass	S	19	

Processor	Parameter	Range	Type	Code
Equalizer	Processor Name	16 characters max.	ST	0
	Frequency	20Hz to 20kHz	F	6
	Q	0.2 to 70.0	F	7
	Low Gain	-24dB to +12dB	F	8
	Mid Gain	-24dB to +12dB	F	9
	High Gain	-24dB to +12dB	F	10
	Bypass	0=normal, 1=bypass	S	11

Processor	Parameter	Range	Type	Code
Gain	Processor Name	16 characters max.	ST	0

Managing BASIS via 3rd Party Control Systems

	Gain Value	-88dB to +12dB	F	4
	Mute	0=normal, 1=mute	S	5
	Polarity	0=normal, 1=invert	S	6
	Meter		F	7

Processor	Parameter	Range	Type	Code
Graphic EQ	Processor Name	16 characters max.	ST	0
	Bypass	0=normal, 1=bypass	S	10
	User Gain	-12dB to +12dB	F	11

Processor	Parameter	Range	Type	Code
Limiter	Processor Name	16 characters max.	ST	0
	Threshold	-150dB to 0dB	F	4
	Attack Time	20us to 60s (must be < Release Time)	F	5
	Release Time	100us to 60s (must be > Attack Time)	F	6
	Delay Time	0ms to 50ms	F	7
	Delay Mode	0=none, 1=use Delay Time, 2=optimum, 3=minimum overshoot	S	8
	Output Gain	-88dB to +12dB	F	9
	Polarity	0=normal, 1=invert	S	10
	Mute	0=normal, 1=mute	S	11
	Bypass	0=normal, 1=bypass	S	12

Processor	Parameter	Range	Type	Code
Mixer	Processor Name	16 characters max.	ST	0
	Crosspoint Level	-100dB to 0dB	F	4
	Crosspoint Mute	0=normal, 1=mute	S	5
	Crosspoint Polarity	0=normal, 1=invert	S	6
	Input Level	-100dB to 0dB	F	7
	Input Mute	0=normal, 1=mute	S	8
	Input Polarity	0=normal, 1=invert	S	9
	Output Gain	-100dB to +12dB	F	10
	Output Mute	0=normal, 1=mute	S	11
	Output Polarity	0=normal, 1=invert	S	12
	Output Peak Meter	0 to -127dB	F	13

The mixer uses a different OID calculation than most of the other processors. Please refer to the OID calculation for the mixer.

Mixer crosspoints must be activated in Venue Manager. Mixer crosspoints cannot be activated during runtime.

Managing BASIS via 3rd Party Control Systems

Processor	Parameter	Range	Type	Code
Noise Gate	Processor Name	16 characters max.	ST	0
	Threshold	-150dB to 0dB	F	5
	Attenuation	-100dB to 0dB	F	6
	Attack Time	100us to 60s (must be < Release Time)	F	7
	Hold Time	0s to 60s	F	8
	Release Time	100us to 60s (must be > Attack Time)	F	9
	Delay Time	0ms to 50ms	F	10
	Delay Mode	0=none, 1=use Delay Time, 2=optimum, 3=minimum overshoot	S	11
	Bypass	0=normal, 1=bypass	S	12

Processor	Parameter	Range	Type	Code
Noise Generator	Processor Name	16 characters max.	ST	0
	Type	0=white, 1=pink	S	3
	Level	-88dB to 0dB	F	4
	Mute	0=normal, 1=mute	S	5

Processor	Parameter	Range	Type	Code
RMS Meter	Processor Name	16 characters max.	ST	0
	Averaging Time	0.1s to 60s	F	3
	Meter Value	0 to -127 dB	F	4

Processor	Parameter	Range	Type	Code
Router	Processor Name	16 characters max.	ST	0
	Input Select	-1=mute, any other number is input channel select. (must be <= number of channels)	S	3
	Peak Meter	0 to -127dB	F	7

Processor	Parameter	Range	Type	Code
Tone Generator	Processor Name	16 characters max.	ST	0
	Frequency	20hZ to 20kHz	F	3
	Level	-88dB to 0dB	F	4
	Polarity	0=normal, 1=invert	S	5
	Mute	0=normal, 1=mute	S	6

OID Calculation for DSP Objects except Crossover and Mixer

The calculation for a control OID that is not in a crossover or a mixer is as follows. Please note that the following is shown in hexadecimal fashion, as the OID is a bit mask comprised of several pieces of data. A decimal representation of the same calculation will also be shown for reference.

OID Definition							
Byte 1		Byte 2		Byte 3		Byte 4	
2 bits	14 bits			8 bits		8 bits	
01	IIIIII	IIII	IIII	PPPP	PPPP	XXXX	XXXX

where:

- 01 = the bit mask to mark a dynamic OID (high order 2 bits)
- I = the processor ID number from Venue Manager (14 bits)
- P = the parameter code from the parameter code table (8 bits)
- X = the zero-relative index of the control in a multi-channel processor (8 bits)

As an example, to calculate the OID for a mute of a single channel gain block that is called 'Block 7', simply perform the calculation as follows:

```

Starting bit mask:          0x40000000
Processor ID (in hex):     0x40070000    ← This is 'I' above.
Parameter Code (in hex):  0x40070500    ← This is 'P' above.
Channel Index (in hex):    0x40070500    ← This is 'X' above.
Final OID for Mute Control: 0x40070500
    
```

As another example, to calculate the OID for a bypass of a peak limiter block that is called 'Block 67', simply perform the calculation as follows:

```

Starting bit mask:          0x40000000
Processor ID (in hex):     0x40430000
Parameter Code (in hex):  0x40430C00
Channel Index (in hex):    0x40430C00
Final OID for Bypass Control: 0x40430C00
    
```

As a final example, to calculate the OID for the level of channel 4 of an 8-channel gain block that is called 'Block 234', simply perform the calculation as follows:

```

Starting bit mask:          0x40000000
Processor ID (in hex):     0x40EA0000
Parameter Code (in hex):  0x40EA0400
Channel Index (in hex):    0x40EA0503    ← X is zero-relative
Final OID for Level 3 Control: 0x40EA0403
    
```

OID values can be calculated with decimal arithmetic. Simply perform the following calculation with the same processor ID, parameter code, and channel index values you obtained for the above:

$$0x40000000 + (\text{Processor ID} * 65536) + (\text{Parameter Code} * 256) + \text{Index}$$

The above-described calculations can be used for all processors with the exception of the crossover and the mixer.

OID Calculation for Crossover

The calculation for a control OID that is in a crossover is as follows. Please note that the following is shown in hexadecimal fashion, as the OID is a bit mask comprised of several pieces of data. A decimal representation of the same calculation will also be shown for reference.

OID Definition						
Byte 1		Byte 2		Byte 3		Byte 4
2 bits	14 bits			4 bits	6 bits	6 bits
01	IIIIII	IIII	IIII	PPPP	XXXXXX	YYYYYY

where:

- 01 = the bit mask to mark a dynamic OID (high-order 2 bits)
- I = the processor ID number from Venue Manager (14 bits)
- P = the parameter code from the parameter code table (4 bits)
- X = the crossover band zero-relative index (6 bits)
- Y = the crossover filter zero-relative index (6 bits)

The crossover band and filter indices require clarification. Each crossover has groups of 1 or 2 filters that are set up to act as a unit to pass audio for a particular part of the audio spectrum. The filter groups can be set up to provide a high pass, low pass, or band pass response, depending upon the crossover type (2-way, 3-way, or 4-way). The part of the response that the specific filter is used for is the crossover band. The specific filter in the band is the filter number.

The definition of a band varies depending upon the crossover type.

- Two-way crossover
 - Band 0 = low pass section (1 low pass filter)
 - Band 1 = high pass section (1 high pass filter)
- Three-way crossover
 - Band 0 = low pass section (1 low pass filter)
 - Band 1 = mid pass section (2 filters – 1 low pass and 1 high pass)
 - Band 2 = high pass section (1 high pass filter)
- Four-way crossover
 - Band 0 = low pass section (1 filter)
 - Band 1 = mid pass low section (2 filters – 1 low pass and 1 high pass)
 - Band 2 = mid pass high section (2 filters – 1 low pass and 1 high pass)
 - Band 3 = high pass section (1 filter)

In a filter pair, the low pass filter always carries the lower filter index. So, for example:

- Filter 0 of crossover band 2 of a 4-way crossover is a low pass filter.
- Filter 1 of crossover band 2 of a 4-way crossover is a high pass filter.
- Filter 1 of crossover band 2 of a 3-way crossover is nonexistent.
- Filter 1 of either crossover band of a two-way crossover is nonexistent.
- Filter 0 of crossover band 0 of a two-way crossover is a low pass filter.

Managing BASIS via 3rd Party Control Systems

To put this to use, let's calculate the OID for a frequency of the high pass filter in a 2-way crossover that is called 'Block 11'. Simply perform the calculation as follows:

Starting bit mask:	0x40000000	
Processor ID (in hex):	0x4 00B 0000	← This is 'I' above.
Parameter Code (in hex):	0x400B 6 000	← This is 'P' above.
Band Index (in hex):	0x400B6 04 0	← This is 'X' above. (only the top 6 highlighted bits get set)
Filter Index (in hex):	0x400B6 04 0	← This is 'Y' above. (only the lower 6 highlighted bits get set)
Final OID for Freq. Control:	0x400B6040	

As another example, to calculate the OID for a frequency of the low pass filter in the mid-high band of a 4-way crossover 'Block 44', simply perform the calculation as follows:

Starting bit mask:	0x40000000	
Processor ID (in hex):	0x4 02C 0000	
Parameter Code (in hex):	0x402C 6 000	
Band Index (in hex):	0x402C6 08 0	(only the top 6 highlighted bits get set)
Filter Index (in hex):	0x402C6 08 0	(only the lower 6 highlighted bits get set)
Final OID for Freq. Control:	0x402C6080	

OID values can be calculated with decimal arithmetic. Simply perform the following calculation with the same processor ID, parameter code, and band and filter index values you obtained for the above:

$$0x40000000 + (\text{Processor ID} * 65536) + (\text{Parameter Code} * 4096) + (\text{Band Index} * 64) + \text{Filter Index}$$

The above-described calculations can only be used for the crossover.

OID Calculation for Mixer

OID Definition						
Byte 1		Byte 2		Byte 3		Byte 4
2 bits	14 bits			4 bits	6 bits	6 bits
01	IIIIII	IIII	IIII	PPPP	XXXXXX	YYYYYY

where:

- 01 = the bit mask to mark a dynamic OID (high-order 2 bits)
- I = the processor ID number from Venue Manager (14 bits)
- P = the parameter code from the parameter code table (4 bits)
- X = the X position zero-relative index (6 bits)
- Y = the Y position zero-relative index (6 bits)

To put this to use, let's calculate the OID for a the level of the crosspoint located at position (input 1, output 3) of an 8x8 mixer that is called 'Block 11'. Simply perform the calculation as follows:

```

Starting bit mask:      0x40000000
Processor ID (in hex): 0x400B0000 ← This is 'I' above.
Parameter Code (in hex): 0x400B4000 ← This is 'P' above.
Input Y Index (in hex): 0x400B4040 ← This is 'Y' above.
                               (only the top 6 highlighted bits get set)
Output X Index (in hex): 0x400B6042 ← This is 'X' above.
                               (only the lower 6 highlighted bits get set)

Final OID for Level Control: 0x400B6042
    
```

As another example, to calculate the OID for an output polarity for output 6 of an 8x8 mixer that is called 'Block 33', simply perform the calculation as follows:

```

Starting bit mask:      0x40000000
Processor ID (in hex): 0x40210000
Parameter Code (in hex): 0x4021C000
Input Y Index (in hex): 0x4021C000 (only the top 6 highlighted bits get set)
Output X Index (in hex): 0x4021C005 (only the lower 6 highlighted bits get set)

Final OID for Freq. Control: 0x4021C080
    
```

OID values can be calculated with decimal arithmetic. Simply perform the following calculation with the same processor ID, parameter code, and input and output index values you obtained for the above:

$$0x40000000 + (\text{Processor ID} * 65536) + (\text{Parameter Code} * 4096) + (\text{Input Index} * 64) + \text{Output Index}$$

The above-described calculations can only be used for the mixer.

Dynamic DSP Command Example

The dynamic DSP controls for all of the DSP processors use the 'set simple' and 'get simple' commands. Equipped with the OID calculation information and the data type in the processor parameter table, you should be able to build a command to set any dynamic control.

For example, if the gain of channel 2 of 'Block 24', a 4-channel gain block is to be set to +12dB, the following command will accomplish the task:

If you prefer decimal arithmetic:

```
OID Calculation: 1073741824 + (Processor ID * 65536) + (Parameter Code * 256) + Index
                 = 1073741824 + (24 * 65536) + (4 * 256) + 1
                 = 1073741824 + (1572864) + (1024) + 1
                 = 1073741824 + 1573889
                 = 1075315713
                 = 0x40180401
```

If you prefer bit shifting:

```
OID Calculation: 0x40000000 | (Processor ID << 16) | (Parameter Code << 8) | Index
                 = 0x40000000 | 0x180000 | 0x400 | 1
                 = 0x40180401
```

The data type for the gain parameter in the gain block is 'F' (float), as taken from the table for the gain processor, so the command to send the data to the BASIS would be (carriage returns and linefeeds added for readability):

```
<?xml version="1.0"?>
<!--0000102-->
<SET_S RT="C00" ID="0" L1PW="QSC">
<ESS O="0x40180401" M="0" F="12" />
</SET_S>
```

The length field should be calculated, not static. The reason is that as the data changes, the length of the message will also change.

Dynamic DSP Control Troubleshooting

To troubleshoot a problem in sending commands to a dynamic control in the DSP chain, it is strongly recommended that Microsoft Network Monitor or some other packet-sniffing tool be used. To work without seeing the data that is being transmitted on the wire is to work with a blindfold on. Doing some packet examination can often enable you to immediately see the source of a communication problem.

To easily get a DSP control OID and data type is to do the following with Network Monitor:

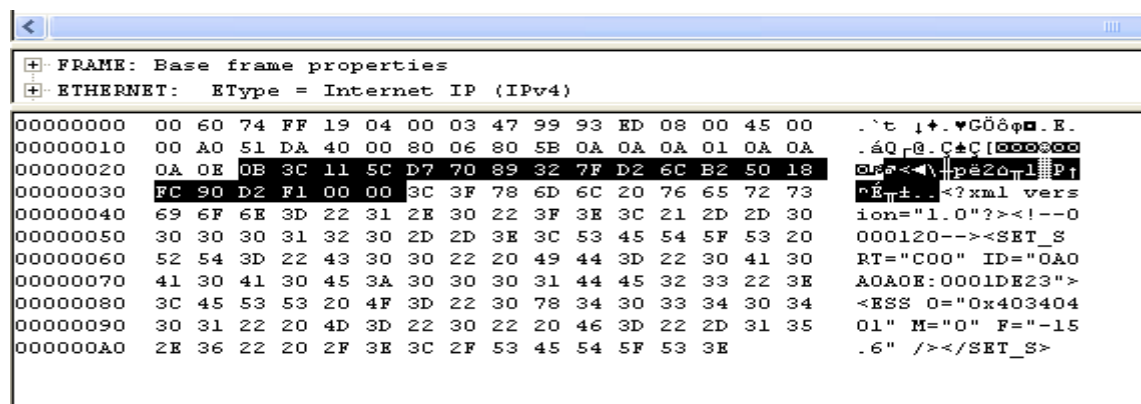
- Set up Network Monitor to see the traffic to/from your BASIS
- Start Venue Manager
- Open the properties dialog of the processor you want to control
- Start the capture on Network Monitor
- Make value changes in the properties dialog for the parameter you want to control
- Stop Network Monitor and review the capture

Network monitor will capture many packets that are for meters. At a rate of 20 meter messages per second, it is helpful to be quick on shutting down the capture, as you will have to weed through the meter packets. However, the meter packets are almost all the same length, so they are easily filtered out.

Skip to the bottom of the capture and start skipping up past meter packets until you locate a packet of a different length. If the length is real short (44 or 52 bytes), you've found a keep-alive (heartbeat) packet, or the keep-alive acknowledgement (response) packet. Find a packet that's not a meter or a keep-alive and you're likely to have found your packet.

To check the packet, examine the data segment of the TCP packet. If the data segment is not readable XML, continue on to the next packet. If it is XML, check to see if it's a '<SET_S>' or '<GET_S>' command or a '<SETACK_S>' or '<GETACK_S>' response. If not, continue to the next packet. If it is, check the OID to see if it is for the processor ID to which Venue Manager was sending commands. If so, you're there. The OID is in the message, as well as the data type, and an example of the actual data being sent to the control.

Below is an example of a partial screen capture of Network Monitor, showing a 'simple set' (<SET_S>) command being directed to OID 0x40340401 with a float value of -15.6 for reference. Note that the frame consists of (unreadable) Ethernet and TCP header data, and a data segment that is readable XML.



```
+ FRAME: Base frame properties
+ ETHERNET: EType = Internet IP (IPv4)
00000000 00 60 74 FF 19 04 00 03 47 99 93 ED 08 00 45 00  .`t |+.*G00p.E.
00000010 00 A0 51 DA 40 00 80 06 80 5B 0A 0A 0A 01 0A 0A  .AQ-r.C+C[000000
00000020 0A 0E 0B 3C 11 5C D7 70 89 32 7F D2 6C B2 50 18  [P*-<V||p#20Tl|P|
00000030 FC 90 D2 F1 00 00 3C 3F 78 6D 6C 20 76 65 72 73  [E+...<?xml vers
00000040 69 6F 6E 3D 22 31 2E 30 22 3F 3E 3C 21 2D 2D 30  ion="1.0"?><!--0
00000050 30 30 30 31 32 30 2D 2D 3E 3C 53 45 54 5F 53 20  000120--><SET_S
00000060 52 54 3D 22 43 30 30 22 20 49 44 3D 22 30 41 30  RT="COO" ID="0A0
00000070 41 30 41 30 45 3A 30 30 30 31 44 45 32 33 22 3E  A0A0E:0001DE23">
00000080 3C 45 53 53 20 4F 3D 22 30 78 34 30 33 34 30 34  <ESS O="0x403404
00000090 30 31 22 20 4D 3D 22 30 22 20 46 3D 22 2D 31 35  01" M="0" F="-15
000000A0 2E 36 22 20 2F 3E 3C 2F 53 45 54 5F 53 3E      .6" /></SET_S>
```

Managing BASIS via 3rd Party Control Systems

Keep Alive Messages

To keep the TCP connection to the BASIS open, it is necessary to transmit data to the BASIS on a periodic BASIS. Depending upon the control device's programming and purpose, this can be accomplished in a couple ways:

- A transmission of a GET request can be made, or
- A 'keep-alive' message can be sent.

Either of the above messages will keep the connection to the BASIS open, if sent every two seconds or less. Sending the message every second and a half (1,500ms), offers a very wide safety margin.

Sending a get request is a perfectly valid approach to keeping the connection alive. If BASIS controls for a user interface at the control device must be polled, this works out very well. As long as the control device polls for control values in a time frame that keeps the connection open (2 seconds or less), keep-alive commands are not required.

Sending keep-alive messages is a good approach for control devices that will register for control changes, and thus will not poll for control values. In this case, the control device will generate no traffic unless it is sourcing a control change, and will receive no traffic unless a control change occurs within a registered control in the BASIS. To keep the connection open, some sort of traffic will have to be sent every two seconds, and the keep-alive is designed for just that purpose.

WARNING! Do not use a SET request command to perform the keep-alive function. Using a SET request command will prevent the BASIS from triggering an automatic save of control values. The automatic save function of the BASIS is triggered after one minute of inactivity. Constant control changes will prevent the required period of inactivity for the automatic save to occur. See the BASIS manual for details on the automatic save function.

Keep Alive Command

To keep the connection to the BASIS open during periods of inactivity, the keep-alive command is issued. An example keep-alive follows.

```
<?xml version="1.0"?><!--0000071--><KA RT="C00" ID="0" L1PW="qsc"></KA>
```

A response to this message will be as follows:

```
<?xml version="1.0"?><!--0000068--><KA_ACK RT="C00" ID="0" ></KA_ACK>
```

An alternate form of the keep alive can be sent which will keep the connection open, but not generate a return Acknowledge message:

```
<?xml version="1.0"?><!--0000044--><KA></KA>
```

There are no portions of the string that must be dynamically created. The command can be safely hard-coded into a device.

There is no response generated by the BASIS. The only result will be that the TCP connection will be kept open for another two seconds.

Control Registration

The registration of an OID with the BASIS allows a control device to receive OID notifications, asynchronously, whenever the value of either a static or dynamic control / OID changes. This allows the control device to be free to do other tasks, instead of spending time polling for control changes.

The BASIS will continue to send notification messages for all registered controls to the control device until one of the following situations arise:

Managing BASIS via 3rd Party Control Systems

- The BASIS is rebooted or power cycled.
- The client fails to keep the connection open with periodic activity.
- The active Configuration is changed in the BASIS.

The registration information is not stored, since it is information related to the handling of data for a TCP connection. The TCP connection is gone, by definition, when any of the above occurs. The control device must be sensitive to these conditions, and take steps to re-register the controls when the registration is canceled.

When using control registration, the amount of message traffic to/from the BASIS will sometimes be quite minimal. In fact, it is quite possible that no messages will be sent or received for days, depending upon the particular control scenario. To keep the TCP connection to the BASIS open, use keep-alive messages on a periodic BASIS. For complete details, see the section entitled Keep Alive Messages.

Control Registration Command

The control registration command can be issued for any static or dynamic OID.

```
<?xml version="1.0"?>
<!--000000-->
<SET_REG RT="C00" ID="0" L1PW="password">
<REG_RATE S="50"/>
<REG O="0x00000000" S="1"/>
</SET_REG>
```

There are five portions of the string that must be dynamically created:

- The first (represented by “000000”) is the length of the entire message.
- The second (represented by “password”) is the Ethernet connection password for the BASIS unit.
- The third is the registration rate, which can act as a hold off for high data rate messages, such as meters. Valid selections are 50, 100, 250, and 500ms.
- The fourth (represented by “00000000”) is the OID number (in hex format) that is being registered.
- The fifth is the registration state. The state is a short data type. A value of 1 will register the control, and 0 will un-register the control.

The response will have both the response code (as indicated by the “R” value) and a return value that indicates the registration state of the control.

```
<?xml version="1.0"?>
<!--000000-->
<SETACK_REG RT="C00" ID="0" L1PW="password">
<REG O="0x00000000" R="0" S="1"/>
</SETACK_REG>
```

After the registration of a control has been completed (as confirmed by a successful SETACK_REG response), control updates will arrive whenever the control in the BASIS is changed. The control device must be prepared for unsolicited TCP messages to arrive at the rate specified in the REG_RATE.

The control update messages arrive in the form of SETACK responses, of the data type of the OID that was registered.

Managing BASIS via 3rd Party Control Systems

So, for instance, if an output mute was registered and the output mute setting was changed by another source (perhaps Venue Manager) to un-mute the output, then the message that would arrive would be:

```
<?xml version="1.0"?>
<!--000000-->
<SETACK_S RT="C00" ID="0" L1PW="password">
<ESS O="0x01120000" R="0" M="0" B="false"/>
</SETACK_S>
```

This message is the same type of message that would be returned had the control device instigated the change through a SET request. As such, the message can be processed through the exact same logic as the normal SETACK responses.

If it is desirable for the messages to be traced to their source, the ID attribute of the message can be employed for message identification. The control device can place source identification into the ID attribute, as shown below:

```
<?xml version="1.0"?>
<!--0000102-->
<SET_S RT="C00" ID="0A0A0A0A:00000001" L1PW="QSC">
<ESS O="0x40180401" M="0" F="12" />
</SET_S>
```

In the above, the IP address is 10.10.10.10 (0x0A0A0A0A), and the sequence number is 1. The formatting of the ID is solely up to the user, and can contain up to 20 characters. The response message will bear the ID of the message that caused it. By examining the response messages for a particular ID, the source of the change can be determined and if desired, alternative action can be performed.

Programs and hardware supplied by QSC do use the ID attribute for message identification. When reading the ID attribute, be prepared to receive the maximum number of characters.

Troubleshooting

During development, things can and do go wrong. Some of the more obvious problems that a developer may encounter are listed below.

Problem	Cause	Solution
<p>Controls at the third party device are sluggish, system seems unresponsive.</p> <p>Audio may not track control changes at the UI.</p>	<p>Polling rate is too high.</p>	<p>Slow the polling rate. If the polling rate exceeds the overall throughput of the controlling device and the BASIS, backup and loss of messages will result.</p> <p>Start slow, and then speed things up when it is working properly.</p> <p>Don't poll faster than the BASIS and network can handle. A network monitor can help in making this determination.</p>
	<p>Data rate is too high.</p>	<p>If the device sources data to the BASIS too quickly, a flood of data can be put onto the network. This is bad for everyone in the same collision domain.</p> <p>Gate the outgoing XML commands so that they are presented at a reasonable rate. A good place to start is every 50-100ms during active changes.</p>
<p>Controls at the third party device are sluggish, system seems unresponsive.</p> <p>Audio does track control changes at the UI.</p>	<p>Polling rate is too low.</p>	<p>Speed up the polling rate to allow the UI to show the control values in a timelier manner.</p>
<p>Controls at the third party device do not update.</p> <p>Audio does track expected control changes.</p>	<p>No control loop closure.</p>	<p>Sending a command is only half of the control loop. Getting a confirmation of the change is the other.</p> <p>Add control polling, or control registration to close the UI loop.</p>
<p>BASIS does not respond to issued XML commands.</p>	<p>No connection.</p>	<p>Verify the IP address and port number. The IP address can be viewed on the front panel of the BASIS, and the port number should be set to 4446 decimal.</p>
	<p>No connection.</p>	<p>A TCP/IP error has occurred during connection.</p> <p>Make sure the device can be pinged. If not, the network is not set up correctly.</p>
	<p>No connection</p>	<p>The BASIS password is incorrect.</p>

Managing BASIS via 3rd Party Control Systems

		Verify that the <i>Front Panel Password</i> has been set in Venue Manager, and make sure the password used in the control device matches.
	No connection	The BASIS is not turned on.
	No connection	The BASIS is not attached to the network. Verify that QSControl link status and activity lights are functioning on the BASIS, and that the BASIS is plugged into the Ethernet.
BASIS connects on port, but does not respond to commands.	Single-wire CobraNet being used.	The Third Party connection port is only available through the QSControl (10Mb) Ethernet port. Connection through the single-wire CobraNet arrangement is not supported. Connect all control processors to the LAN to which the QSControl port is connected.
BASIS returns an error code in the "R" parameter in the acknowledgement.	Bad XML format	No linefeeds or carriage returns are allowed. View the XML with a network analyzer to verify it is correct on the wire. Microsoft Network Monitor works very well, but be sure the monitor and the devices are all in the same collision domain.
	Bad XML content	The XML is well formed, but the control values are not valid. Make sure that all data is within bounds.
	Bad OID value	Make sure the OID being passed in the command is in the table of valid OIDs.
BASIS does not load / run expected Configuration or Snapshot.	Event manager not set up appropriately.	The BASIS can be set to respond to external events via the OMNI inputs. When the events are set up to load a Configuration or Snapshot, the event will override any Configuration or Snapshot commands issued through this API.
	Configuration or Snapshot number not sent in zero-relative form.	Configuration and Snapshot numbers appear in Venue Manager as 1 relative. The API accepts data in zero relative form. For example, if a Configuration has 10 Snapshots, the valid Snapshot numbers are 0-9. If the Configuration is the first one in

Managing BASIS via 3rd Party Control Systems

		the device's flashed Configurations, it is Configuration number 0.
	Improper command format is being used.	The Configuration and Snapshot commands are complex commands with different syntax than simple commands. Make sure the complex command format is being used.
Control changes in Venue Manager don't show up at the controller UI.	Controls not polled or registered.	The control values shown in the controller UI must be polled or registered so that external changes that are made to the controls will cause an update to be sent to the controller.
	Registration has expired.	When the TCP connection is closed (by either side), the controls need to be re-registered.
	Configuration changed.	When the active Configuration has changed, all controls must be re-registered. The active Configuration number can be registered, so that it can be monitored.
BASIS mutes audio unexpectedly	Configuration change	When the active Configuration is changed, the BASIS mutes audio. This is normal. For a change to the BASIS that does not mute audio, issue a Snapshot change.
BASIS not producing expected audio after a Snapshot change.	Snapshots are not set up correctly for the desired use case.	Snapshots can include any number of controls, and no assurance is made that there is any uniformity from Snapshot to Snapshot with regard to the controls that are in the Snapshots. Thus, moving through a series of Snapshots in one order is not assured to be equivalent to moving through the series in another order, unless the user makes sure to include the affected set of controls in all of the Snapshots. Control changes resulting from Snapshot changes are cumulative and potentially non-overlapping control-wise.

Appendix A – Configuration Contents

The following is a list of all parameters types that are included in a BASIS Configuration. The sections are based upon the major subsystems of the BASIS. This table is provided to give a general understanding that there is a separation of data in the device that makes certain types of changes easy to perform.

The detail list of the processors that can be stored in the DSP chain are shown in Appendix C.

Config Section	Purpose
Configuration-Wide Settings	<p>Items that must be set constant while the device is running a particular Configuration. These items can (and often do) vary between Configurations.</p> <p>Configuration-wide settings, when changed, can cause the device to stop sending or receiving audio or stop communicating with Venue Manager or any connected controllers.</p>
DSP Signal Flow	<p>The user programmable signal chain. This must remain constant while a device is running a Configuration, and any Snapshots associated with that Configuration. The DSP chain can (and often does) vary between different Configurations that reside in the device.</p>
Snapshot Settings	<p>The device controls that are allowed to change. This can include controls in the DSP chain as well as a select number of Configuration settings. Attachment B covers the controls that can be used in Snapshots in detail.</p>

What's not in a Configuration:

Global Section	Purpose
Device Setup Data	<p>The IP Address, subnet mask, default gateway. The device name, passwords.</p> <p>These items remain the same, regardless of the Configuration or Snapshot that is active in the device.</p> <p>These items can't be changed unless the user changes them through the "device properties" in QSC's Venue Manager program, through Telnet or serial setup.</p>

Appendix B – Snapshot Parameters

The following is a list of all possible parameters that can be contained in a Snapshot, including Configuration settings, as well as DSP settings.

Venue Manager Page	Parameters
Inputs	Analog input sensitivity.
DSP	See processor table.
Levels	All levels and mutes.
Amp	A/C Power, Gains, Mutes for all amps.
Outputs	Nothing.
Triggers	All relay and logic outs.

Since the Configuration of the DSP chain is completely user-programmable, a specific Snapshot may include zero, one or many of the following parameters.

Processor	Parameters
AGC	Threshold, Target Gain, Release, Ratio, HP Filter and Auto Set.
Automixer	Input Mute, Priority, and Level. Program Reference Level and on/off. Priority Gain, Hold and Response. Automix Range, Mix Level and Mute.
Compressor	Threshold, Ratio, Soft Knee, Attack, Release, Output Gain, Mute, Polarity, Bypass
Crossover	Frequency for all bands
Delay	Delay Time, Bypass
Ducker	Threshold, Priority Gain, Fade Out, Hold, Fade In, and Output Attenuation.
Dynamics Processor	Compression Threshold, Expansion Threshold, Noise Gate Threshold, Compression Ratio, Expansion Ratio, Soft Knee, Attack, Release, Output Gain, Mute, Polarity, Bypass
Gain	Gain, Mute, Polarity for all channels
Graphic EQ	Gain, Q Optimization, Bypass all
Input	Analog input level.
Mixer	Input Level, Input Mute, Crosspoint Level, Crosspoint Mute, Output Gain, Output Mute, Output Polarity
N-Band EQ	Frequency, Q, Low Mid and High Gain, Bypass for each filter
Noise Generator	Level, Mute, Type
Output	Output level and mute.
Peak Limiter	Threshold, Attack, Release, Output Gain, Mute, Polarity, Bypass
RMS Meter	Averaging Time
Router	Output Select.
Tone Generator	Frequency, Level, Mute, Polarity

Appendix C – DSP Processors

The following is a brief list of all DSP processors that can be in a BASIS DSP chain. An actual signal chain may include any combination of the processors in this list (including none), up to the limitations imposed by the BASIS model that the signal chain is to reside within.

The primary limitation is overall DSP resources, but for some BASIS models the number of input or output channels vary – thus restricting the number and Configuration of inputs and outputs, respectively. For the specific limitations imposed by a particular BASIS model, please consult the product documentation for the model in question.

Processor	Snapshot Capable
Compressor	Yes
Crossover	Yes, except for filter type and filter order
Delay	Yes, except for max delay
Dynamics Processor	Yes
Gain	Yes
Graphic EQ	Yes
Input	Yes
Mixer	Yes, except for active crosspoint matrix
N-Band EQ	Yes, except for filter order
Noise Generator	Yes
Output	Yes
Peak Limiter	Yes
RMS Meter	Yes
Tone Generator	Yes

Appendix D – XML Overview

The XML structure that is used with BASIS is slightly different than the typical XML a designer might be accustomed to using. This section outlines the differences that will be encountered and documents the rules that are enforced on the XML that is sent to the device.

XML Structure

The typical XML structure that is used for most applications is very similar to HTML, inasmuch that it is tag-based and the text to be used (what will be called the payload) appears within the tags. An example of this follows:

```
<XML_TAG>PAYLOAD</XML_TAG>
```

The BASIS often uses a different syntax for its XML, which is referred to as “empty element” syntax. In this syntax, the XML tag pair does not actually carry any data in the text area, but carries the data as attributes of the tag itself. An example of the above XML as expressed as an empty element is:

```
<XML_TAG na="PAYLOAD"/>
```

Where “na” is a predefined attribute tag that would have been defined. When there is hierarchy to an XML message, this syntax is extended slightly due to the normal structuring of XML when child tags reside within a parent:

```
<XML_TAG na="PAYLOAD">  
  <OTHER_TAG na="PAYLOAD2"/>  
</XML_TAG>
```

XML Formatting

It is possible that the syntax of the XML from the control device will be incorrect if a high-level XML class is used to generate the XML that is sent to the BASIS. The reason for this could be that the XML class can insert extra characters that are not expected by the BASIS. Also, it is important that the resulting differences between attributes and XML text are understood since the two generate XML that is quite different. Often a high level XML generator will not use empty element syntax, and will generate starting and ending tags with no text.

The BASIS will only process the syntax it is expecting to see. So for example if the following is the expected format:

```
<XML_TAG na="PAYLOAD">  
  <OTHER_TAG na="PAYLOAD2"/>  
</XML_TAG>
```

This format will not be accepted, even though it has the same meaning.

```
<XML_TAG na="PAYLOAD">  
  <OTHER_TAG na="PAYLOAD2"></OTHER_TAG>  
</XML_TAG>
```

If simple text substitution into predefined XML text strings is being performed in the controlling device and the resulting strings are being sent, this should not be an issue as there is no XML formatting being performed. The examples in this document can be used as a starting point, since they contain the proper XML syntax.

Managing BASIS via 3rd Party Control Systems

As noted elsewhere in this document, the XML that is sent to the BASIS should not contain any carriage returns or linefeeds. The XML in this document is formatted for readability. Any XML that contains carriage returns or linefeeds will be rejected by the BASIS.

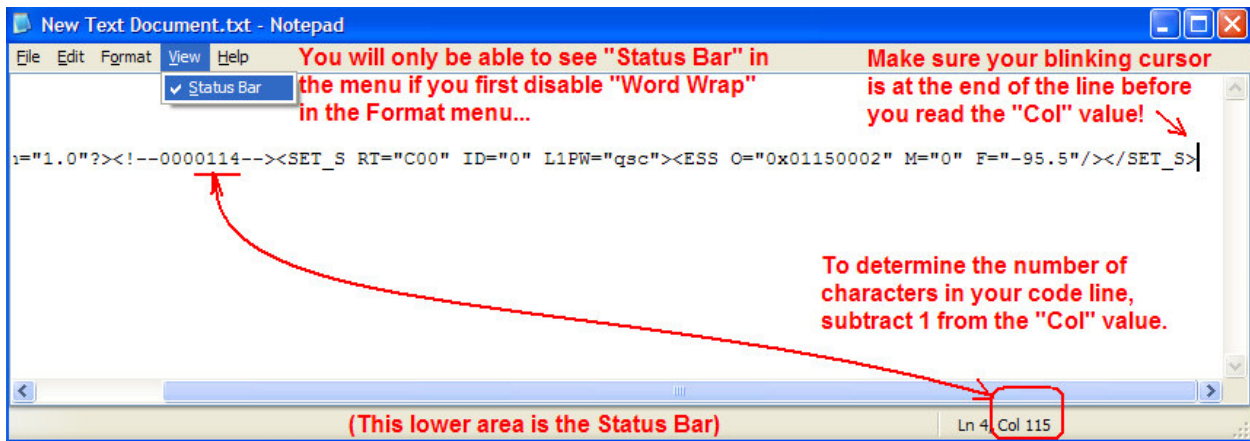
NOTE: XML is a text-based data format that can represent a variety of data. Due to its human-readable form and flexibility, QSC has decided to use XML to communicate with the BASIS 922az. This means that systems with basic string manipulation functions should be easily configured to communicate with all QSC BASIS devices.

There have been many good books written on XML and a good introduction to XML can be found at http://www.w3schools.com/xml/xml_what.asp on the Internet. For specific details on the XML used in the BASIS, see Appendix D – XML Overview.

Appendix E - Using a HyperTerm telnet session to send/receive Basis data

You can use Notepad and HyperTerm (they both come with Windows) to send XML strings directly to a Basis; this is another way (besides BasisTerm) to build up a string and test it. Your HyperTerm connection to a Basis would essentially be a telnet session using port 4446, as opposed to a serial connection. But to create the XML strings you cannot type directly into the HyperTerm window. Yes, you could copy the strings from BasisTerm directly into HyperTerm, but this discussion is how to do this without using BasisTerm.

Use the 3rd Party document to build each string, typing them as one long string in the Notepad window (turn off "word wrap"). Make sure you count the number of characters carefully. Notepad has a built-in feature (View>Status Bar) to help count characters but you must first turn off "word wrap", otherwise you can't see the Status Bar. Then copy (Ctrl-C) that line of code from Notepad and send it via HyperTerm using the "Paste to Host" command.



It won't work to "Paste" (Ctrl-V) the strings into the HyperTerm window itself, nor can you type the strings directly into the HyperTerm window; you have to use the "Paste to Host" command from the HyperTerm menu instead. The HyperTerm window will then show the response message from the Basis device.

Appendix F - Managing Global Control Objects

Abstract

This document describes the global control objects that are included in QSC's control and monitoring protocol and how these objects can be managed through the BASIS Third Party Control Interface. Though there are any number of possibilities available for designing a control system with various hardware components, this document is focused exclusively on the software implementation (third party XML coding) required to manage QSC's global control objects. To that end, this document provides a number of working XML examples for managing specific global control object parameters and we include all associated object identifiers necessary for the third party programmer to create their own application code.

1.0

Prerequisites:

Though this document discusses QSC's implementation of global control objects, it does so at a fairly high level. It is expected that the system designer and/or third party programmer is familiar with the global control objects discussed in this document and that the system designer and/or third party programmer knows how to configure these objects through the QSControl.net suite of applications (namely, QSC's Venue Manager application). This paper is only concerned with the management of these objects through the BASIS Third Party Control Interface... it does not provide a detailed discussion on the instantiation or configuration of these objects. If you are unfamiliar with QSC's implementation of global control objects or how to set them up in a "Venue" for subsequent management, please read the Venue Manager "Help" files in QSControl.net versions 3.00.569 or later. Specifically, read the Help file sections entitled "Masters" and "Global Presets".

1.1

Prerequisite: Supporting Firmware

Global control objects were introduced in the QSControl.net suite of applications with the public release of QSControl.net version 3.0. Product firmware was included in the 3.0 software release to support global control objects in BASIS 700 and 900 series products, RAVE 500 series products and DSP 300 series products. Regardless of whether any of the applications in the QSControl.net suite of applications are ultimately used for managing BASIS platform products during runtime, all products must have 3.x.x firmware installed in order to support global control objects. Venue Manager will prompt the system designer or end user if product firmware is out-of-date. A firmware utility is included (along with all product firmware files) to perform any necessary updates in support of global control objects or any other features specific to the 3.0 software release. Refer to the Help files or on-screen prompts in the 3.0 software for assistance in updating product firmware.

1.2

Prerequisite: Appropriate Control System

When we speak of third party control systems in this document, we are speaking of any external system that is deployed for the intent of managing one or more BASIS platform products in a "Venue" by implementing a message exchange service with the BASIS products via QSC's XML based protocol implementation in the BASIS Third Party Control Interface. The third party control system could be a simple networked wall-mounted controller or it could be a system of multiple purpose-built controllers such as a group of networked touch panels or rack-mount

Managing BASIS via 3rd Party Control Systems

control processors. A third party control system could also be a custom Windows application running on a PC or a portable handheld device. No distinction is made in this document as to the type of third party control system deployed. However, the third party control system must be able to participate on a QSControl network and it must be able to provide a message exchange service that is compliant with QSC's XML based protocol. Note that QSC's protocol is implemented over TCP/IP and requires access to TCP port 4446.

2.0 A Brief Consideration of Available Tools:

Before discussing third party management of global control objects, it is worth spending a few moments to consider the options available for managing these objects through the QSControl.net suite of applications. The QSControl.net suite of applications may offer the tools you need to do the job at hand without the need for a third party control system. A number of new capabilities were introduced in the QSControl.net 3.0 release to address the needs of the more sophisticated management systems. Needs that were previously addressed through the use of third party control systems. For example, with the introduction of "QScreator", the QSControl.net suite now offers the tools to create custom control and monitoring panels or complete custom GUIs that are venue-specific and even user-specific. These user interfaces can be designed with advanced visual components that provide a look and feel that is polished and consistent with other interfaces in the system and they can be created in a way that provides an interface that is simple to use, intuitive and informative for the end user. Collectively, QScreator and QSCAD offer the system designer the ability to create venue-specific interfaces with limited end user accessibility on a per-screen or per-control basis. The 3.0 release of Venue Manager also offers automatic network discovery of BASIS platform products deployed on one or more subnets. And all of these applications now support global control objects. Global control objects, such as Masters and Global Presets, provide a means for managing multiple products (or all products) from a single control... effectively providing "venue-wide" management. Venue-wide management will become even more flexible when QSC's NAC-100 networked wall-mount controller is introduced in late 2007 or early 2008. To address the needs of "mission critical" systems, QSControl.net's "Notify" utility can be linked to specific control objects to provide "unsolicited" alerts when specific events occur. These alerts can then be sent to a system administrator or to a member of a maintenance staff via e-mail.

As a result of the capabilities introduced in QSControl.net 3.0, there should no longer be a need to deploy a third party control system just to address venue-wide or remote management concerns or to address the needs of wall-mount controllers. In addition, there should no longer be a need to deploy a third party control system just to address the need for an automated alert system or to provide an enhanced graphical user interface or to limit end user accessibility.

But of course, the need for third party control systems hasn't been eliminated. A third party control system can provide a single user interface that manages products from multiple manufacturers. For example, when managing a venue that includes managed audio, video and lighting products. Some third party control systems may also be a better fit for interfacing to specific life safety systems. And of course a custom designed third party interface may be a better choice if data encryption or advanced security measures are required at the user interface. Third party control systems may also be a better choice when portable (wireless) controllers are required or if specific display and/or graphics elements are required that are only available with purpose-built control products. Since these third party applications do exist, the 3.0 release of QSControl.net continues to support the BASIS Third Party Control Interface, otherwise known as the BASIS API or the 3rd party control port.

3.0 Global Controls - Overview:

QSC provides the BASIS Third Party Control Interface as a means for "managing" BASIS platform products from a third party control system. The BASIS Third Party Control Interface is not a replacement

Managing BASIS via 3rd Party Control Systems

for Venue Manager or any other system design and/or system configuration tool in the QSCControl.net suite of applications. It is therefore expected that the system designer will use the QSCControl.net tools to create and to configure their “Venues” and all products contained within these Venues. Venue Manager must be used to define the behavior and performance of all complex controls resident within each BASIS platform product. This includes defining the audio signal flow for each “Config” (BASIS signal flow configuration) and defining the objects to associate with each Snapshot (defining which objects to “Mark”). Likewise, Venue Manager must be used to create and to configure global controls such as Masters and Global Presets. Once all of this “up front” work is completed, the BASIS Third Party Control Interface can then be used to manage the BASIS platform products from a third party control system.

The QSCControl.net 3.0 release introduced two global control objects, these are the “Masters” objects and the “Global Preset” objects. Both of these objects allow an end user to affect a change in multiple BASIS platform products through a single control.

The Masters control objects allow the end user to affect a change in multiple objects of the same type so long as the objects include value ranges... objects such as output gain levels for example. In a typical application, a Master control is created and then multiple “Slave” control objects are associated with the Master. The Slave control objects can reside in one or more BASIS platform products. When the Master control is exercised, all Slave control objects are manipulated in the manner defined when the Master control object was created. Master control objects can affect Slave control objects in a *single* and specific Config within each BASIS unit or they can affect Slave control objects in *all* Configs within a BASIS.

The Global Presets allow the end user to recall “Config and Snapshot” combinations, which are resident in one or more BASIS platform products, using a single control. Global Presets can also assign a starting value to any Masters control objects associated with the recalled Config. The combination of Config, Snapshot and Masters control objects can be viewed as the operating “scene”... essentially the combination of all control objects currently operating in a given BASIS product. In a typical application, a single Global Preset control could be used to recall “Config 1 and Snapshot 3” in one BASIS product and to recall “Config 4 and Snapshot 7 and assign Master 3 a value of +3dBu” in another BASIS product.

In QSC’s implementation, global controls reside outside of the BASIS hardware. That is, the actual Master or Global Preset *control* is a part of the control system. The control system could include one of the applications in the QSCControl.net suite of applications or it could be a third party control system. It is the subordinate *control objects*, such as Slave control objects associated with a Master control or recipient control objects associated with a Global Preset control, that reside in the BASIS platform product. When a Master control or a Global Preset control is exercised, independent messages are directed at each BASIS product, which notify the subordinate control objects as to the “new” state of the Master control or Global Preset control. The subordinate control objects then respond accordingly and execute the expected change (if any changes are expected).

Managing BASIS via 3rd Party Control Systems

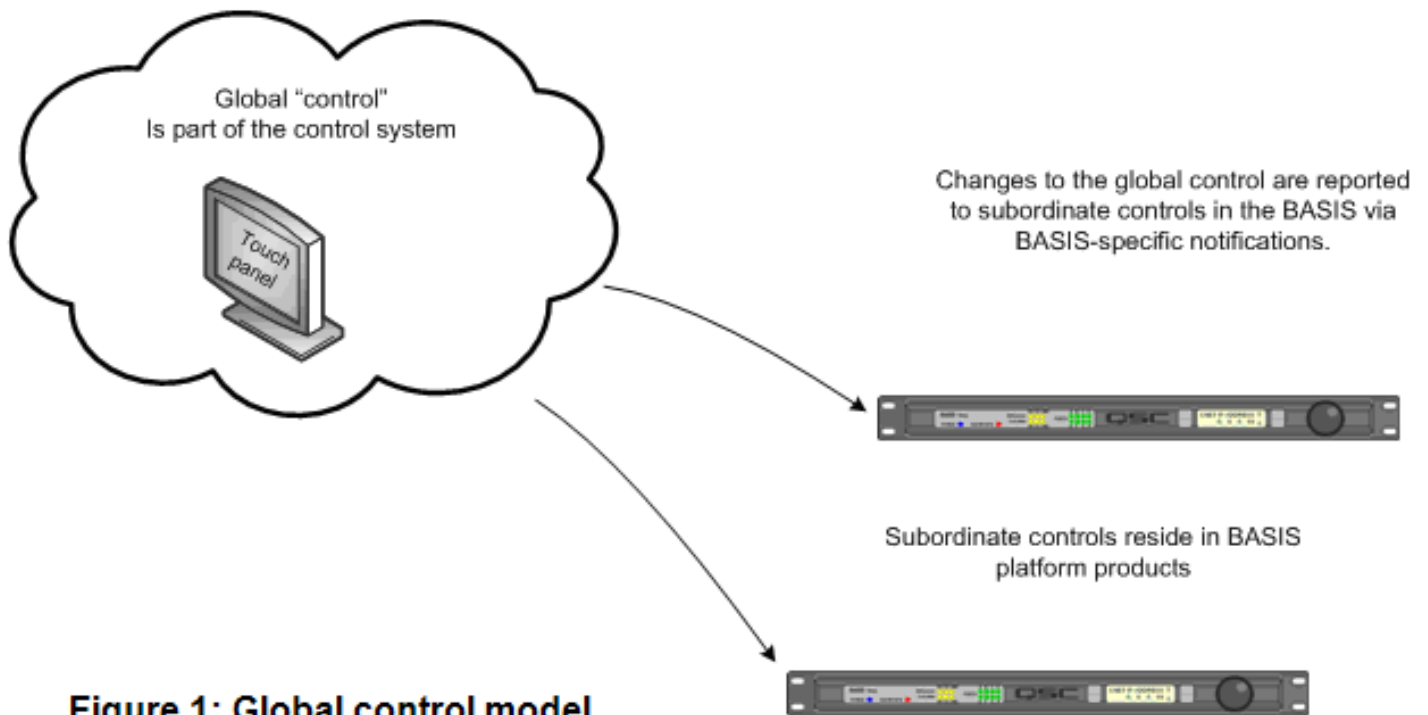


Figure 1: Global control model

Figure 1 illustrates the basic global control model. The Figure shows a global control hosted on a third party control system. Subordinate control objects (Slave control objects or a Config and Snapshot combination associated with a Global Preset) reside in the BASIS products. Changes to the global control are reported to each BASIS unit that includes subordinate control objects associated with the global control. It is important to note that each and every BASIS associated with a global control must be notified when its associated Master control changes value or when its associated Global Preset is recalled. From the perspective of a third party control system, the power of global controls is not in reducing the quantity of messages sent over the network to manage multiple BASIS products. The power is in the reduction or elimination of data management. Since all information that associates a subordinate control object to a global control is ultimately stored in the BASIS product, the control system is freed from having to store and manage this information. The control system can simply issue notifications that report the state of the Master control or the Global Preset control and let each BASIS process these notifications accordingly. Though global controls were not architected as a means to reduce message overhead or network traffic, the impact in this area should not be understated since it is possible to realize a significant improvement in efficiency by affecting multiple Slave control objects within a BASIS product with just a single global control message.

BASIS products do more than just react to global controls. For Master controls, each BASIS device keeps track of its own Slave control object settings relative to the Master control. For example, for a Master control that manages DSP Output Levels, each BASIS would keep track of its own DSP Output Levels and DSP Output Level offsets (deltas) with respect to the Master control. Each BASIS device also maintains a local record of the Master control's current value. Therefore, if a Master control were to change value by +3 dBu, each BASIS device would have the information it needs to be able to manipulate its Slave control objects (its DSP Output Levels) to either increment appropriately or to track the exact value of the Master control, depending on the directive issued from the control system. For Global Presets, each BASIS device keeps track of its Config and Snapshot combination that is to be recalled when the Global Preset is executed. Each BASIS device also keeps track of any information associated with a Master control that is part of the Global Preset. Therefore, when a Global Preset is executed, each

Managing BASIS via 3rd Party Control Systems

BASIS can recall the appropriate Config and Snapshot combination and quickly apply the value assigned in the Master control to the appropriate subordinate control objects in the BASIS.

3.1 Global Controls: OIDs for Master Controls

Table 1 provides a list of supported OIDs (object identifiers) related to Master controls. The elements of the OIDs used for managing Master controls is also presented. Note that only three control parameters are shown in Table 1. If a third party programmer requires access to parameters not included in Table 1, it is recommended that they contact QSC's Technical Services Department for further assistance.

Table 1 shows the OID structures used to manage Master controls. The structures form the basis of the OIDs that are encapsulated in messages sourced from a control system to request specific action from BASIS devices hosting Slave control objects. These OIDs are also encapsulated in response messages that are returned from a BASIS back towards the requesting control system. Each OID structure in Table 1 has a specific function related to a specific type of Master control. Embedded in each OID is the ID of the Master control and a parameter code that specifies the "type" of Slave control object addressed in the message and/or the "directive" to enact on the Slave control object.

Note that Table 1 segments the OID	Prefix for all Master controls (bits 31-16)	IDX1 (bits 15-13)	IDX0 (bits 12-0)
Name	0x06 03	0x0	0x0 – 0x1FFF
Level	0x06 03	0x1	0x0 – 0x1FFF
Sync Slave to Master	0x06 03	0x3	0x0 – 0x1FFF

Table 1: OID structures associated with Master controls

All BASIS OIDs are represented as 32-bit values in hexadecimal form. All OIDs associated with Master controls have a prefix of 0x0603. The least significant 16 bits of each OID indicate the ID of the Master control object and the control parameter to be exercised. The ID of each Master control object is unique and is stored in each BASIS product that hosts one or more Slave control objects associated with the Master control. Note the difference between a "Master control" (which lives on the control system) and a "Master control object" (such as the name of a Master control or the ID of a Master control that lives in a BASIS that has an associated Slave control object).

The Master control ID and the Master control parameter code are represented in the IDX0 and IDX1 fields of the OID, respectively. Note that these two fields do not fall on convenient 4-bit boundaries. IDX1 is represented in 3 bits and IDX0 is represented in 13 bits. As a result, the hexadecimal representation in the OID for the Master ID value reflects the result of the IDX1 field concatenated with the leading bit of the IDX0 field (e.g., 011 concatenated with 0 becomes 0110 binary or 0x6 in hexadecimal... see the table below). Table 2 shows a breakdown of these two fields in binary format. Table 2 also provides some OID examples using various combinations of values for IDX0 and IDX1.

Comments	IDX1 (bits 15-13)	IDX0 (bits 12-0)	OID (in hex)
Base OID structure (template)	000	0 0000 0000 0000	0x06 03 00 00
OID for parameter 0 and Master ID = 6	000	0 0000 0000 0110	0x06 03 00 06
OID for parameter 1 and Master ID = 3	001	0 0000 0000 0011	0x06 03 20 03
OID for parameter 3 and Master ID = 20	011	0 0000 0001 1101	0x06 03 60 1D

Table 2: Binary breakdown of OID examples using various combinations of IDX0 and IDX1

As shown in Tables 1 and 2, the OID structure for Master controls supports up to 8K (8,191) unique Master control objects per Venue. The IDX1 parameters that are currently supported through the BASIS Third Party Control Interface address three basic functions: GET the name of a particular Master control object, GET or SET the level of a Master control object, SET the Slave control objects to be equal in value to that of a particular Master control object.

The *name* function, which is available by setting the IDX1 field in the OID to “000”, allows a secondary controller to “GET” the name associated with a Master control object when the Master’s ID is known. This function is probably of little value in single controller systems. However, it is useful in systems with multiple third party controllers and in systems where QSCControl.net tools share the network with third party controllers.

The *level* function, which is available by setting the IDX1 field in the OID to “001”, allows a Master control to manipulate all associated Slave control objects in a Venue by a relational amount. The message used for the “SET” level control function includes a new value assigned to the Master control object. The new Master value is used by BASIS devices to determine the amount of change to apply to their Slave control objects. The BASIS is able to determine the value change amount because it maintains information as to the previous Master value in relation to the new Master value as well as information on the Slave control object’s value relative to the Master control. For example, if a Master control (let’s say a Master gain) is initially set to +2.5 dBu and one of its associated Slave control objects is set to -56.3 dBu, a new Master value of +7.6 dBu would produce a new value in the Slave control object of -51.2 dBu. With both the initial Master value of +2.5 dBu and the new Master value of +7.6 dBu, the Slave control object maintains an offset of -58.8 dBu relative to the Master control.

Note that even though a gain control is used to illustrate the level functions discussed in this document, the level function applies to all supported controls in BASIS products so long as they are represented with a range of values (typically level controls, frequency controls etc.).

The “GET” level function is most useful in systems with multiple controllers. A secondary controller can poll a Master control object to obtain its current state (the value of the “level” parameter). However, a better practice is to set up secondary controllers to register any Master control objects associated with Master controls that are hosted on remote control devices. Doing so provides the secondary controller with automatic updates when changes occur to the registered object (refer to the BASIS Third Party Control Interface document).

The *sync* Slave to Master function, which is available by setting the IDX1 field in the OID to “011”, allows a Master control to “SET” all associated Slave objects in a Venue to a value equal to that of the Master control. For example, if a Master control is set to a value of +9.3 dBu and

Managing BASIS via 3rd Party Control Systems

the Master issues a sync directive to all associated BASIS Slave objects, the Slave object's values will be changed to +9.3 dBu.

Note that Slave objects can only track a Master control if they are aware of changes to a Master control. This requires that a control system issue appropriate messages to all BASIS products on a network that host Slave control objects that are intended to track a given Master control. This is handled automatically within the QSCControl.net suite of applications... but there is no "automatic" facility for doing so through the BASIS Third Party Control Interface. Therefore, it is up to the third party programmer to ensure that appropriate communication is provided to "all" BASIS products hosting Slave control objects so that they reflect the intent of the Master control.

3.2

Global Controls: Examples of Master Control Messages

This section provides some examples of QSC protocol messages (formatted in XML) for managing Master control objects using third party control systems. Included are a number of XML examples directed at BASIS products hosting Slave control objects and the response messages from these devices, which are directed back at the control system.

In example M1, the intent is to change the level of a Master control object. Since we are only working with one Master control object in these examples, the object's enumeration is simply "1". The value "1" is therefore the ID of the Master control object. To change the level of a Master control object with an ID of 1, we use the OID 0x06032001 (refer to Table 2). In example M1 we are changing the level of the Master control object from some previous state to a new value of -3.2. Note that the unit of measure associated with the value is dependent on the control type and is not included in the protocol message. In example M1, we are changing the level of DSP Outputs (the Slave control objects) on a BASIS product so the appropriate unit is "dBu". It should be understood however that there is no facility through the BASIS Third Party Control Interface to determine what type of control (gain, frequency etc.) is being mastered. The programmer must first create the Master control object and assign Slave control objects to the Master control object using Venue Manager. The type of control to be mastered is defined when the Master control object and Slave control objects are created. This "known" information must then be applied to the third party application.

Example M1: SET Master Level

```
<?xml version="1.0"?>
<!--0000113-->
<SET_S RT="C00" ID="0" L1PW="qsc">
<ESS O="0x06032001" M="0" F="-3.2"/>
</SET_S>
```

Example M1 is a SET message that is directed at a BASIS product hosting one or more target Slave control objects. Note that each BASIS product hosting one or more target Slave control objects must be sent a separate message (all messages are unicast). The message is sourced from a controller that hosts the Master control. The actual Master control "object" is distributed amongst all Slave control objects in the sense that the Master control object's state, type and value are stored in the local memory of each BASIS that hosts a Slave control object.

As with all QSC protocol messages, the XML string in example M1 includes the password of the target BASIS, the appropriate XML string length, the message type etc. The message also includes the new value for the Master control object (**F="-3.2"**). Figures 2 and 3 show the effect the message in example M1 would have on a Slave control object that was previously set at -5.0 dBu if the Master control was previously set at +7.6 dBu.

Managing BASIS via 3rd Party Control Systems

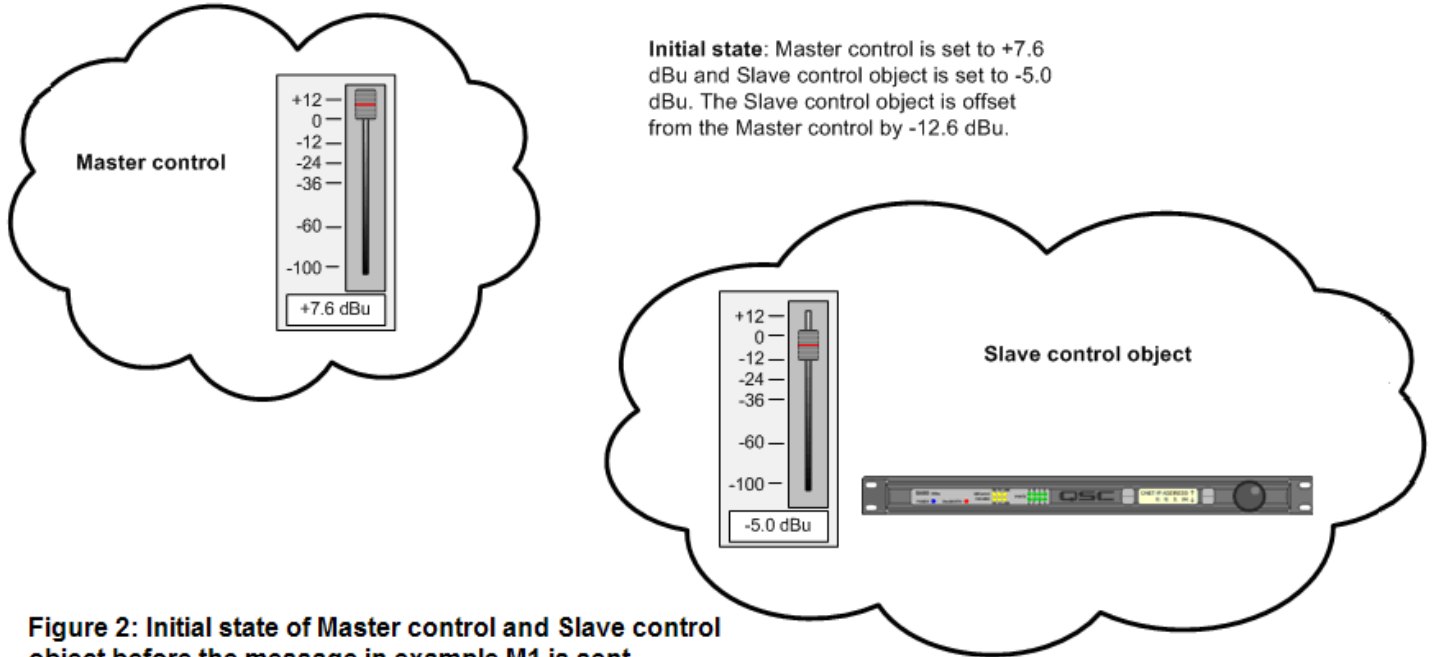


Figure 2: Initial state of Master control and Slave control object before the message in example M1 is sent

In the “Initial state” shown in Figure 2, the BASIS keeps track of the Slave control object’s value relative to the Master control. In this case, the Slave control object’s value deviates from the Master control by -12.6 dBu (+7.6 dBu - -5.0 dBu).

Managing BASIS via 3rd Party Control Systems

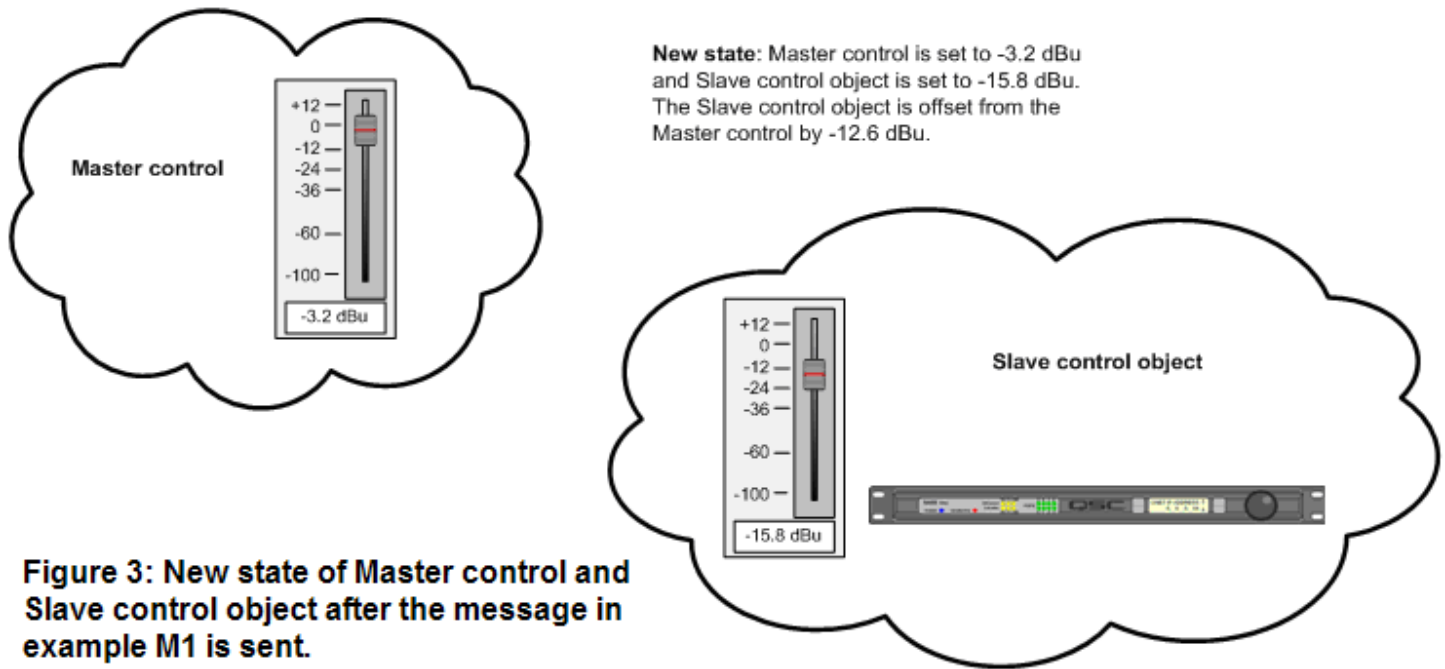


Figure 3: New state of Master control and Slave control object after the message in example M1 is sent.

In the “New state” shown in Figure 3, the BASIS manipulates the Slave control object’s value to maintain a value that is -12.6 dBu offset from the Master control. Since the Master control is now -3.2 dBu, the Slave control object’s value is changed to -15.8 dBu ($-15.8 \text{ dBu} - -3.2 \text{ dBu} = -12.6 \text{ dBu}$).

Example M2: SET Master Level (response message)

```
<?xml version="1.0"?>
<!--0000119-->
<SETACK_S RT="C00" ID="0">
<ESS O="0x06032001" R="0" M="0" F="-3.20000"/>
</SETACK_S>
```

Example M2 is simply the response to the SET message in example M1. The response is sourced from a BASIS product hosting a Slave control object that was manipulated. The response message is directed at the device hosting or managing the Master control. The response message in example M2 indicates that the Slave control object received the SET Master level message and that the Slave control executed the directive successfully. The new Master value is also reported in the response message for confirmation.

Example M3: Register the Master Level object

```
<?xml version="1.0"?>
<!--0000124-->
<SET_REG RT="C00" ID="0" L1PW="qsc">
<REG_RATE="50"/>
<REG O="0x06032001" S="1"/>
</SET_REG>
```

Managing BASIS via 3rd Party Control Systems

Example M3 is a SET registration message for the Master control object. The SET registration directive can be sourced to any BASIS product that has an active Slave control associated with the Master control object. Registration is used primarily in systems with multiple controllers. A secondary controller can register for Master control objects hosted and/or managed by remote controllers so that the secondary controller can be kept up to date when value changes occur to the Master control.

Example M4: Sync Slave to Master

```
<?xml version="1.0"?>
<!--0000113-->
<SET_S RT="C00" ID="0" L1PW="qsc">
<ESS O="0x06036001" M="0" F="-8.4"/>
</SET_S>
```

Example M4 is a SET message that changes the value of a Slave control object so that it matches that of the Master control value. In the example, the new value of the Master control object is set to -8.4 dBu. After the message is sent to the BASIS product hosting the Slave control object, the Slave control object's value will also be set to -8.4 dBu. Figures 4 and 5 illustrate this concept.

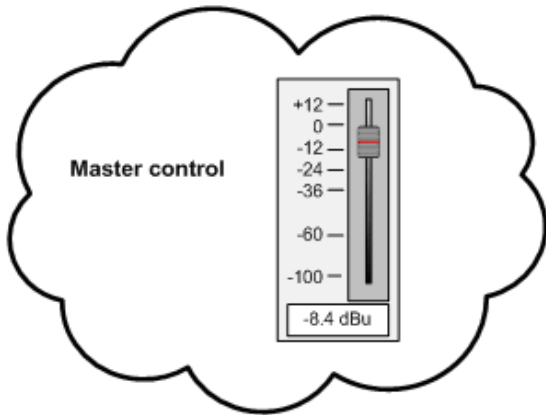
In the "Initial state" shown in Figure 4, the Master control value is set at -8.4 dBu and the Slave control object's value is set at -19.3 dBu. The offset between the Master control value and the value of the BASIS Slave control object is -10.9 dBu. The offset is provided as a reference, though it is of little concern when synchronizing Slave controls to a Master control value since the offset will be set to zero after the directive is executed.

In the "New state" shown in Figure 5, the Master control value *remains* at -8.4 dBu. This is important! Slaves cannot sync to the Master control value if the Master control value is altered in the SET message. If there are multiple controllers in the system that have the ability to manipulate the Master control value, the controller issuing the SET message should first poll or register the Master control object to determine its current value. After the SET message is sent to the BASIS product hosting the Slave control object, the Slave control object's value tracks the Master control value exactly and is changed to -8.4 dBu. Example M5 shows the response message returned from a BASIS product hosting a Slave control object that was manipulated according to the message sourced in example M4. The response message in Example M5 indicates that the Slave control received the SET message and was able to successfully execute the directive. The message also indicates the new value of the "Master" control as it is stored in the BASIS memory.

Example M5: Slave Sync to Master (response message)

```
<?xml version="1.0"?>
<!--0000119-->
<SETACK_S RT="C00" ID="0">
<ESS O="0x06036001" M="0" R="0" F="-8.400000"/>
</SETACK_S>
```

Managing BASIS via 3rd Party Control Systems



Initial state: Master control is set to -8.4 dBu and Slave control object is set to -19.3 dBu. The Slave control object is offset from the Master control by -10.9 dBu.

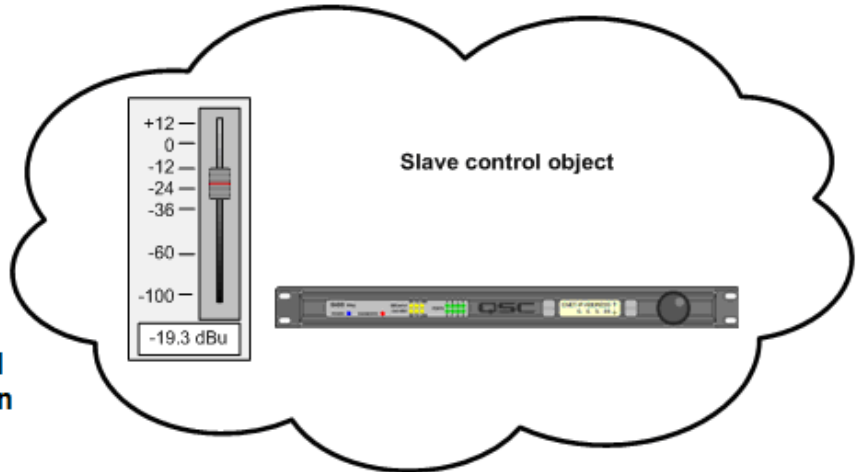


Figure 4: Initial state of Master control and Slave control object before the message in example M4 is sent



New state: Master control is set to -8.4 dBu and Slave control object is set to -8.4 dBu. The value of the Slave control object exactly matches that of the Master control value.

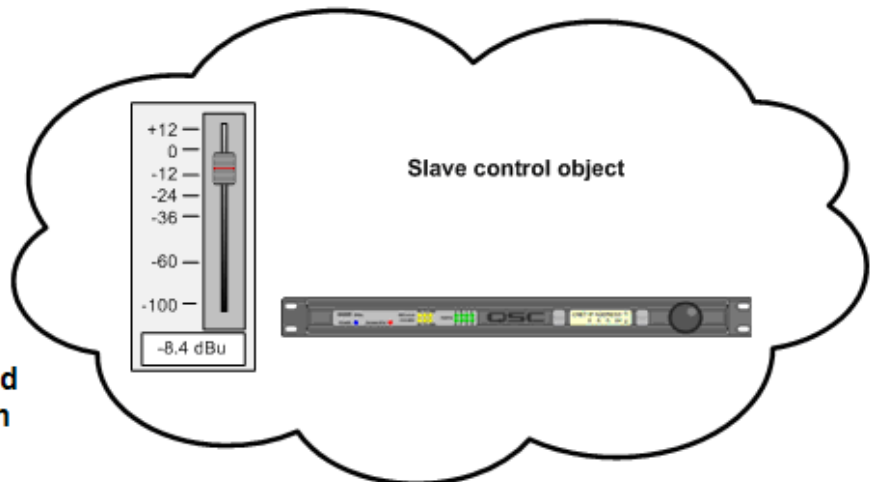


Figure 5: New state of Master control and Slave control object after the message in example M4 is sent.

Managing BASIS via 3rd Party Control Systems

The last two Master control examples pertain to the name assigned to a Master control object. A name (ASCII string) is assigned to the Master control object when it is created in Venue Manager. A third party controller can GET this name by issuing a request to any BASIS unit hosting an active Slave control object associated with the Master control. This is useful in systems with multiple controllers or in systems that share management responsibilities with applications in the QSCControl.net suite of applications. In such systems, a secondary controller can inquire as to the name of a Master control object hosted on another controller in the system. Example M6 shows the GET request message used to obtain the name. The OID used in the request message must include the ID of the Master control object. However, it is possible to issue a series of requests using an incrementing Master control object ID scheme in the OID in order to “discover” Master control objects and their associated names.

Example M6: GET name of Master control object

```
<?xml version="1.0"?>
<!--0000104-->
<GET_S RT="C00" ID="0" L1PW="qsc">
<EGS O="0x06030001" M="0"/>
</GET_S>
```

Example M7 shows the response message returned from a BASIS unit hosting an active Slave control object. The response is directed back to the inquiring controller (the requester). The example in M7 shows that the GET message was received and that the Slave control was able to execute the directive successfully. Of course the name is also returned in the response message. In this case, the name of the Master control object with ID = 1 is “**Master Cylinder**”.

Example M7: GET name of Master control object (response message)

```
<?xml version="1.0"?>
<!--0000126-->
<GETACK_S RT="C00" ID="0">
<EGS O="0x06030001" R="0" M="0" ST="Master Cylinder"/>
</GETACK_S>
```

3.3 Global Controls: OIDs for Global Presets

Table 3 provides a list of supported OIDs (object identifiers) related to Global Preset control objects. As can be seen from the Table, there is only one function available: recall Global Preset. A Global Preset can be recalled using either its name or its enumeration as the reference.

OIDs used for Global Preset Recall	Function
0x06 01 00 00	Recall a Global Preset by referencing the Global Preset Name
0x06 02 00 00	Recall a Global Preset by referencing the Global Preset Number

Table 3: OIDs associated with Global Presets

From the perspective of a third party control system, recalling Global Presets is just a simple directive. However, the directive must be issued to each and every BASIS product that is a member of the Global Preset. While Global Presets can get quite complex, they are defined and configured using Venue Manager. As such, there is little to be concerned with when managing Global Presets through the BASIS Third Party Control Interface. The third party programmer need only know the Global Present name or the Global Preset number (enumerated sequentially at the time of creation) to issue the recall directive.

3.4 Global Controls: Examples of Global Preset Messages

Examples GP1 and GP2 demonstrate the Global Preset recall directive. These two SET messages direct a BASIS device to recall a given Global Preset. The SET directives reference either the Global Preset name (ASCII string) or the Global Preset number.

Example GP1: SET Global Preset by Number (recall)

```
<?xml version="1.0"?>
<!--0000110-->
<SET_S RT="C00" ID="0" L1PW="qsc">
<ESS O="0x06020000" M="0" S="2"/>
</SET_S>
```

Example GP1 shows the Global Preset recall message (SET message) using the Global Preset's number. In this example, Global Preset 2 is recalled. The Global Preset number is included in the S="2" field, which indicates the Global Preset number as the identifier.

Example GP2: SET Global Preset by Name (recall)

```
<?xml version="1.0"?>
<!--0000118-->
<SET_S RT="C00" ID="0" L1PW="qsc">
<ESS O="0x06010000" M="0" ST="Hercules"/>
</SET_S>
```

Example GP2 shows the Global Preset recall message (SET message) using the Global Preset's name. In this example, Global Preset **Hercules** is recalled. The Global Preset name is included in the **ST="Hercules"** field, which indicates the Global Preset name as the identifier.

4.0

Summary:

QSC provides the BASIS Third Party Control Interface with a mechanism for “managing” a limited number of global, or venue-wide, controls from a third party control system. This allows a third party control system to use a single “Master” control to affect a change in all BASIS products in a Venue or to affect a change in a subset of these products. Using the BASIS Third Party Control Interface, a third party control system can also use a single Global Preset control to recall Config and Snapshot combinations in all BASIS products in a Venue or to recall a Config and Snapshot combination in a subset of these products. Since the Global Preset object supports linking of Master control objects to Global Preset objects, the system designer is able to set specific levels (or other control values) throughout the system when a Global Preset is recalled.

A third party programmer must keep in mind however that the BASIS Third Party Control Interface does not provide a means for creating or configuring these global controls. QSC's Venue Manager application must be used to create these controls and to configure them. Once configured, a third party control system can then manage the controls and the BASIS platform products that these control objects reside in.

The power of global control objects is that they remove the burden of data management from the control system. The control system no longer has to store fixed, relational, and historical values in order to keep track of Master and Slave control objects and/or to manage “scenes” that include combinations of Configs, Snapshots and Master control objects that need to be recalled as a set. In many applications, the use of global controls can also reduce the quantity of messages processed by the third party controller and/or the quantity of messages sent over the network since several actions can be grouped into a single directive using a single protocol message (single XML string).

5.0

Glossary:

Global Presets

- (a) A generic reference to controls that affect a venue-wide Config and Snapshot recall (and optionally a Master control setting recall).
- (b) The combination of Config, Snapshot and (optionally) Master control object(s) associated with a specific recall set.

Global Preset Control

A control element, that when exercised, results in the recall of a Config and Snapshot and (optionally) a Master control setting combination. Global Preset controls reside in the control system not in BASIS hardware.

Global Preset Control Objects

Virtual components that link or associate one or more Configs, Snapshots and (optionally) Master control settings combinations to a Global Preset control. Global Preset control objects reside in BASIS firmware and provide subordinate controls with information about the Global Preset. This information is currently limited to the name and enumeration associated with a Global Preset. Global Preset control objects are generally instantiated through QSCControl.net's Venue Manager application.

Masters

A generic reference to controls that affect a venue-wide change. Typical examples are master level controls. "Master" refers to the control portion of the system and "Slave" refers to the subordinate portion of the system... or the portion that is affected by changes to the Master.

Master Controls

Control elements, that when manipulated, result in the delivery of one or more commands towards a BASIS product that hosts one or more Slave control objects. Master controls reside in the control system not in the BASIS hardware. Examples include Master gain faders on a touch panel or a Master volume pot on a wall controller.

Master Control Objects

Virtual components that link or associate one or more Slave control objects to a Master control. Master control objects reside in BASIS firmware and provide the Slave control objects with information about the Master Control. For example, Master control object's provide Slave control objects with the ID of the Master control, the current value of the Master Control, and (optionally) with the name of the Master control. Master control objects are generally instantiated through QSCControl.net's Venue Manager application.

Managing BASIS via 3rd Party Control Systems

Slaves

A generic reference to controls and/or objects that are subordinate to a venue-wide Master control.

Slave Controls

Virtual components in BASIS firmware that execute the Master control's directive to affect a change to Slave control objects. For example, if a Master control issues a directive to change the value of a group of output levels, this directive would be sent to Slave controls that in turn execute the directive to affect a change to specific Slave control objects (which would be linked to actual output levels in our example). Slave controls are represented by the OID structures shown in Table 1.

Slave Control Objects

The assignment of signal flow objects that are subordinate to a Master control. In a more detailed sense, Slave control objects are virtual components in the BASIS firmware that identify specific static or dynamic objects in the BASIS signal flow as being Slaves to a Master control. As such, changes to the Master control affect a change to all associated Slave control objects. Output levels and filter parameters are good examples of objects in the signal flow that can be assigned to Slave control objects. Slave control objects are represented by the OIDs assigned to a given static or dynamic signal flow object. Static OIDs are fixed and are documented in QSC's "BASIS Third Party Control Interface" document. Dynamic OIDs are assigned at the time of signal flow creation (during the design of the Config). Dynamic OID assignments are reported in the Config file (they can also be discovered using QSC's BasisTerm utility).

Appendix G - BASIS / RAVE CobraNet OIDs (Rev. 3.0.9)

Introduction

This document was created to help describe the construction of QSCControl.Net object identifiers (OIDs) that represent BASIS’ managed objects and how these OIDs relate to BASIS platform products (the platform includes BASIS-derived RAVE models).

32 Bit OID breakdown

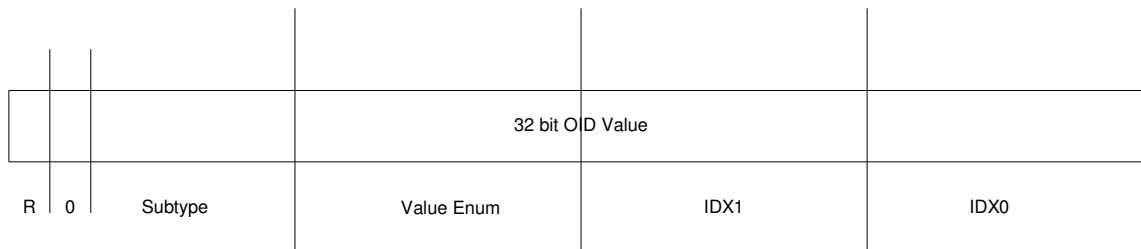
The OID is a signed 32 bit value that uniquely identifies an individual parameter stored in a BASIS or RAVE product. In order to preserve CLR compatibility, the sign bit (upper bit) is reserved and always set to 0. There are two major types of OID values. The first type of OID represents a static parameter type. When present, these parameters are fixed. A BASIS or RAVE product either always has these parameters or never has them, depending upon the model. The type of OID represents objects associated with the dynamic DSP signal flow. These parameters will come and go as dictated by the signal flow design.

CobraNet OID type

This document is concerned with CobraNet OIDs. All CobraNet OIDs represent static parameters whose values are obtained and/or manipulated when configuring, controlling or monitoring the CobraNet interface and its associated audio streams. CobraNet parameters are available on all BASIS models except the 722az and on all BASIS-derived RAVE models.

Static Value OID Construct

The Static Value OID Construct consists of four parts. The first part is the “Subtype” that the parameter belongs to. All CobraNet parameters belong to Subtype “04”. The second part is the “Value Enum”, which is an arbitrary value that represents a specific type of value in the product, such as the Boot Version or Conductor Priority value. The third and fourth parts of the OID are used for tabular or list indexing into the value. For example, there will be a Tx Bundle Assignment for each of the four available CobraNet transmitters. In this case, the index would identify the exact transmitter that a particular Tx Bundle Assignment value is associated with.



Sample OID values for CobraNet objects

Parameter: Hardware restart
OID: 0x04 01 00 00
Description: Restarts CobraNet via processor hardware reset

Parameter: Tx Bundle Assignment
OID: 0x04 1D 00 02
Description: Assigns bundle value to third transmitter (zero based indexing)

Parameter: Rx Bundle Dropout count
OID: 0x04 28 00 01
Description: Reports the quantity of isochronous cycles that were not processed for a given receiver (due to non-arrivals, dropped packets, late packets etc.)

Parameter: DSP Input Bundle Assignment
OID: 0x04 30 00 09
Description: Assigns one of four CobraNet receivers to the tenth DSP input channel*

Parameter: DSP Input Channel Assignment
OID: 0x04 31 00 09
Description: Assigns one of eight bundle channels to the tenth DSP input channel*

* Some OIDs, such as DSP Input Bundle Assignment and DSP Input Channel Assignment, are used collaboratively. In the example above, the two OIDs are used to “unbundle” a CobraNet delivery (a virtual bundle of audio channels), extract the contents of the bundle (the discrete digital audio channels), and route one of these audio channels to one of 24 inputs on a BASIS or RAVE DSP engine.

Note also that some BASIS and BASIS-derived RAVE models have physical analog or digital audio inputs on their chassis rear panels. Such models will reserve a block of 8 or a block of 16 DSP inputs exclusively for these physical audio inputs. As such, the OID indices available for routing CobraNet channels may be limited to 8 or 16 (rather than all 24 channels supported by the architecture). For example, CobraNet channels may only be routed to DSP input channels 8 through 23 on a RAVE 522aa because the 522aa has 8 analog inputs that reserve the first 8 inputs of the DSP engine (channels 0 through 7).

CobraNet control and monitoring

Routing type: BASIS

Subgroup 0x04 Note: all CobraNet controls are non-snapshotable

Object Name	Value Enum	Devices	Index
Reserved	0x00	ALL	0
Hardware Restart	0x01	BASIS / RAVE	1
Firmware Restart	0x02	BASIS / RAVE	1
System Description	0x03	BASIS / RAVE	1
Firmware Version	0x04	BASIS / RAVE	1
Firmware Manuf. ID	0x05	BASIS / RAVE	1
Boot Version	0x06	BASIS / RAVE	1
SNMP Mod. Permission	0x07	BASIS / RAVE	1
CurrentIPAddress	0x08	BASIS / RAVE	1
StaticIPAddress	0x09	BASIS / RAVE	1
System Up Time	0x0A	BASIS / RAVE	1
Flash Persistence	0x0B	BASIS / RAVE	1
Store Variables	0x0C	BASIS / RAVE	1
Flash Size	0x0D	BASIS / RAVE	1
Flash Sector Size	0x0E	BASIS / RAVE	1
Last Error	0x0F	BASIS / RAVE	1
Conductor Priority	0x10	BASIS / RAVE	1
Conductor Cycle Rate	0x11	BASIS / RAVE	1
Conductor Status	0x12	BASIS / RAVE	1
Sync. Cond. Clock Mode	0x13	BASIS / RAVE	1 (not supported)
Total Tx Channels	0x14	BASIS / RAVE	1
Total Rx Channels	0x15	BASIS / RAVE	1
Total Allowed Channels	0x16	BASIS / RAVE	1
Local Loopback Input	0x17	BASIS / RAVE	0
Local Loopback Output	0x18	BASIS / RAVE	0
Enable/Disable Tx Bundle	0x19	BASIS / RAVE	0
Tx Bundle Dropout Count	0x1A	BASIS / RAVE	4
Tx Bundle Position	0x1B	BASIS / RAVE	4
Tx Bundle Receivers	0x1C	BASIS / RAVE	4
Tx Bundle Assignment	0x1D	BASIS / RAVE	4
Tx Bundle Trans. Priority	0x1E	BASIS / RAVE	4
Tx Bundle Assign. Pr.	0x1F	BASIS / RAVE	4
Tx Bundle Channel Count	0x20	BASIS / RAVE	4
Tx Bundle Buddy Exclude	0x21	BASIS / RAVE	4 (not supported)
Tx Bundle Unicast Mode	0x22	BASIS / RAVE	4
Tx Bundle Max. Unicast Receivers	0x23	BASIS / RAVE	4
Tx Channel Assignment	0x24	BASIS / RAVE	4 x 8
Tx Channel Format	0x25	BASIS / RAVE	4 x 8
Enable/Disable Rx Bundle	0x26	BASIS / RAVE	0

Managing BASIS via 3rd Party Control Systems

Rx Bundle Reception Status	0x27	BASIS / RAVE	4
Rx Bundle Dropout count	0x28	BASIS / RAVE	4
Rx Bundle Delay	0x29	BASIS / RAVE	4
Rx Bundle Min. Delay	0x2A	BASIS / RAVE	4
Rx Bundle Assignment	0x2B	BASIS / RAVE	4
Rx Bundle Prv. Src. MAC	0x2C	BASIS / RAVE	4
Rx Bundle Priority	0x2D	BASIS / RAVE	4
Rx Bundle Buddy Exclude	0x2E	BASIS / RAVE	4 (not support)
Rx Channel Receive Resolution	0x2F	BASIS / RAVE	4 x 8
DSP Input Bundle Assignment	0x30	BASIS / RAVE	24 Basis 922 & 914, RAVE 520 & 522: indices 8-23 Basis 902 & 904: Indices 0 - 23
DSP Input Channel Assignment	0x31	BASIS / RAVE	24 Basis 922 & 914, RAVE 520 & 522: indices 8-23 Basis 902 & 904: Indices 0 - 23
DSP Input Format	0x32	BASIS / RAVE	24 Basis 922 & 914, RAVE 520 & 522: indices 8-23 Basis 902 & 904: Indices 0 - 23
Latency Control(Config) (NS)	0x34	BASIS / RAVE	1
Latency Status	0x35	BASIS / RAVE	1
Rx Channel Receive Latency	0x36	BASIS / RAVE	4 x 8
Rx Channel Receive Sample Rate	0x37	BASIS / RAVE	4 x 8
Rx Channel Receive Valid Status	0x38	BASIS / RAVE	4 x 8
Tx Bundle Transmit Status	0x39	BASIS / RAVE	4

Revision Log (CobraNet-only information)

Version: based on 3.0.9

Date: 8/24/07

Description:

1. Abridged original document for external customers requiring CobraNet OIDs