

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 Global Control Objects (via the BASIS Third-Party Control Interface)

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

Software release: 3.00.569
Date: 11/06/2007

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 QSCControl.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 QSCControl.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 QSCControl.net suite of applications with the public release of QSCControl.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 QSCControl.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 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 QSCControl 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 for Venue Manager or any other system design and/or system configuration tool in the QSControl.net suite of applications. It is therefore expected that the system designer will use the QSControl.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 QSControl.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 QSControl.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).

Figure 1 illustrates the basic global control model. The Figure shows a global control hosted on a third party control system. Subordinate control

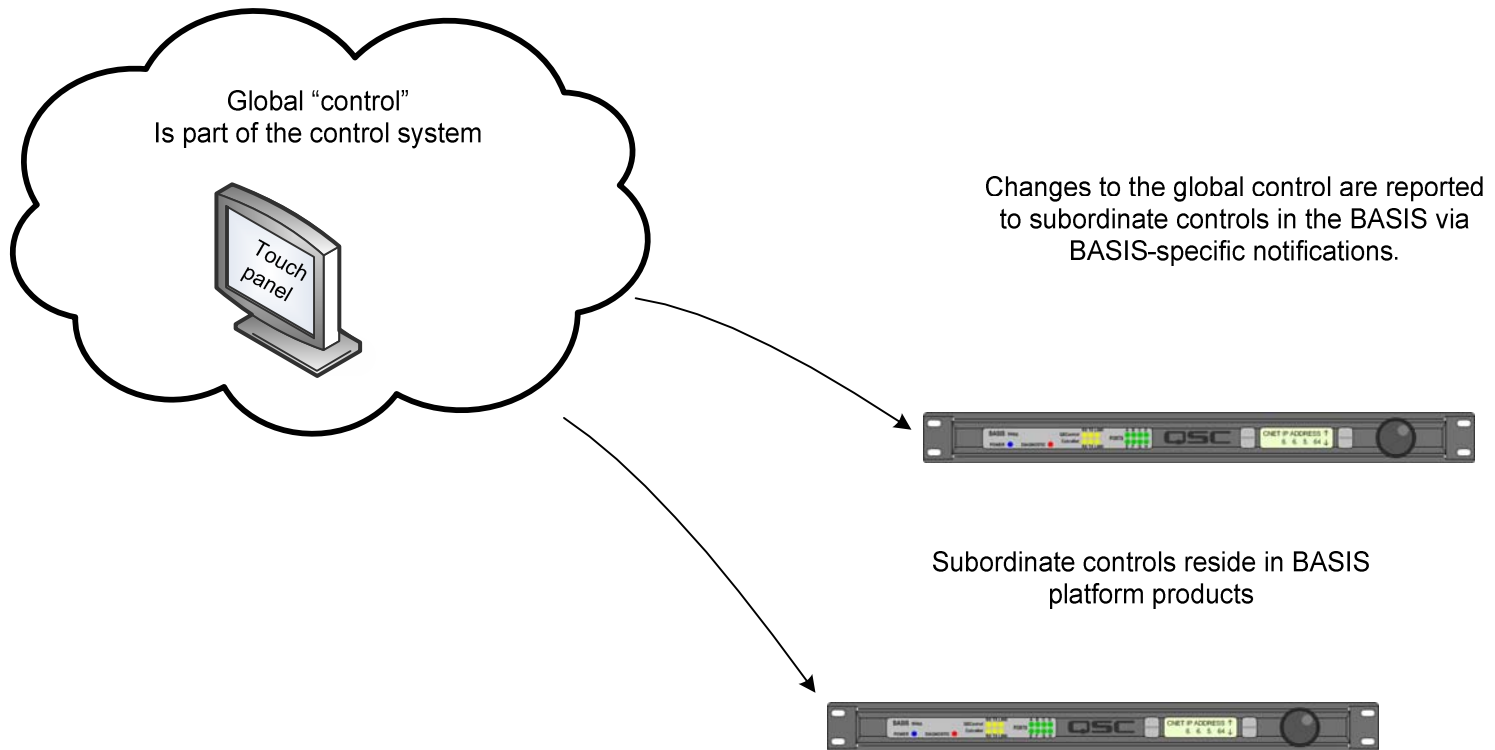


Figure 1: Global control model

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 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 structures into three components (OID prefix, IDX1 and IDX0), which form the basis for the OIDs applicable to Master controls. Complete OIDs are created when the OID prefix field and the IDX1 and IDX0 fields are concatenated.

Parameter Description	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 I.D. = 6	000	0 0000 0000 0110	0x06 03 00 06
OID for parameter 1 and Master I.D. = 3	001	0 0000 0000 0011	0x06 03 20 03
OID for parameter 3 and Master I.D. = 29	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 QSControl.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 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 QSControl.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.

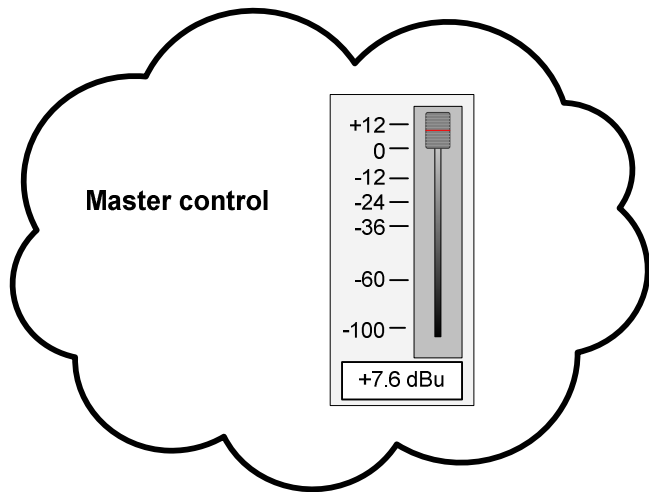
In example M1, the intent is to change the level of a Master control object. Since we are only working with one Master control object in these examples, the object's enumeration is simply "1". The value "1" is therefore the ID of the Master control object. To change the level of a Master control object 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.



Initial state: Master control is set to +7.6 dBu and Slave control object is set to -5.0 dBu. The Slave control object is offset from the Master control by -12.6 dBu.

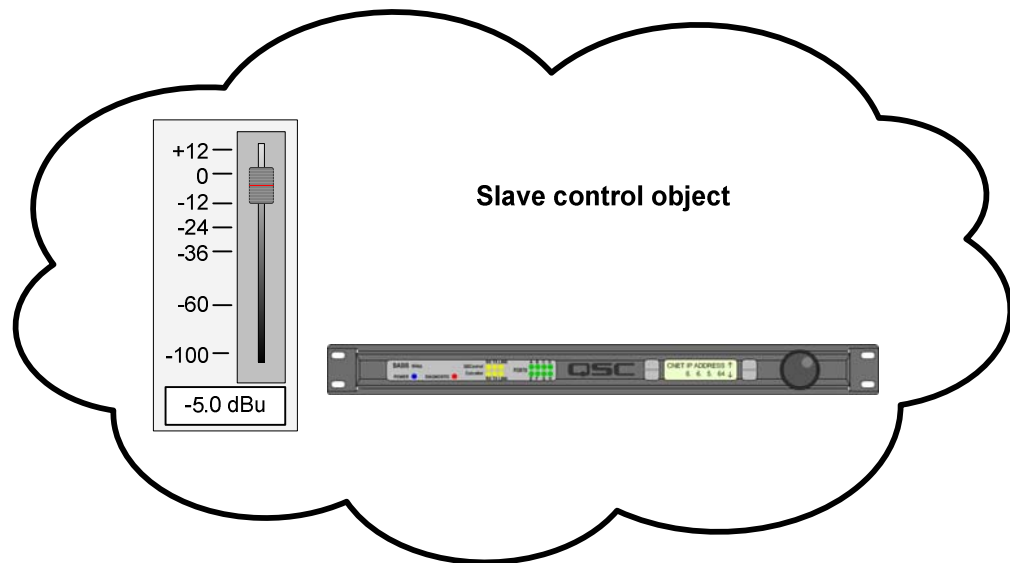
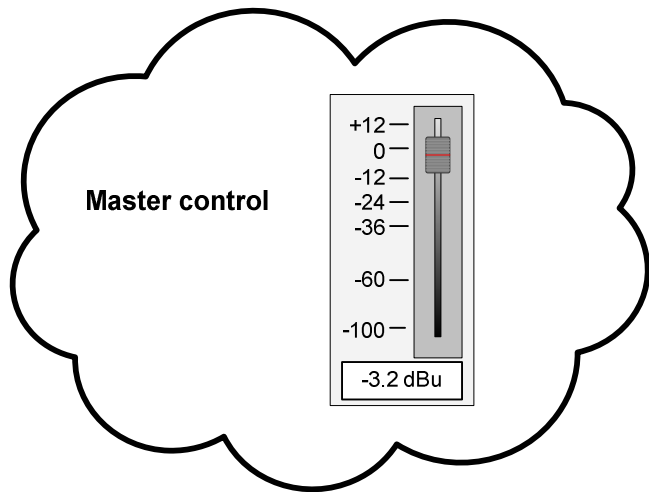


Figure 2: Initial state of Master control and Slave control object before the message in example M1 is sent

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).



New state: Master control is set to -3.2 dBu and Slave control object is set to -15.8 dBu. The Slave control object is offset from the Master control by -12.6 dBu.

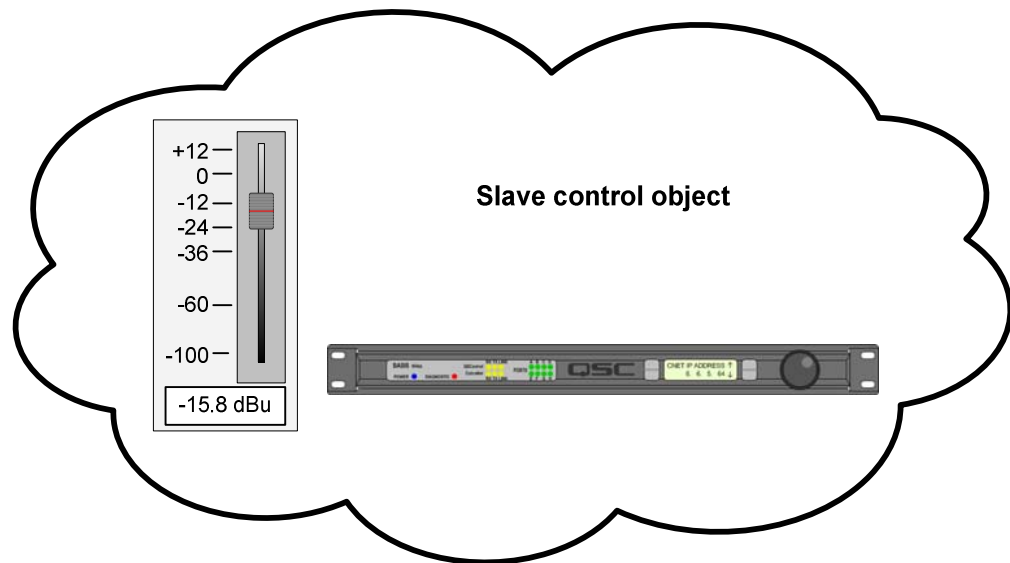


Figure 3: New state of Master control and Slave control object after the message in example M1 is sent

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="-3.200000"/>
</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>
```

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>
```

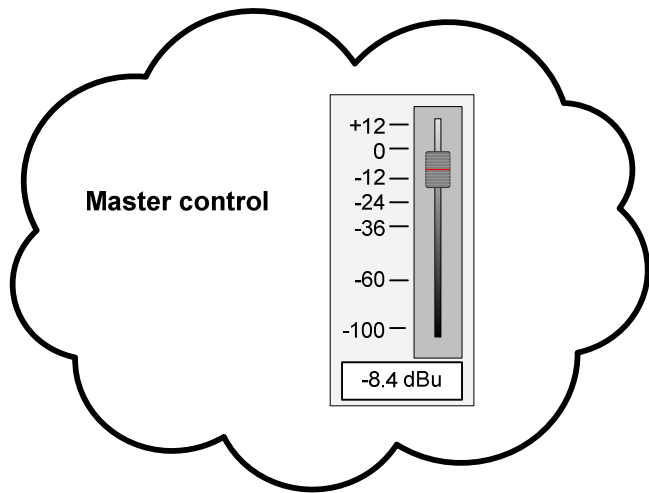
Example M4 is a SET message that changes the value of a Slave control object so that it matches that of the Master control value. In the example, the new value of the Master control object is set to -8.4 dBu. After the message is sent to the BASIS product hosting the Slave control object, the Slave control 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: Sync Slave 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>
```



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.

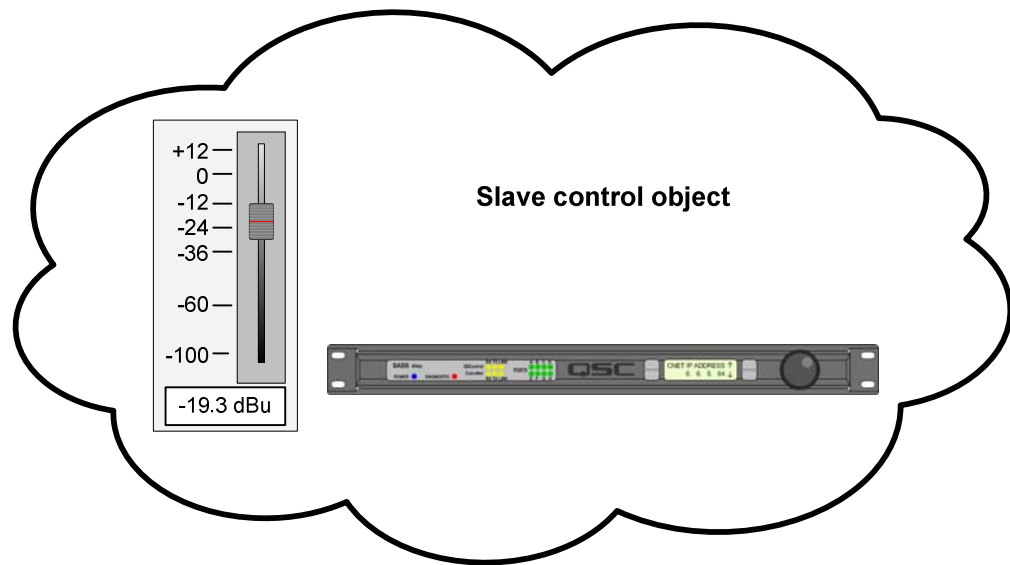
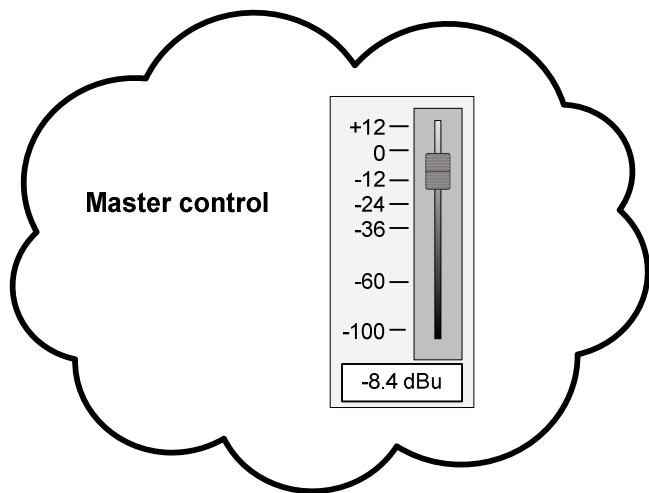


Figure 4: Initial state of Master control and Slave control object before the message in example M4 is sent



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.

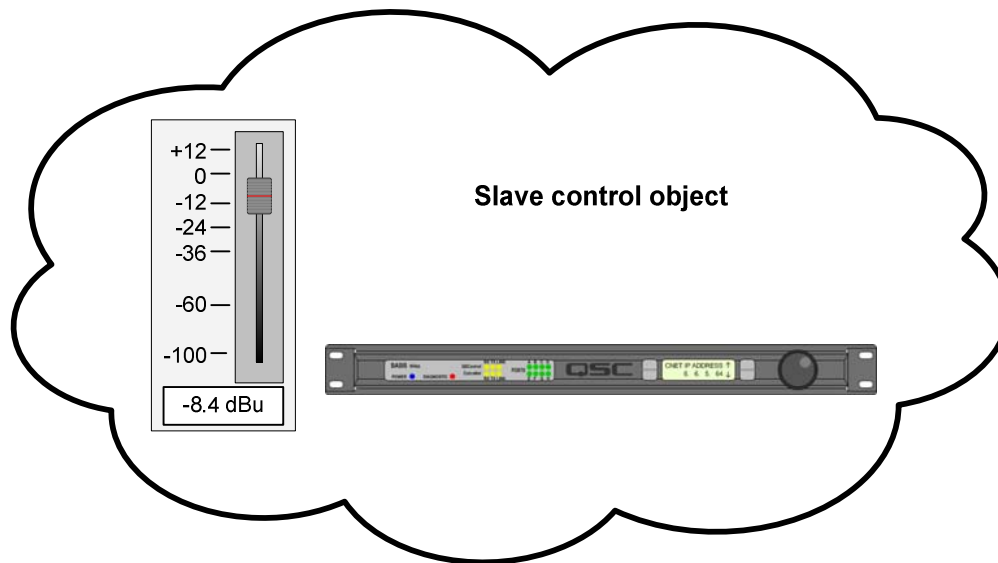


Figure 5: New state of Master control and Slave control object after the message in example M4 is sent

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 objects 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.

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).



BASIS, RAVE, QSCControl, QSCControl.net, QSCreator, QSCAD and Venue Manager are registered trademarks of QSC Audio Products, Inc. "QSC" and the QSC logo are registered with the U.S. Patent and Trademark Office

QSC Audio Products, Inc. • 1675 MacArthur Boulevard • Costa Mesa, California 92626 USA • PH: (714) 754-6175 • FAX: (714) 754-6174

rev. 2

© Copyright 2007 QSC Audio Products, Inc. All rights reserved