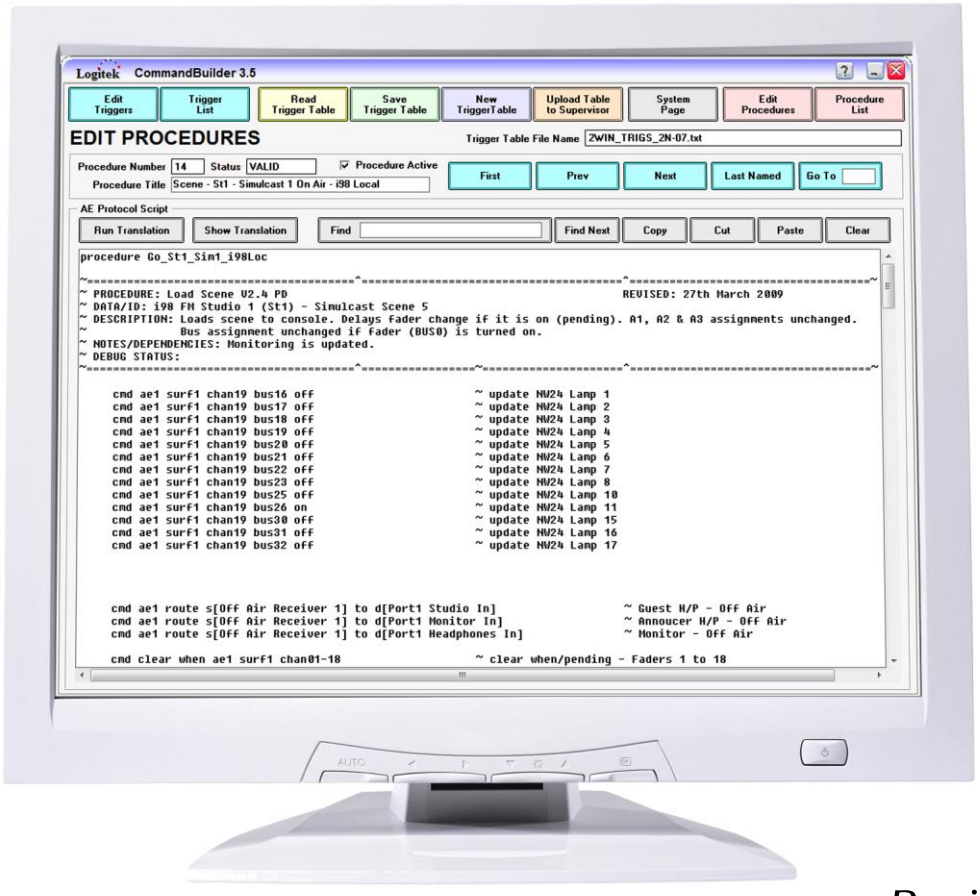
Logitek

Logitek Electronic Systems

CommandBuilder 5
Reference Manual



Revision 5
Aug 2015

Logitek Electronic Systems, Inc.
5622 Edgemoor Drive
Houston, Texas 77081
USA

Tel +1-713-664-4470
Fax +1-713-664-4479

Email support@logitekaudio.com
Web www.logitekaudio.com

Notice
Every effort has been made to supply complete and accurate information. However, Logitek Electronic Systems, Inc. assumes no responsibility for its use, nor any infringement of patents or other rights of third parties, which would result.

All specifications are subject to change without notice.

Document Revisions

Date	Revision	Author	Notes
May 2009	3.5	Ben Hietbrink	First release of CommandBuilder 3.5 manual
August 2009	3.6	Ben Hietbrink	First release of CommandBuilder 3.6 manual
August 2015	5	Tag Borland/John Davis	Revised to include version 5 commands

Contents

Command Index

Part A: Using CommandBuilder

This section provides an overview of the *CommandBuilder* program, and how to use it to design, create, load, save and upload **Logitek Trigger Tables**.

1 Introduction

About this Manual

This manual describes the installation and operation of the **Logitek** *CommandBuilder* application.

Intended Audience

This manual is aimed at Engineers and Technical Operators responsible for installing, configuring and supporting a **Logitek Console Router System** utilizing the *Supervisor* and *vTools* applications.

CommandBuilder is used to design macros, or "triggers" that perform advanced functionality. The **Logitek Scripting Language** is a simple programming language that provides a level of control over your **Logitek** system that is limited only by your imagination. It is assumed that the person responsible for installing and configuring *Supervisor* has a solid understanding of Microsoft Windows desktop operating systems, or has ready access to IT support. Users wishing to develop complex **Trigger** functionality will benefit from any programming experience.

Manual Conventions

The following conventions are used in this manual:

This text indicates a menu choice to be made, with an arrow separating a multi-level selection, e.g. Control Panel ➢ Users & Passwords. This can be a menu choice in a Logitek application, or within Windows.

➥ *Indicates a "see-also" section in this manual, or another Logitek manual.*

The exclamation symbol signifies an important note or critical information.

```
This text represents a command, script block example, instruction to
        be typed, or directory path.
```

TIP: A useful tip from our knowledge base!

This page is intentionally left blank.

About CommandBuilder

The *CommandBuilder* program is a special text editor developed by **Logitek** to add advanced functionality to your **Logitek** system. It allows you to "program" the various buttons and GPIs on the **JetStream** and **Surfaces**, and write "triggers" that react to almost anything that occurs.

In *CommandBuilder* the user writes **JetStream** action commands that are executed when a trigger occurs in the **JetStream**. **Triggers** are basically events, for example a button is pressed, a GPI is activated, or a fader source is changed. **Triggers** are built in *CommandBuilder* and stored in text format. The **Trigger Table** is compiled into proprietary format and uploaded to *Supervisor*.

The true power of **Triggers** is unleashed in multi-engine systems. *Supervisor* can control interactive events among multiple **JetStream**, such as an intercom between studios, or on-air switching.

How to Use This Manual

The primary purpose of this manual is to introduce the reader to the **Logitek Scripting Language** and to provide a useful starting point for developing your own **Triggers**.

The manual is divided into three main parts. These can be read in isolation, to suit the level of knowledge the reader requires.

Part A – Using CommandBuilder

The first section of the manual provides an introduction to the application and how to navigate your way around it. *CommandBuilder* operates like most **Logitek** applications, so these topics are quick summaries.

Part B – Logitek Scripting Language

The majority of the manual is devoted to a **Logitek Scripting Language** reference, providing detailed explanations of all keywords and functions. It is not necessary to read these chapters completely, but we suggest you skim read the topic summaries and read the detail for any areas of interest.

Part C – Trigger Applications

The last section provides real world **Trigger** examples to help you achieve common functions quickly. These include intercoms, on-air switching and console scenes. These examples are based on the best **Triggers** users, integrators, resellers and **Logitek** itself has designed and implemented worldwide.

Which Parts to Read

This manual can be read from cover to cover, but not all users will need to do so.

System Administrators

If you are a System Administrator, responsible only for uploading new **Trigger** tables (supplied by other engineers at your station, Logitek, or resellers), then you only need to read the first few chapters of Part A, stopping at the end of Chapter 4.

Maintenance Engineers

If you need to maintain or modify existing **Trigger Tables**, we suggest you read all of Part A, to learn your way around the entire application.

You can then use Part B as a reference, looking up only the commands and **Trigger** types used by your station. Part C may be of interest to get ideas for future changes or additions.

System Engineers

If you are designing and configuring a **Logitek System**, you'll probably want to read this manual from cover to cover. We suggest you become familiar with the application and commands first, and then use the examples in Part C to help you develop and test **Triggers** to suit your needs.

System Integrators / Value Added Resellers

System Integrators and VARs should also read this manual from cover to cover.

In particular, we encourage you to thoroughly evaluate the example Triggers in Part C, as these provide comprehensive examples of how to perform complex tasks.

System Requirements

CommandBuilder is installed on every JetStream on its embedded computer. It runs in a Windows Embedded environment.

Compatibility

This manual is for *CommandBuilder 5*, designed for use with *JetStream Server v5*.

CommandBuilder 5 is a release that supports multiple JetStreams. One trigger table is used for all JetStreams. If you are familiar with Command Builder 3.6 and Supervisor as used on the AE-32 Audio Engine you will find that many of the commands and processes are the same. However, the toplogy of the system is very different and you should read chapter 2, Trigger Concepts, very carefully to understand how the new system works.

CommandBuilder and *JetStream Server* are inter-related when it comes to **Trigger** compilation. The version of both applications should be the same, and the build dates should also match. This is because the compiled **Trigger Table** file format must match between both applications.

2 Trigger Concepts

You can use *JetStream Server* without making use of any of the **Trigger** functionality of *CommandBuilder*. However, most recent **Logitek** installations now make use of **Triggers** in some way. If you have any button panels on your surfaces, you will almost certainly need *CommandBuilder* to program those buttons. *Pilot*, ROC, *Artisan* and *Mosaic* surfaces all include programmable buttons in their basic configurations. Additional panels may be added if more buttons are needed.

CommandBuilder – do I need it?

If you want to make use of the advanced macro programming functionality offered by *JetStream Server*, you will need *CommandBuilder*. There are two primary uses for the application:

Developing Triggers and Procedures

Depending on your situation, you may be developing triggers and procedures, or using a third party such as **Logitek**, a reseller or a systems integrator.

CommandBuilder is used to develop **Triggers** and **Procedures** into a **Trigger Table** that is later uploaded to *JetStream Server*.

Uploading Existing Triggers and Procedures

Regardless of the origin of your **Trigger Table**, it needs to be uploaded to *JetStream Server* to be executed. If your station uses a third party to provide **Trigger Tables**, you'll only need to read and upload the **Triggers** and **Procedures** with *CommandBuilder*.

What are Triggers and Procedures?

A Trigger is an event "hook" that tells *JetStream Server* what to do when something inside an **Audio Engine** happens. Triggers contain two primary parts:

1. The first line, which tells *Supervisor* what to "trigger" on (e.g. a particular button press)
2. A number of lines of code to execute when that **Trigger** event occurs.

For example, if you have a Numix Button24 Wedge (a panel with 24 buttons/lamps), you have a trigger for each button on that panel that you assign a function to. Often, you might have Triggers assigned only to the "ON" function, i.e. to run a function when the button is pressed. In some cases, you will also have a corresponding "OFF" trigger, that runs when the button is released (e.g. a push-to-talk type function).

Procedures are reusable code blocks that can be called by **Triggers**, or other **Procedures**. These are useful for reducing code repetition for frequently used functionality.

➥ *The exact format of Trigger and Procedure commands is covered in Part B of this manual.*

Where are Triggers stored?

Like standard software development, there are two parts to **Trigger Tables** – the original source code used to develop them, and the compiled code to be executed.

The process below outlines the various stages of **Trigger** development and execution. The **Trigger** source code is stored in text format by *CommandBuilder*. The actual source files are editable in a text editor, provided the line numbers and trigger numbers are not upset.

When you upload a **Trigger Table** to *JetStream Server*, *CommandBuilder* compiles the table into commands that *JetStream Server* understands, and uploads the compiled table to JetStream Server. *Supervisor* then stores the compiled file in memory and on disk. The disk file is only replaced when you upload a new copy of the **Trigger Table**.

Where are Triggers executed?

The trigger execution process is generally as follows:

1. An event is originated by a control surface (this step is skipped if the event originates in the engine).
2. The Audio Engine receives this event, and sends a copy of the command to *JetStream Server*.
3. *JetStream Server* logs the event and updates its JetStream State and Log pages.
4. *JetStream Server* checks the event to see if it has a matching trigger for that command.
5. If a trigger exists, *JetStream Server* processes the logic and commands of the trigger.
6. If the trigger contains commands for that JetStream, *JetStream Server* sends these to the DSP card.
7. The DSP card executes the commands, or for lamp/text displays, sends it to the surface.
8. The Surface executes the commands.

System Topology

If you are upgrading to the JetStream from the AE-32 Audio Engine, there is a fundamental difference between the AE-32 topology and the JetStream. In the AE-32, there was a single Supervisor computer connected to each Audio Engine via a serial cable using one common trigger table.

In the JetStream system, we split Supervisor into pieces and turned it into JetStream Server. Instead of one single Supervisor computer that kept track of every engine, we embedded a computer inside each JetStream. JetStream Server is responsible for executing commands for itself. We use one Trigger Table with commands for every JetStream and upload the same Trigger Table to each JetStream Server. JetStream Server knows to execute its own commands and leave the commands for the other JetStream Servers to execute themselves. This allows an action on one JetStream to fire a command on another JetStream without a centralized computer.

Networking

There are two types of network connections on every JetStream; the Audio LAN and the Admin LAN. Generally speaking, the Admin LAN is used for the embedded computer. Automation systems such as Sony ELC in Television and ENCO in Radio can make guest connections to JetStream Server on the Admin LAN. In addition, when you upload a trigger table to a remote JetStream, you will do this across the Admin LAN.

However, the communications for each JetStream Server to send control communications to each other happen on the Audio LAN. The same multicast channel that is used for JetStreams to discover each other and announce their networked sources is used to send commands for Command Builder Triggers from one JetStream to another. Technically speaking, you could have the Admin Net disconnected from every JetStream and still have triggers firing between JetStreams over the Audio Net; it just would make administering the system more difficult.

Where files are stored

JetStream Server.exe and CommandBuilder.exe are located in C:\Logitek or D:\Logitek.
The trigger table should always be located in C:\Logitek\Configs or D:\Logitek\Configs with the .xls JetFiles from each JetStream.

TIP: JetStream Minis with serial numbers 1 – 199 have their Logitek folder on the D drive. JetStream Minis with serial numbers 200 and above as well as the JetStream Plus and AE-IP have their Logitek folder on the C drive. Moving forward in this manual we will refer to the C drive only; if you have one of the older JetStream Minis you will be using drive D.

How Command Builder knows the names of sources and destinations

On the Administration page of JetStream Server, clicking **Save to JetFile** will save a copy of the entire JetStream configuration to an Excel formatted file (xls, not xlsx). Place a copy of the Excel file for each JetStream in C:\Logitek\Configs.

When Command Builder opens, it will read every Excel file in C:\Logitek\Configs and compile a list of sources and destinations for every JetStream it finds in the folder.

Do not keep backup copies of the same JetStream's JetFile in C:\Logitek\Configs. Command Builder will not be able to tell the difference between your current and backup files and this will cause triggers to fail. If you wish to keep backup copies of your JetFiles, place them in a separate folder outside of the Configs folder.

TIP: JetFiles are only valid when stored in xls format. If you open them in modern versions of Excel, do not convert them to the current format of Office. Do not use Google Apps to edit JetFiles. A free alternative to Excel that will properly read and write a JetFile is Open Office or Libre Office.

Command Builder will use the list of sources and destinations it compiles to convert the Unique Name from JetSet into a hexadecimal device number that the JetStream understands. A pick list in Command Builder lets you choose sources and destinations.

If you change unique names on sources or destinations or if you add or subtract inputs or outputs on the JetStream, as a bit of housekeeping you should take a new JetFile for that JetStream, open Command Builder, edit any triggers that were calling for sources and destinations that you had changed, and upload the trigger table again to every JetStream.

We recommend using one JetStream as the primary place to write your triggers and store the JetFiles, and then back up your work to the other JetStreams. While file sharing comes disabled on the JetStream by default, you can enable sharing in Windows for the **Configs** folder to save files. (File sharing is not necessary for uploading, but it makes moving the JetFiles and backups much easier.)

Uploading

In order for JetStream Server to know what commands to process, you will need to upload your trigger from Command Builder to each JetStream Server in your system. This process compiles the trigger table into a binary format and makes it part of JetStream Server itself. It is not enough to just save the text file in Command Builder. Changes to your trigger table are effective on that JetStream immediately after Command Builder reports that the trigger table has been accepted by JetStream Server.

3 Installing CommandBuilder

This chapter describes the first time installation of *CommandBuilder*. If you are upgrading *CommandBuilder* to this version, certain aspects of this chapter are also relevant.

Software Installation

The **Logitek** *CommandBuilder* software is pre-installed on every JetStream in the Logitek folder.

Logitek software is developed without reliance on components that are not standard in Windows. Upon first execution, *CommandBuilder* will setup default registry keys and make configuration files in the program directory.

If you are upgrading to a new version of Command Builder, simply replace the old CommandBuilder.exe in the Logitek folder with the new one. In many cases you will need to replace JetStream Server 5 with a new version when you are updating Command Builder 5. When in doubt, consult with Logitek support first.

The *CommandBuilder* program doesn't require any auxiliary or special files to run. The program creates user named trigger table files that are read or stored as ASCII text files with the file extension `.txt`. Some features of the *CommandBuilder* program require data available only in the `.xls` files created by the *JetStream Server* program. For this reason, it is required that the **Trigger Tables** be in the same folder as the *JetFile* configuration files in C:\Logitek\Configs or D:\Logitek\Configs. See "Where Files Are Stored" in chapter 2 for further information.

4 Working with Trigger Table files

This chapter outlines how to create, save, read and upload **Trigger Table** files. You do not need to be familiar with **Trigger** programming to use these functions.

The files are standard ASCII `.txt` files, with the addition of trigger and line numbers throughout the file. Whilst the files can be viewed and edited in a standard text editor, care should be taken not to disrupt the structure of the file – in particular no lines should be deleted or added.

CommandBuilder provides the ability to copy and paste blocks of **Triggers** and **Procedures**, to allow for faster duplication of similar code. Advanced users can also directly edit the **Trigger** file in a text editor, provided the above warnings are heeded.

When you first open *CommandBuilder*, you will see a number of buttons across the top. These are used to access core functions of the program.

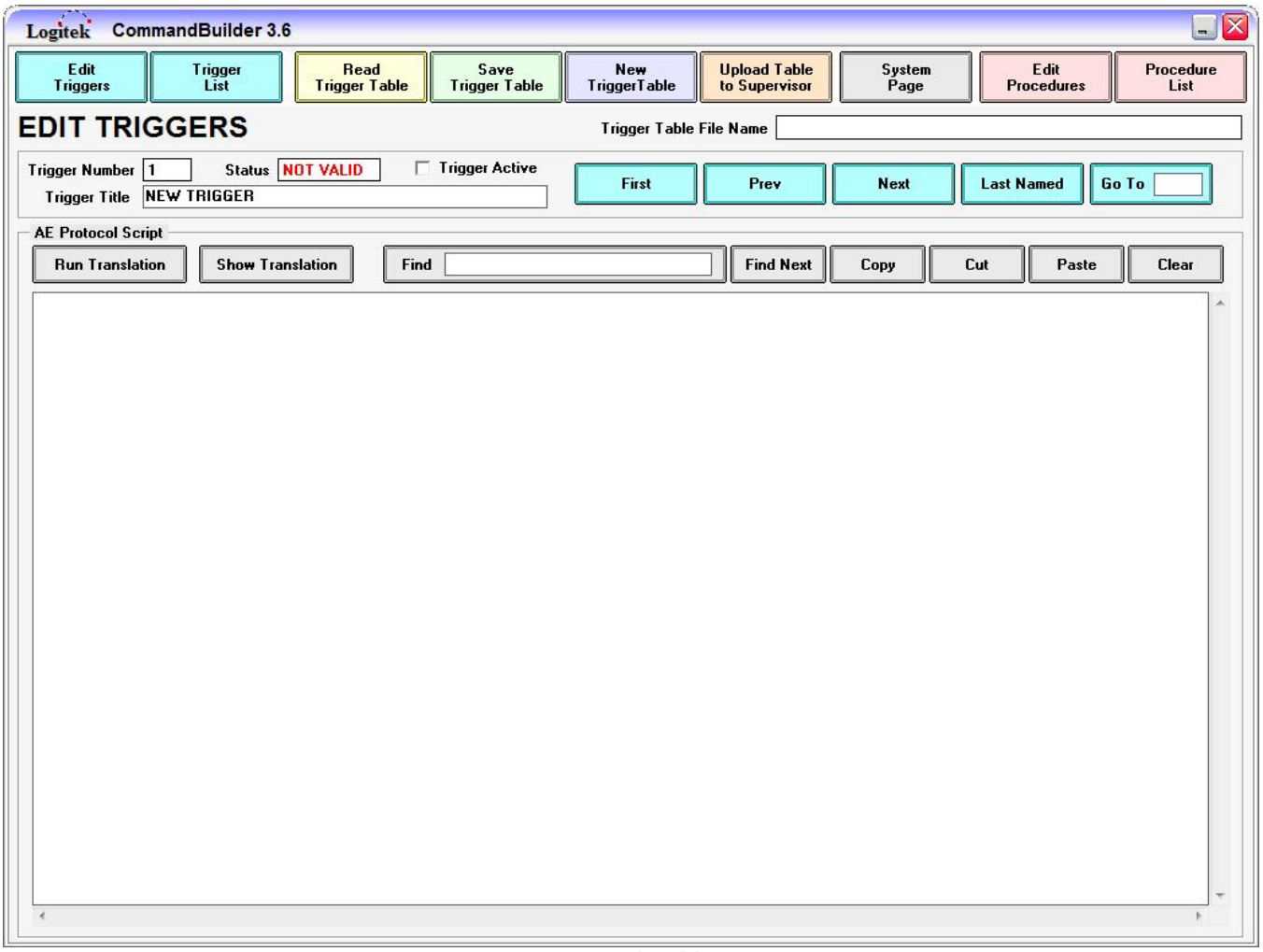


Figure 1 - CommandBuilder main page

Read Existing Trigger Table

The ***Read Trigger Table*** button allows you to read an existing file into *CommandBuilder*. A **Trigger Table** must be read before it can be edited or uploaded to *Supervisor*. Clicking the ***Read Trigger Table*** button displays a standard Windows Open File dialog box. Select the appropriate file and press ***Open*** to read the file. Choose ***Cancel*** if you accidentally click the button.

A warning message will be displayed if you attempt to read an existing **Trigger Table** after editing the current **Trigger Table** without saving.

The last file that was read or saved is automatically loaded the next time *CommandBuilder* is started. The ***Edit Triggers*** screen is automatically displayed after the **Trigger Table** file is read.

Create New Trigger Table

The ***New Trigger Table*** button allows you to create a new empty **Trigger Table**.

A warning message will be displayed if you attempt to make a new Trigger Table after editing the current **Trigger Table** without saving.

Any changes to the current file are not saved unless explicitly done so by the user.

Save Trigger Table

The ***Save Trigger Table*** button allows you to save the current **Trigger Table** in *CommandBuilder* to a file. Clicking the ***Save Trigger Table*** button displays a standard Windows Save As dialog box. Select an existing file name or enter a new file name and press ***Save*** to write the file to disk. Select ***Cancel*** to go back to the program without saving.

A warning message will be displayed if you attempt to make a new **Trigger Table**, read an existing **Trigger Table** or exit the program after editing the current **Trigger Table** without saving it. Any changes to the current **Trigger Table** are not saved unless explicitly done so by the user. Although not required, it is a good idea to save the current **Trigger Table** before uploading it to *Supervisor***.**

TIP: Command Builder saves a backup file automatically in C:\Logitek\Configs\Backup with the date and time appended to the file name every time you save the trigger table.

Upload Trigger Table

Trigger Tables must be uploaded to *JetStream Server* to be put into use on your **Logitek System**. You do not need to be familiar with **Trigger** programming to upload to *Supervisor*. In many cases, **Triggers** may be supplied by **Logitek**, a value added reseller or a systems integrator, and uploaded by the on-site system administrator.

The upload process

The ***Upload Table to JetStream*** screen contains three basic functions, as follows:

1. Connects to *JetStream Serverr* via the IP network.
2. Checks all active **Triggers** and **Procedures** for errors and compiles them.
3. Uploads the compiled **Trigger Table** to *Supervisor*.

The checking of the current **Trigger Table** for errors may be performed at any time, even if *CommandBuilder* is not connected to *Supervisor*.

Before you upload

In order to upload the **Trigger Table**, *CommandBuilder* must be connected to *JetStream Server* via the Admin LAN. Data fields on the right side of the ***System Page*** must be filled in correctly. In the ***JetStream Server*** box either the ***IP Address*** or ***Computer Name*** must be entered, as it appears in the ***Administration*** screen in the *JetStream Server*. The ***Server Port*** number must also contain the ***Server Port*** number found on the same ***Administration*** screen. The default ***Server Port*** number is `10200`.

In the ***User Profile*** screen in the *JetStream Server*, a valid user name and password must be entered in the proper fields, with the checkbox for "CommandBuilder" ticked. This data only needs to be entered once, and is usually setup when your system is installed.

> ⚠ Remember that you need to upload the same Trigger Table to every JetStream. If you are uploading to the same JetStream that you are using Command Builder, you can use the loopback address (127.0.0.1) to connect. After you have uploaded to your local JetStream, you then disconnect and reconnect to the next JetStream by entering its IP address.

TIP: You do not need to enter both the Computer Name and IP Address, Use one or the other. Command Builder can only resolve the computer name if you are using a Windows domain and Active Directory. Most sites do not do this and use the IP Address instead.

Uploading to JetStream

Click the ***Press to Connect*** button to attempt to connect to *Supervisor*. When successfully connected, the red status "Not Connected" bar will change to a green status "Connected" bar and the button title will change to ***Press to Disconnect***. The log is cleared and the ***Check Trigger Table*** button is also reset at this time.

Click the ***Check Trigger Table*** button to check all active **Triggers** and **Procedures** for errors. If any errors are found, the **Trigger Table** can not be uploaded to *JetStream Server*.

The ***Check Trigger Table*** button can be used at any time you want to verify the entire table, even when there is no connection to *JetStream Server*. A green progress bar at the top of the log window shows the status, and any problems are listed below. Possible errors include mistakes in **Triggers** or **Procedures**, duplicated **Trigger** events, or duplicated **Procedure** names.

When the checking is complete, a grid below the progress bar will display a summary of the number of duplicate, invalid, active and total **Triggers** and **Procedures**. If there are no errors or duplicates and *JetStream Server* is connected, the log will indicate that the **Trigger Table** may be uploaded.

Click the ***Upload Trigger Table*** button to transfer the compiled **Trigger Table** to *JetStream Server***.** A yellow progress bar in the log shows the status of the upload as it progresses. The progress bar will turn green when the **Trigger Table** has been successfully uploaded and accepted by *JetStream Server*.

If your upload stalls, click **Abort Table Upload**, then **Press to Disconnect**. Then reconnect and repeat. Most of the time this clears the issue. If not, abort, disconnect, restart *JetStream Server*, and reconnect and repeat the upload after *Server* has finished starting up.

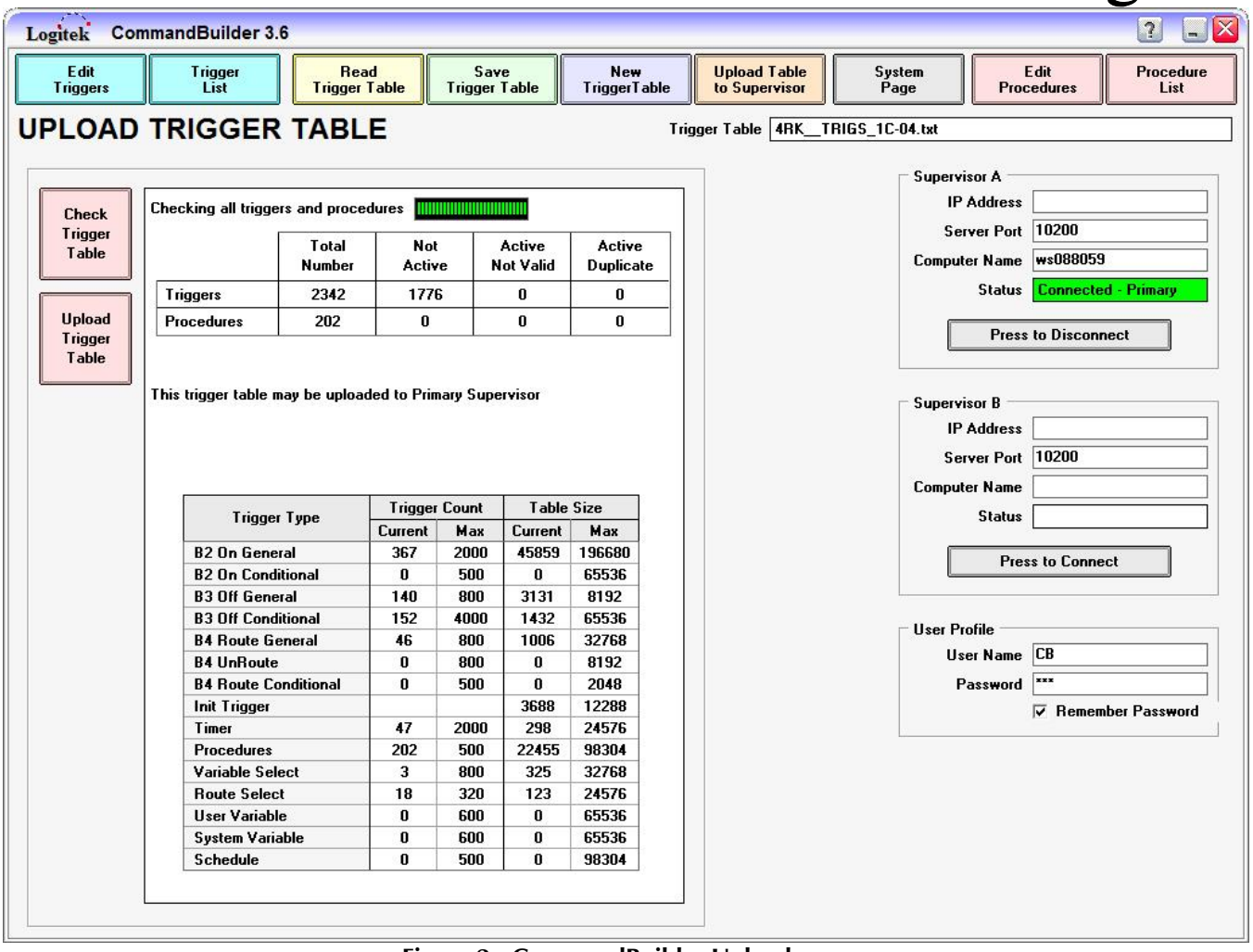


Figure 2 - CommandBuilder Upload page

After Uploading

Click the ***Press to Disconnect*** button to disconnect from *JetStream Server*. The connection is automatically closed when the *CommandBuilder* program is terminated.

If you have multiple JetStreams on the system, remember to repeat the upload process for each JetStream. You can **Press to Disconnect**, change the IP address to the next *JetStream*, and then **Press to Connect** to that machine. Once you have connected to that *JetStream Server*, you then repeat the process with **Check Trigger Table** and **Upload Trigger Table**.

5 Creating & Editing Trigger Tables

This chapter explains the tools required to setup a new **Trigger Table** and edit an existing one. The editing tools follow most Windows conventions and provide many functions for streamlining the process.

System Page

The ***System Page*** is used to setup information about a **Trigger Table** and the associated **Audio Engine** configuration files. The current **Trigger Table** file name and its associated project folder are displayed in the upper right hand portion of the screen.

TIP: The **Trigger Table** file is always stored in the same folder as the **JetFiles**: c:\Logitek\Configs or d:\Logitek\Configs. Every time you save the **Trigger Table**, a backup is created in c:\Logitek\Configs\Backup. Backup copies of JetFiles CANNOT be left in the same folder as the current configs. See chapter 2 for further details.

The ***System Page*** is used to create a link between the **Trigger Table** and the **Audio Engine Configuration** files. This allows *CommandBuilder* to reference **Audio Engine** resources by name, rather than by **Device Number**. The friendly name of an input or output is not only easier to read and work with, but is less likely to change during configuration changes.

In addition, **Trigger** programming for large, networked systems is greatly aided, as the same input name will almost certainly have different **Device Numbers** across each **Engine**. By referencing the input by name, only the **Audio Engine** number needs to be changed.

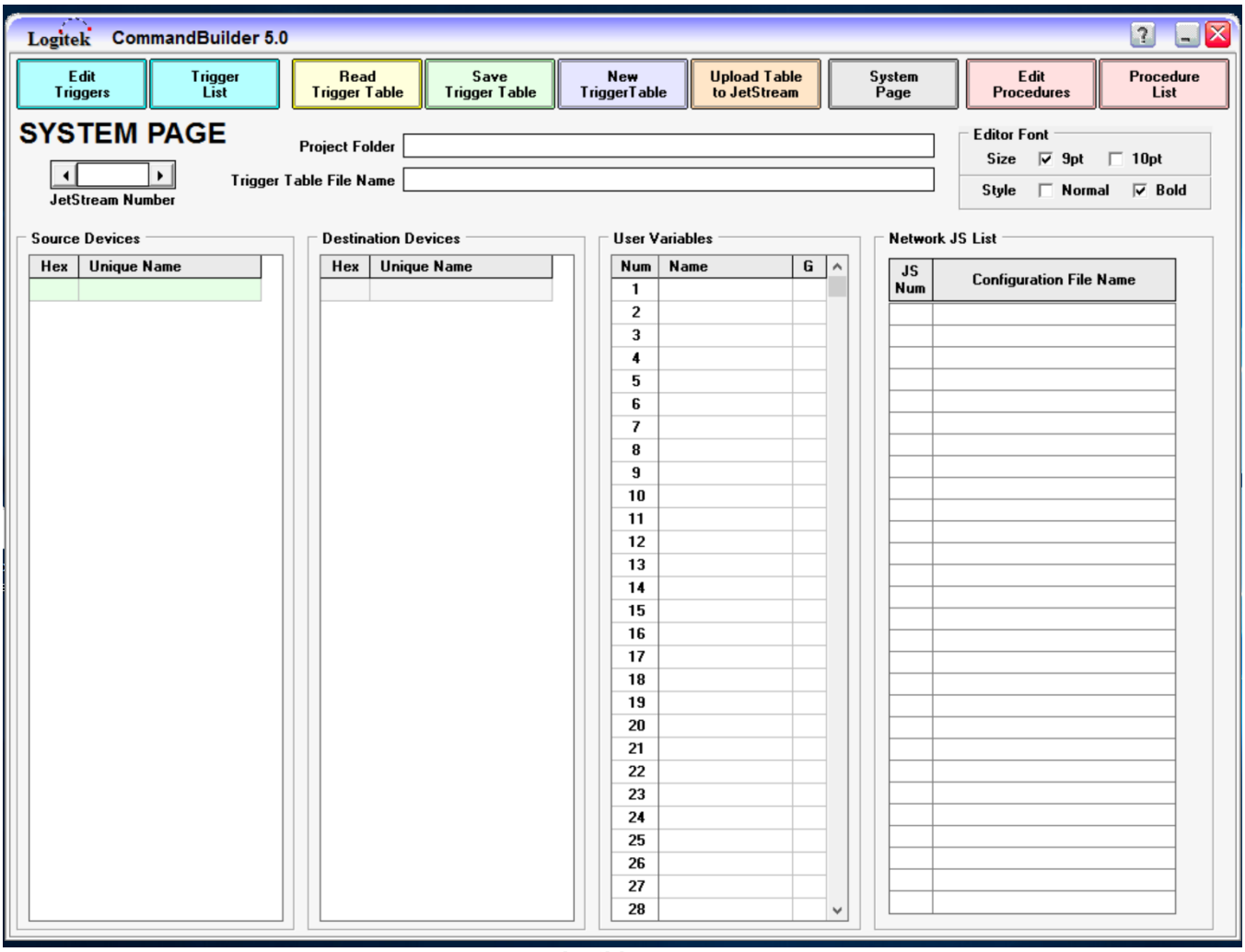
User Variables

The System Page is also used to declare the names of variables used in the Trigger Table. A variable must be declared on this page before you can use it. You will receive an error when you translate a Trigger or try to upload the table if a variable has been used but not declared.

To enter a new variable name or edit an existing variable, double click the appropriate cell in the list. You can also use the up and down arrows to navigate the list, the F2 key to edit a name, and enter to accept.

➥ *Using variables in Triggers is covered in later Language Reference chapters.*

TIP: User Variables are prefixed with a small "v" when they are referenced in a **Trigger** or **Procedure**. Do not enter the "v" when naming variables on the System Page.

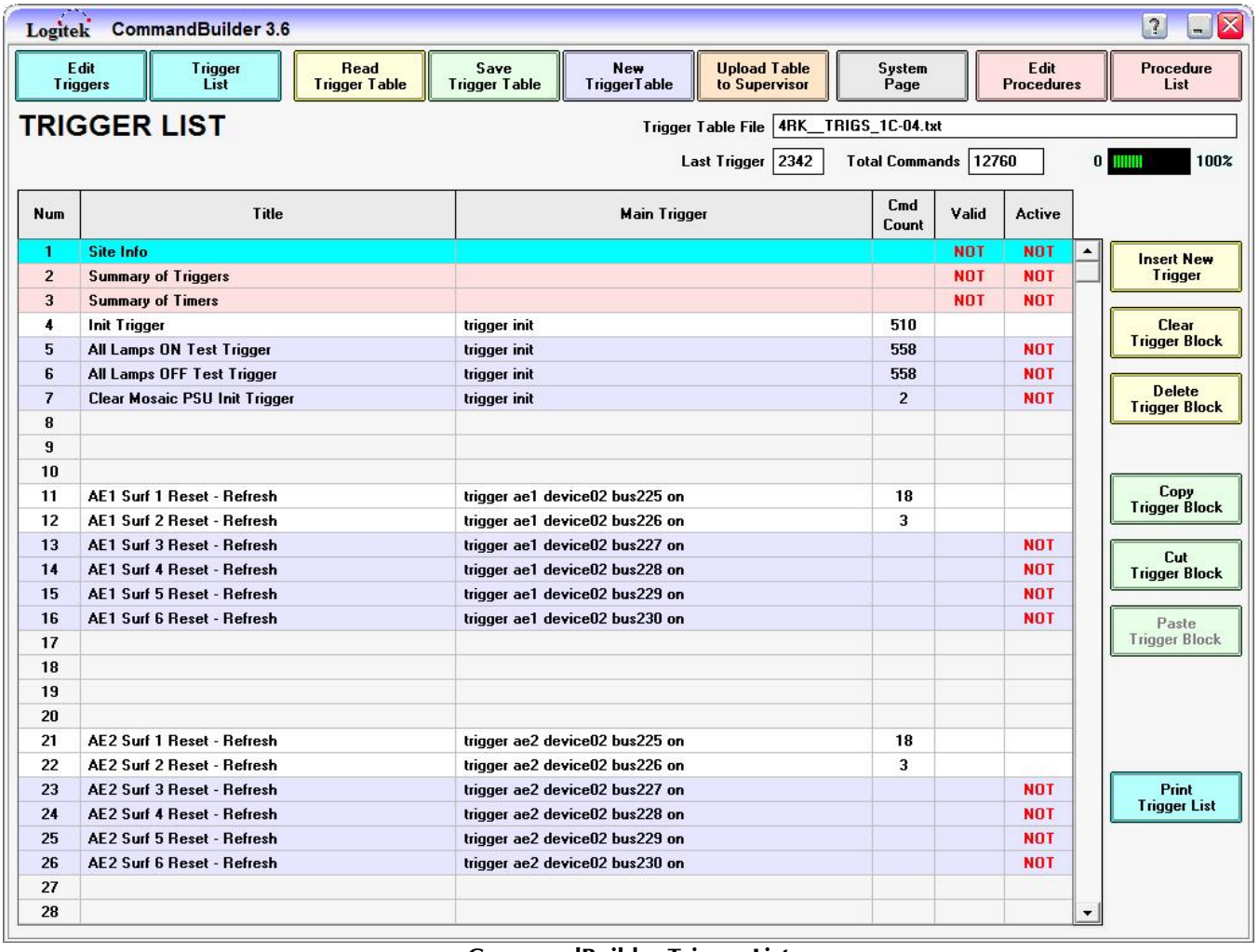


CommandBuilder System Page

Triggers & Procedures

The uses of **Triggers** & **Procedures** are covered in detail in the Language Reference chapters of this manual. In short:

- **Triggers** are events that contain commands to be executed when something happens in your **Logitek System**.
- Procedures are reusable code blocks that can be called by **Triggers**, or other **Procedures**. These are useful for reducing code repetition for frequently used functionality.



CommandBuilder Trigger List

Trigger & Procedure Lists

Use the buttons at the top of *CommandBuilder* to access the ***Trigger List*** and ***Procedure List*** screens. These screens provide a summary of the **Triggers** and **Procedures** in the **Trigger Table**. The functions on both screens are similar.

Each screen contains a grid that displays the **Trigger** or **Procedure** number, along with its descriptive title, the number of commands, whether it is valid and whether it is active. The ***Procedure List*** also shows the name of the **Procedure**, whilst the ***Trigger List*** shows the **Main Trigger** it will fire on.

Both screens also show the file name of the **Trigger Table** you are currently working with, as well as a percentage bar showing the amount of available space occupied by **Triggers** and **Procedures**. *CommandBuilder* has been tested on very large systems with thousands of triggers, so it is unlikely you will run out of internal memory for **Triggers** and **Procedures**.

The grid can be scrolled to view all **Triggers** and **Procedures**. Use the arrows on the scroll bar to move through the list slowly, and the page up/down function of the scroll bar to move quickly.

Double clicking on a desired **Trigger** or **Procedure** line will open the edit window, detailed in the next section. This will occur even if the line is blank.

Planning Trigger & Procedure Allocation

Like any programming language, it pays to keep associated functionality in groups, and leave room for the addition of future **Triggers** and **Procedures**. Experience has shown that the most logical layout of your **Trigger** list is to keep the related **Triggers** together – i.e. setup the **Audio Engine** GPI on and off triggers, ordering them by GPI input number. It also pays to leave some blank lines between each group.

➥ *See the Trigger Examples later in the manual to get a feel for a logical Trigger layout.*

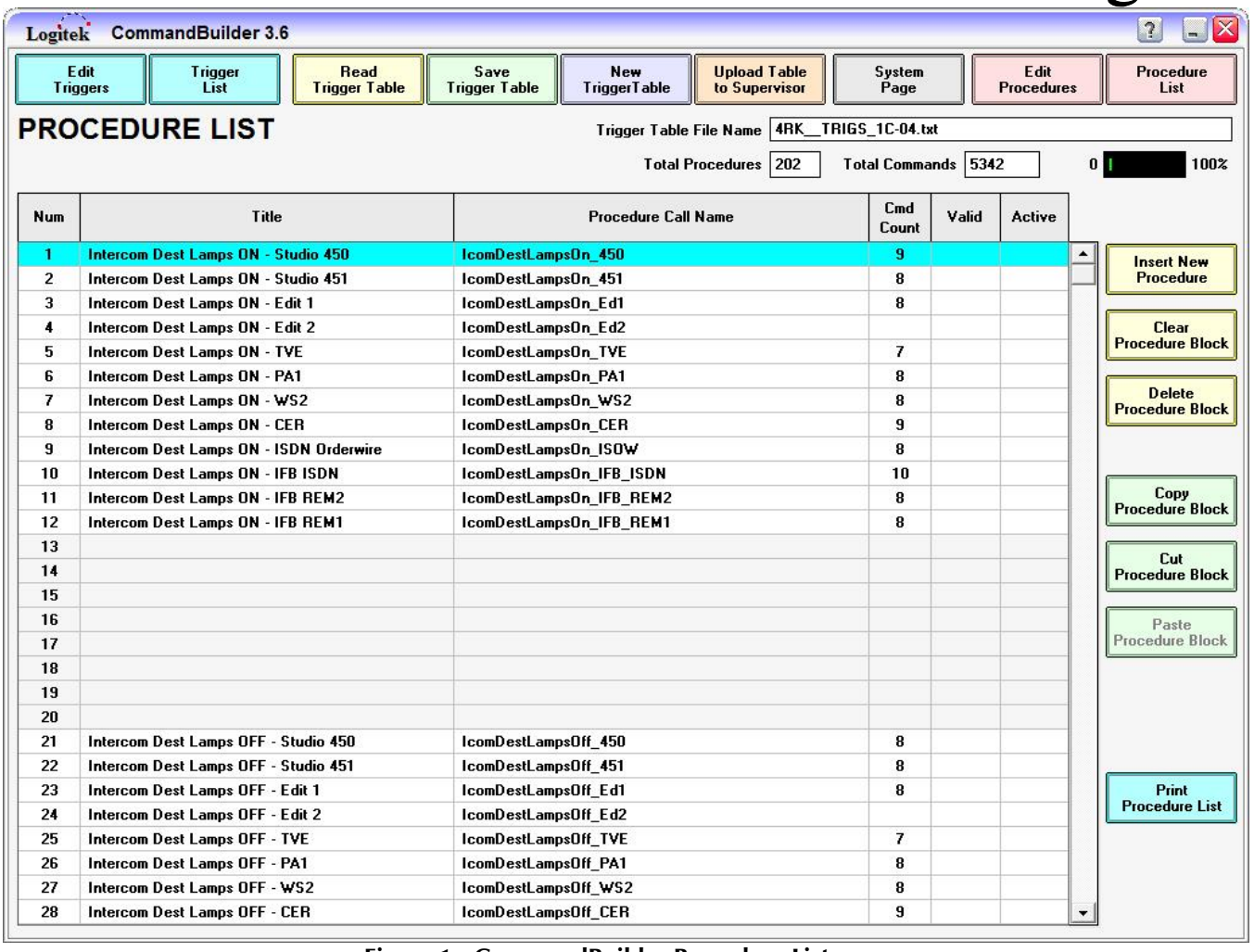


Figure 1 - CommandBuilder Procedure List page

Insert New Trigger/Procedure

This function will insert a new **Trigger** or **Procedure** directly above or below the line that is selected. If you wish to enter a **Trigger** or **Procedure** against a blank line, you do not need to use the ***Insert*** function; you can just double click the line you wish to use.

Selecting a block of Triggers or Procedures

To select a block of **Triggers** or **Procedures** click on the desired first **Trigger** or **Procedure** and drag to the desired last item in the list. Alternatively you can click the first item, then shift-click the last item to select the desired block. This allows you to scroll through the item list to select a large block.

A highlighted block can be cut, copied or pasted using the buttons described below.

Copy Trigger/Procedure Block

Once you have selected a block, click the ***Copy Trigger Block*** or ***Copy Procedure Block*** button to copy that block into the clipboard. These functions are used to duplicate **Trigger** and **Procedure** blocks.

Cut Trigger/Procedure Block

You can also cut a block of **Triggers** or **Procedures** to the clipboard, using the ***Cut Trigger Block*** or ***Cut Procedure Block*** functions. In addition to putting the block into memory, you will be asked whether you wish to move the remaining **Triggers** or **Procedures** up to fill the cut block. This is useful for moving an entire set of **Triggers** or **Procedures** to another location.

Paste Trigger Block

The ***Paste Trigger Block*** button when clicked, will pop up an option screen. From this screen, you can choose to ***Insert Before***, ***Insert After***, ***Overwrite*** or ***Cancel***. The first two options will insert the selected block from the clipboard to the list just before or after the item that is currently highlighted. The ***Overwrite*** option will insert the selected block from the clipboard in place of the item or items that are currently highlighted. The ***Cancel*** option will cancel the Pasting of the block. Once you have pasted a block, it is discarded from the clipboard.

Paste Procedure Block

The ***Paste Procedure Block*** button inserts the selected block from the clipboard to the list just before the item that is currently highlighted. Once you have pasted a block, it is discarded from the clipboard.

Editing Triggers & Procedures

The ***Edit Triggers*** and ***Edit Procedures*** buttons open an editing screen where the respective **Trigger** or **Procedure** can be created or edited. The editing interface is very similar between both **Triggers** and **Procedures**. Only the ***Edit Triggers*** page is shown below – the same functions are available in the ***Edit Procedures*** page. In each edit window, the current **Trigger Table** file name is displayed for reference. The file name will be blank for a new table.

The **Trigger** or **Procedure** number and its current valid or invalid status are also displayed. When making a new item, a description should be entered in the ***Name/Title*** box. This is displayed in the list summary view, and should briefly describe the function and relevant information.

A check box allows the **Trigger** or **Procedure** to be made active or not active. Items that are unchecked (not active) will not be uploaded to *JetStream Server* and, therefore will not respond to events.

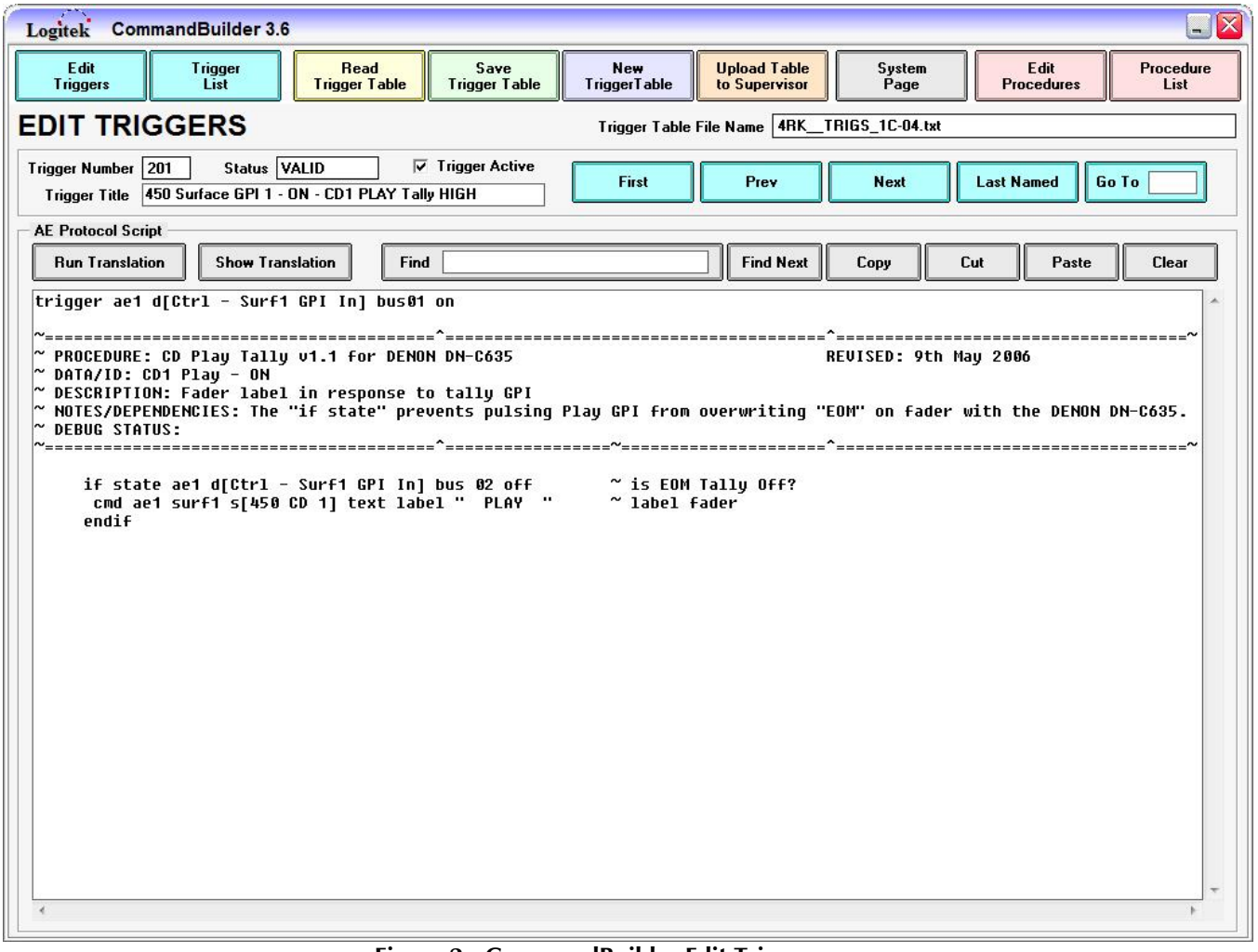


Figure 2 - CommandBuilder Edit Triggers page

Navigation Buttons

A group of buttons is provided to move quickly between items in the table without going back to the list view. Many of these buttons also have keyboard shortcuts, listed at the end of this chapter.

The ***First*** button will display the first **Trigger** or **Procedure** in the file.

The ***Prev*** button will go the previous **Trigger** or **Procedure** in the list. This item will be displayed even if it is blank.

The ***Next*** button will go to the next **Trigger** or **Procedure** in the list. This item will be displayed even if it is blank.

The ***Last*** button will display the last **Trigger** or **Procedure** that has a name that is not blank.

The ***Go To*** button allows an item number to be entered in the box. Pressing the ***Go To*** button, or Enter, will display the numbered **Trigger** or **Procedure** even if it is blank.

Function Buttons

The ***Edit Triggers*** and ***Edit Procedures*** screens have a text editing window in which commands are created and edited. The edit window works like most text editors and may be scrolled left or right and up or down. Blocks of text may be highlighted by clicking and dragging the mouse over the desired text.

The ***Run Translation*** button will cause *CommandBuilder* to scan the text and attempt to compile it. Commands are not checked as you write, but only when the ***Run Translation*** button is pressed.

The ***Show Translation*** button will display an additional ***Machine Hex*** window that shows the commands translated into **Logitek** hex code. This window is intended for advanced debugging and error message display purposes. If there is a problem translating commands to hex code, the error message will be displayed in red in the ***Machine Hex*** window. During debugging, if you have changed the trigger, you must press the ***Run Translation*** button again to retranslate the commands to hex code. The ***Machine Hex*** window can be hidden by pressing the ***Hide Translation*** button.

The ***Find*** button finds the first **Trigger** or **Procedure** that contains the word or phrase that has been entered in the text box. The ***Find*** function always scans from the first **Trigger** or **Procedure**. The ***Find Next*** button finds the next **Trigger** or **Procedure** that contains the word or phrase in the ***Find*** edit box.

The ***Cut***, ***Copy*** and ***Paste*** buttons act like standard Windows text editing functions, allowing for manipulation of highlighted text.

The ***Clear*** button completely erases everything in the text edit window. The ***Clear*** button will not erase the contents of the clipboard.

Keyboard Shortcuts

The following keyboard shortcut keys can be used to assist with text editing:

Home	Go to the beginning of the current line.
End	Go to the end of the current line.
CTRL-Home	Go to the beginning of the current trigger.
CTRL-End	Go to the end of the current trigger.
ALT-D	Display the source and destination device lists.
ALT-K	Delete from cursor to end of line.
ALT-L	Highlight entire current line.
ALT-V	Display User Defined Variable list.
Del	Delete next character or highlighted lines.
Keypad +	Copies current line or highlighted lines to Clipboard.
Keypad -	Cuts current line or highlighted lines to Clipboard.
Insert	Same as Paste button.
CTRL-C	Same as Copy button.
CTRL-X	Same as Cut button.
CTRL-V	Same as Paste button.
PageUp	Same as Prev button.
PageDown	Same as Next button.

Part B: Logitek Scripting Language

This section of the manual covers the Logitek Scripting Language, starting with an overview of the summary, followed by a command reference.

6 Language Overview

The **Logitek Scripting Language** allows the user to write **Triggers** and action commands that control the operation of **Audio Engines**. These events and commands are written in the *CommandBuilder* program and uploaded to the *JetStream Server* program. The **Audio Engine** only understands its own **Audio Engine Protocol**. This Protocol is published and available to the user if required. The **Logitek Scripting Language** allows control of the **Audio Engine** using commands that are more user friendly than those of the **Audio Engine Protocol** itself.

Triggers

Triggers are user defined events in the **Audio Engine** or control surface. For example, an event may be a button press or release, the opening or closing of a relay, or even a routing assignment. When *JetStream Server* detects the trigger event, any associated action commands from a **Trigger** are executed.

There are three types of triggers: **General Triggers**, the **Init Trigger**, and **Conditional Triggers**.

General Triggers

General Triggers are **Audio Engine** events that *JetStream Server* watches for at all times. **General Triggers** can be the turning ON or OFF of a button or relay, the "route" or "unroute" of an input assignment, or the change in the value of a variable.

➥ *See Chapter 7 for more details on General Triggers.*

Init Trigger

The **Init Trigger** is a special **Trigger** that occurs each time the *JetStream Server* program starts and whenever the user clicks the ***Execute Init Triggers*** button in *JetStream Server*.

➥ *See Chapter 8 for more details on the Init Trigger.*

Conditional Triggers

A **Conditional Trigger** is a **Trigger** that responds when a **Conditional** event occurs, for example a timer. A **General trigger** must be used to set a **Conditional Trigger**. After being started by a **General Trigger**, the firing of the timer is the **Conditional Trigger** event.

➥ *See Chapter 9 for more details on Conditional Triggers.*

Procedures

Procedures are sets of action commands that can be called by name in other **Triggers** and **Procedures**. They are like subroutines in programming or macros in word processing, and allow a repetitive group of action commands to be stored in one place and used many times. Each **Procedure** is given a unique user defined name. A simple call of the procedure name replaces typing many lines of action commands in a **Trigger**. The use of **Procedures** reduces the overall size of the stored **Trigger Table** by avoiding the duplication of frequently used code. They also improve the manageability of the **Trigger Table** by reducing the number of places where changes might need to be made. We strongly encourage the use of **Procedures** for code blocks that are used in many **Triggers**.

Unlike full programming languages, **Procedures** in the **Logitek Scripting Language** cannot accept any arguments or return values to the calling function. However, through the smart use of **User Variables**, similar functionality can be achieved.

➥ *See Chapter 11 for more details on Procedures.*

Action Commands

Action Commands are direct commands to the **Audio Engines** or to *JetStream Server* that are executed when their **Trigger** event or **Procedure** occurs. **Action Commands** make the things happen that we want to occur in an event. These commands can be entered in both **Triggers** and in **Procedures**, and follow the **Trigger** or **Procedure** definition.

➥ *See Chapter 12 for more details on Action Commands.*

General Syntax

All **Triggers**, event declarations and **Action Commands** must be completed on one line. **Triggers** and action commands can not be continued on a second or subsequent line.

The ~ (tilde) character is used to denote the beginning of documentation comment. All words to the right of the tilde character are ignored. Thus if the first character in a line is the tilde character, the entire line is considered a comment. The tilde character may be used to temporarily "comment out" certain lines for testing. These lines can then be restored to service by removing the tilde character, saving you from re-typing the entire line. The tilde character is not allowed on the first line of a **Trigger** or **Procedure**. This line is reserved for the definition of the **Trigger** event or the **Procedure** name. All comments are stored in the **Trigger Table** and are available for future reference. A documentation banner near the beginning of a **Trigger** or **Procedure** with further descriptions throughout the command listing is considered good practice.

The scripting language decodes the key words that it finds on each line. Keyword entries in the scripting language are not case sensitive. They may be entered in upper, lower or mixed case as desired. All keywords must be separated by at least one blank space on either side. All words that are not defined or recognized by the scripting language are treated as spaces. Since extra spaces and words are ignored, they may be used to improve the readability of command lines. Unless noted, the keywords may be in any order on the line. The recommended order shown in examples is for the most logical readability.

➥ *See Appendix A for a complete list of keywords.*

Device Numbers

All signals in the **Audio Engine** are referenced by an internal number called a **Device Number**. Each input signal is assigned a unique **Device Number** by the *AEConfig* program. Each output signal is assigned a **Device Number** defined by the DSP table associated with a configuration file, or assigned by an output in *AEConfig*. Input **Device Numbers** are called **Source Devices** and the output **Device** numbers are called **Destination Devices**. Most commands sent from *JetStream Server* to the **Audio Engine** refer to a specific **Source** or **Destination Device Number**.

Source Devices are referred to using the keyword `DEVICE` followed by the device number in hexadecimal notation. For example, `DEVICE 010B` could refer to a **Source Device** for an input on a given **Audio Engine**.

Destination Devices are referred to using the keywords `CHANNEL` or `FADER` followed by the console channel or fader number in decimal notation. The console channel or fader number is translated by *CommandBuilder* to the proper destination device number in hexadecimal. You can also directly enter the hex device number if known, although for code readability this is not recommended.

Source Devices can also be referred to using the "s bracket" notation – the device name is enclosed in square brackets prefixed with the letter "s". For example, a device named "Host Mic" could be referred to as `s[Host Mic]` in a trigger or action command.

Destination Devices can be referred to using the "d bracket" notation – the device name is enclosed in square brackets prefixed with the letter "d". For example, a device named "Fader 1" could be referred to as `d[Fader 1]` in a trigger or action command.

The bracket notation is not dependent on knowing device numbers and is automatically looked up by *CommandBuilder* when triggers are uploaded. This greatly simplifies the use of **Device Numbers** over previous versions of *CommandBuilder*.

Pressing the ***Alt-D*** key combination will display a pop-up window from which Source and Destination Devices can be selected and inserted into the code using the "bracket" notation.

7 Trigger Types – General Triggers

Introduction

General Triggers are **Audio Engine** or **Surface** events that *JetStream Server* watches for at all times. When the event occurs, *JetStream Server* responds by executing the action commands listed for the **General Trigger**.

The basic form of a **General Trigger** is "`TRIGGER DEVICE ACTION`" where `TRIGGER` is the required keyword; `DEVICE` is where the action will occur, and `ACTION` is what happens. The DEVICE description includes items such as **Audio Engine** and **Surface** numbers, **Device** number, **Bus** number, button names, variable names and **Source** or **Device** names. The `ACTION` item includes things like `ON, OFF, ROUTE, UNROUTE, SET,` and `CHANGE`.

General Rules

- The keyword `TRIGGER` must be the first word on the first line.
- The first line can't be a comment line.
- A **Trigger** need not contain **Action Commands** – it can be a definition only.

Declaring General Triggers

A **Trigger** is defined by placing the appropriate **Trigger** command on the first line of an available **Trigger** in the ***Trigger List*** page. Once you have found the **Trigger** you wish to define, double click that row to edit the **Trigger** text. The **Trigger** definition is then the first line in the edit box.

The **Trigger** must be marked active for *JetStream Server* to process the **Action Commands** that follow it.

General Trigger Types

There are six types of **General Triggers**, as follows:

Type	Description	Example Scenarios
ON	An ON Trigger is fired when a channel ON event occurs	Button pressed, GPI input goes high
OFF	An OFF Trigger is fired when a channel OFF event occurs	Button released, GPI input goes low
ROUTE	A ROUTE Trigger is fired when a specified route occurs to a specified device.	Specific Input routed to fader
UNROUTE	An UNROUTE Trigger is fired when a specified route is replaced on a specified device	Other Input routed to fader
USER VARIABLE	A USER VARIABLE Trigger is fired when there is a change to the value of a user defined variable	User Variable value change
SYSTEM VARIABLE	A SYSTEM VARIABLE Trigger is fired when there is a change to the value of a system variable.	System Variable value change

The **General Triggers** are described in more detail on the following pages.

General Trigger Format

On Trigger

Fires when the specified Bus On event occurs.

Keyword	Engine	Surface	Device	Bus	State	Option
trigger	ae#	*Not Used*	device#### *(dest)*	bus#	on	{toggle}
trigger	ae#	*Not Used*	device#### *(source)*	bus#	on	{toggle}
trigger	ae#	surf# | surface#	fader#	bus#	on	{toggle}
trigger	ae#	surf# | surface#	chan# | channel#	bus#	on	{toggle}
trigger	ae#	*Not Used*	d[*DeviceName*]	bus#	on	{toggle}
trigger	ae#	surf# | surface#	s[*DeviceName*]	bus#	on	{toggle}
trigger	ae#	surf# | surface#	bridge | softkey	butt# | button#	on	{toggle}

- This **Trigger** must include the **Audio Engine** number, **Surface** number, **Device** or **Fader** number, **Bus** number and the required keyword `ON`.
- One of the keywords `CHANNEL`, `FADER`, `DEVICE`, `BRIDGE` or `SOFTKEY` is required.
- The required **Device** or **Fader** number may be given using the `s[Source Device]` or `d[Destination Device]` notation instead of using the `DEVICE` keyword.
- If using a **Source Device**, the **Surface** number is not required and will be ignored if present – the **Trigger** will be fired no matter which **Surface** on the **Engine** causes the **Bus On** event.
- If using the `BRIDGE` or `SOFTKEY` **Device** keyword, use `BUTT` or `BUTTON` number.
- There are two types of **On Triggers** – the **Single State Trigger** and the **Toggle Trigger**.
- Single state **On Triggers** are like momentary contact switches – when they are pressed the commands are executed each time.
- A **Toggle Trigger** will switch between two different states like a push-on/push-off button. Different sets of **Action Commands** can be associated with the two states. Each time the **Toggle Trigger** fires, only the action commands associated with the current state are executed, and then the state is changed to the opposite value ready for the next execution.

The following are examples of **Single State On Triggers**:

```
trigger ae3 surface1 device 000B bus0 on
trigger ae4 d[Port 2 Fader 1 In] on
trigger ae1 surface2 softkey button2 on
```

The following are examples of **Toggle State On Triggers**:

```
trigger ae3 surface1 device 000B bus0 on toggle
trigger ae4 d[Port 2 Fader 1 In] on toggle
trigger ae1 surface2 softkey button2 on toggle
```

TIP: Toggle Triggers are useful for push-on/push-off function buttons, for example a button that changes monitoring to post-delay and restores it when pressed again.

➥ *See Chapter 17 for more details on using the If Toggle Equals Test Statement.*

Off Trigger

Fires when the specified Bus Off event occurs.

Keyword	Engine	Surface	Device	Bus	State
trigger	ae#	*Not Used*	device#### *(dest)*	bus#	off
trigger	ae#	*Not Used*	device#### *(source)*	bus#	off
trigger	ae#	surf# | surface#	fader#	bus#	off
trigger	ae#	surf# | surface#	chan# | channel#	bus#	off
trigger	ae#	*Not Used*	d[*DeviceName*]	bus#	off
trigger	ae#	surf# | surface#	s[*DeviceName*]	bus#	off
trigger	ae#	surf# | surface#	bridge | softkey	butt# | button#	off

- This **Trigger** must include the **Audio Engine** number, **Surface** number, **Device** or **Fader** number, **Bus** number and the required keyword `OFF`.
- One of the keywords `CHANNEL`, `FADER`, `DEVICE`, `BRIDGE` or `SOFTKEY` is required.
- The required **Device** or **Fader** number may be given using the `s[Source Device]` or `d[Destination Device]` notation instead of using the `DEVICE` keyword.
- If using a **Source Device**, the **Surface** number is not required and will be ignored if present – the **Trigger** will be fired no matter which **Surface** on the **Engine** causes the **Bus Off** event.
- If using the `BRIDGE` or `SOFTKEY` **Device** keyword, use `BUTT` or `BUTTON` number.
- The **Off Trigger** does not have a **Toggle State**, it only has a **Single State**.

The following are examples of an **Off Trigger**:

```
trigger ae3 surface1 device 000B bus0 off
trigger ae4 d[Port 2 Fader 1 In] off
trigger ae1 surface2 softkey button 2 off
```

TIP: When writing button Triggers, an Off Trigger is only required if commands should occur when the button is released, otherwise you only need an On Trigger.

Route 3 Button Trigger

A variation of the On Trigger for use with Route 3 CH1 and CH2 button presses.

Keyword	Engine	Surface	Device	Bus	State
trigger	ae#	*Not Used*	device#### *(dest)*	butt # | button#	on

- The *Route 3* must be in **Message Mode** for this **Trigger** to occur.
- This command only supports a hard coded **Destination Device** number. It does not support the `SURFACE` or `CHANNEL`. It does not support the `d[Destination Device]` notation.
- The **CH1** button uses the keyword `BUTTON1`.
- The **CH2** button uses the keyword `BUTTON2`.
- There is only an **On Trigger** for the *Route 3* buttons. There is no **Off Trigger**.

TIP: You will need a Logitek Device Reference table to determine the correct Device.

Route Trigger

Fired when a specified Source Device is routed (assigned) to a specified Destination Device (e.g. fader).

Keyword	Engine	Device	Keyword	Destination	Option
trigger	ae#	device ####	{to}	chan# | channel #| fader #	{any}
trigger	ae#	s[*Device Name*]	{to}	d[*Device Name*]	{any}

- This **Trigger** must include the **Audio Engine** number, **Surface** number, **Source Device** number, **Destination Device** number and the required keyword `ROUTE`.
- The **Source Device** can be specified using keyword `DEVICE` or using the `s[Source Device]` notation.
- The **Destination Device** can be specified by using the keywords `FADER` or `CHANNEL` or using the `d[Destination Device]` notation. **Device** number is not supported.
- The use of the `TO` keyword is optional. If omitting the `TO` keyword, place the **Source Device** first for improved readability.
- If no **Source Device** is specified, the **Trigger** will fire whenever any source is routed to the specified **Destination Device**.
- If no **Destination Device** is specified, the **Trigger** will fire whenever the specified source is routed to any **Destination Device**.
- The optional keyword `ANY` can be used to improve the readability in these scenarios.

The following are examples of a **Route Trigger**:

```
trigger ae1 surface1 route device 0103 channel2
trigger ae3 route s[Port1 Aux 1 Out] to d[Port1 Fader 7 In]
trigger ae1 surface1 chan25 route any
trigger ae1 d[MD 1 Record] route any
```

🎧 TIP: If using a Route Trigger on a Route 3 device (or similar) which is configured to use a Direct Route address (e.g. 6E or higher), you cannot specify a Route Trigger using the Device Number, Surface Number or Channel Number. In this case you will need to use the d[Destination Device] notation.

Unroute Trigger

Fired when a specified Source Device is replaced (unassigned) on a specified Destination Device.

Keyword	Engine	Surface	Keyword	Source	Keyword	Destination
trigger	ae#	surf# | surface#	route	device ####	{from}	chan# | channel #| fader #
trigger	ae#	*Not Used*	route	s[*Device Name*]	{from}	d[*Device Name*]

- This **Trigger** must include the **Audio Engine** number, **Surface** number, **Source Device** number, **Destination Device** number and the required keyword `UNROUTE`.
- The **Source Device** can be specified using keyword `DEVICE` or using the `s[Source Device]` notation.
- The **Destination Device** can be specified by using the keywords FADER or CHANNEL or using the `d[Destination Device]` notation. **Device** number is not supported.
- The use of the `FROM` keyword is optional.
- If no **Source Device** is specified, the **Trigger** will fire whenever any source is replaced on the specified **Destination Device**.
- If no **Destination Device** is specified, the **Trigger** will fire whenever the specified source is replaced on any **Destination Device**.
- The optional keyword `ANY` can be used to improve the readability in these scenarios.

The following are examples of an **Unroute Trigger**:

```
trigger ae2 surface1 unroute device 0105 channel4
trigger ae8 unroute s[Port2 Aux 1 out] from d[Port2 Fader 3 In]
trigger ae1 surface1 chan25 unroute any
```

Variable Change Trigger

Fired when a value is assigned to a User or System Variable.

Keyword	Variable Name	Option
trigger	v[*UserVariable*] | z[*SystemVariable*]	{change}
trigger	v[*UserVariable*] | z[*SystemVariable*]	{change}

- This **Trigger** must contain the name of a **User Variable** using the `v[UserVariableName]` notation, or a **System Variable** using the `z[SystemVariableName]` notation.
- For **User Variables**, the name must be declared in the **System Page**.
- For **System Variables**, the name must be an internally declared *JetStream Server* **System Variable**.
- Appending the optional keyword `CHANGE` to the **Trigger** will cause the trigger to fire only if the new value is different from the previous value.
- Otherwise, this **Trigger** is fired each time a value is assigned, even if the value is unchanged.
- **System Variables** are read only variables that can only be set by *JetStream Server* itself.

The following **Trigger** will fire every time a value is assigned to the user variable `Studio5OnAirSource`, regardless of if that value has actually changed:

```
trigger v[Studio5OnAirSource]
```

The following **Trigger** will fire only if the new value of the variable is different from the previous value of the **User Variable** `Studio5OnAirSource`:

```
trigger v[Studio5OnAirSource] change
```

The following **Trigger** will fire every time a new value is assigned to the **System Variable** `SupervisorANetworkFail`:

```
trigger z[SupervisorANetworkFail]
```

The following **Trigger** will fire only if the new value of the variable is different from the previous value of the **System Variable** `SupervisorANetworkFail`:

```
trigger z[SupervisorANetworkFail] change
```

➥ *See Chapter 14 for more details on using User & System Variables.*

Special Case: AE 0 (AE Zero)

Any trigger that specifies `ae0` in place of the engine number will refer to the JetStream it is executed on.

Any command that specifies `ae0` in place of the engine number will refer to the JetStream is is executed on.

If the command that is specified for `ae0` calls for a device or function that does not exist on that JetStream, the command will be ignored.

Since the same trigger table is uploaded to every JetStream, this gives you the ability to write triggers that are common to several JetStreams only once and have them execute everywhere.

Example:

```
trigger ae0 bridgebutton 1 on
   cmd ae0 bridgelamp 1 on
```

8 Trigger Types – Init Trigger

Introduction

The **Init Trigger** is a special **Trigger** that is run each time that *JetStream Server* starts, and when the ***Execute Init Triggers*** button is clicked. The purpose of the **Init Trigger** is to perform one-off setup commands, for example writing text labels to surfaces.

The user can fire the **Init Trigger** at any time by pressing the ***Execute Init Triggers*** button on the **Audio Engine Log** page of the *JetStream Server*. Care should be taken in selecting the commands in an **Init Trigger** since these items will be executed every time *JetStream Server* is started. The **Init Trigger** is not required to be active or even exist.

General Rules

- The keyword `INIT TRIGGER` must be the first line of the **Trigger**.
- The first line can't be a comment line.
- There is no **Audio Engine** or **Surface** numbers for the **Init Trigger** – it is a global event.
- The **Init Trigger** is usually listed first in the **Trigger** list, but may be inserted at any position.
- The **Init Trigger** is not required – it is optional.
- There can only be one **Init Trigger** per **Trigger Table**.

Declaring the Init Trigger

The **Init Trigger** is defined by placing the keywords `INIT TRIGGER` on the first line of an available **Trigger** in the ***Trigger List*** page. Once you have found where you wish to place the **Init Trigger**, double click that row to edit the **Trigger** text. The **Init Trigger** definition is then the first line in the edit box.

Init Trigger Format

Init Trigger

Occurs on JetStream Server startup and when manually activated from JetStream Server.

Keyword	Keyword
trigger	Init

- The **Init Trigger** must be marked active for *JetStream Server* to process the **Action Commands** that follow.

Applications

The **Init Trigger** is useful for setting up various parts of your **Triggers** upon startup of *JetStream Server*. It can also be used to reset the state of variables, toggles, lamp indicators, etc to a known good scenario. This is particularly important if your **Trigger Table** performs logic tests that disable functions based on certain situations.

Some examples of applications for the **Init Trigger** are:

- Reset Not Active Triggers
- Reset User Variables used for tracking statuses
- Write text labels to Surfaces
- Setup clock and temperature display on Surfaces
- Set maximum delay time for Profanity Delay
- Update lamp indicators from variables

Before writing an **Init Trigger**, carefully evaluate the logic of your **Trigger Table** to determine what needs to be setup and reset in the **Init Trigger**. The **Init Trigger** should not perform tasks that could have a detrimental effect to your on-air operation.

9 Trigger Types – Conditional Triggers

Introduction

A **Conditional Trigger** is a **Trigger** that is set to respond the next time the conditional event occurs. A **General Trigger** is used to set a **Conditional Trigger**. There are three types of **Conditional Triggers**: `IF TRIGGER`, `IF BUTTON`, and `IF TIMER`.

Any **Conditional Trigger** can be cleared by a `CMD CLEAR` action command in another **General Trigger**.

A **Conditional Trigger** is useful for setting up **Trigger** events that will occur within or after a specified delay time.

TIP: It is easy to confuse **General** and **Conditional Triggers**. **General Triggers** are defined as an action that will run each and every time a specified event occurs. **General Triggers** are declared as the first line of a Trigger. **Conditional Triggers** are declared "on the fly" inside **General Triggers**. These can be set to run upon a particular event occurring, but can be later cancelled so they no longer occur.

General Rules

- **Conditional Triggers** are declared inside **General Triggers**.
- There can only be one **Conditional Trigger** for a particular event (e.g. timer number) – setting the same event **Conditional Trigger** a second time will replace the first set of commands.

Declaring Conditional Triggers

A **Conditional Trigger** is declared inside a **General Trigger**, using one of the three **Conditional Trigger** types detailed below. A **Conditional Trigger** cannot be declared on its own. The **Conditional Trigger** `IF` statement must have a corresponding `ENDIF`.

Conditional Trigger Types

There are three types of **Conditional Triggers**, as follows:

Type	Description	Example Scenarios
IF TRIGGER	Declares a Trigger "on the fly", which can later be replaced or cancelled	A Trigger needs to set temporarily, or the actions of a Trigger changed depending on circumstances
IF BUTTON	Similar to IF TRIGGER, but uses button keywords and can include timeouts if button is not pressed in a certain time	Setup an event to occur if the ACCEPT or CANCEL button is pressed
IF TIMER	Sets a Trigger that will occur after the time expires (specified time, or tied to Talk Delay)	Set an event to occur after a specified time period

Conditional Trigger Format

If Trigger

Sets a Conditional Trigger that will fire when the specified Trigger action occurs.

Keyword	Keyword	Trigger	Keyword
if	trigger	*TriggerDefinition* | *(TriggerDefinition)*	{*then*}

- The specification of the **Trigger** is the same as a **General Trigger**.
- The **On Trigger**, **Off Trigger**, or **Route Trigger** may be used as a **Conditional Trigger**.
- Commands must appear between the `IF TRIGGER` and a corresponding `ENDIF` keyword.
- This **Conditional Trigger** can be canceled using the `CMD CLEAR TRIGGER` command.
- The keyword `THEN` and parenthesis are optional.

➥ *See Chapter 7 for more information on General Trigger syntax.*

The following is an example of the **If Trigger Conditional Trigger**:

```
if trigger ( ae2 surface2 route s[Cass 1] to d[Port 2 Fader 3 In] ) then
 cmd ae2 surface2 d[Port 1 Fader 2 In] bus0 on
endif
```

If Button

Sets a Conditional Trigger that will fire when the specified button is pressed or released.

Keyword	Engine	Surface	Keyword	Button	Keyword	Keyword	Keyword
If	ae#	surf# | surface#	function | softkey | bridge	button #	{on | off}	{delay#}	then
If	ae#	surf# | surface#	accept | cancel	button	{on | off}	{delay#}	then

- The allowable button keywords are `FUNCTION BUTTON`, `SOFTKEY BUTTON`, `BRIDGE BUTTON`, `ACCEPT BUTTON` and `CANCEL BUTTON`.
- A button number can be used with the keywords `FUNCTION`, `SOFTKEY` & `BRIDGE`.
- The action keywords for the buttons are `ON` and `OFF`. `ON` is the default if not specified.
- The optional keywords `TIMEOUT` or `DELAY` followed by a number of seconds can be used to automatically fire the **If Button Conditional Trigger**.
- Commands must appear between the `IF BUTTON` and a corresponding `ENDIF` keyword.
- This **Conditional Trigger** can be canceled using the `CMD CLEAR BUTTON` command.

The following are examples of the **If Button Conditional Trigger**:

```
if ae1 surf1 button cancel delay30
 cmd ae1 surf1 question text clear all
endif
```

➥ *See Page 75 for more information on the Clear Trigger command.*

If Timer

Sets a Conditional Trigger that will fire when a specified time period elapses.

Keyword	Keyword	Number	Keyword	Seconds	Keyword
if	timer	#	wait	#	{*then*}

- The required keyword `WAIT` is used to specify the time in seconds before the timer fires.
- Each timer must have a unique user defined number. This number is used in the `CMD CANCEL TIMER` action command that is used to cancel the timer and prevent its commands from executing.
- The **If Timer** requires a unique timer number.
- The timers are not specified for a given **Audio Engine** or **Surface**.
- Commands must appear between the `IF TRIGGER` and a corresponding `ENDIF` keyword.
- The keyword `THEN` is optional.

The following are examples of the **If Timer Conditional Trigger**:

```
if timer 3 wait 25 then
 cmd ae1 surface2 d[Port 1 Fader 2] bus 0 on
 cmd ae1 surface2 bridge button 3 off
endif
```

If Timer Talk Delay

Sets a Conditional Trigger that will fire when a time period equal to a Talk Delay elapses.

Keyword	Keyword	Number	Keyword	Destination	Keyword
if	timer	#	talk	d[*DeviceName*]	{*then*}

- Instead of specifying a `WAIT` time, the **If Timer** can be tied to a **Talk Delay**, to set a **Timer** equal to the current delay time of a **Talk Delay**.
- This function is useful when you need to delay a command to match an audio delay.
- When the **Timer** is set, the current delay for the specified **Destination Device** is used.
- If the **Destination Device** is not a **Talk Delay** device, or the **Talk Delay** is off, the wait time is set to zero and the enclosed commands execute immediately.
- The resolution is 0.1 seconds; ensuring commands match delayed audio very closely.
- The **If Timer Talk Delay** requires a unique timer number.
- Commands must appear between the `IF TRIGGER` and a corresponding `ENDIF` keyword.
- The keyword `THEN` is optional.

The following is an example of the **If Timer Talk Delay Conditional Trigger**:

```
if ae1 timer 102 talk d[Port1 Router 1 In] then
 cmd port 8 "MC1EVT1"
endif
```

10 Trigger Types – Schedule Event Triggers

Introduction

A **Schedule Event Trigger** is a special **Trigger** that responds at a user designated date and time. **Schedule Event Triggers** can be made to be activated just once or can be repeated at a specified interval. These **Triggers** can also be made to repeat on given days of the week.

General Rules

- The keywords `TRIGGER SCHEDULE` must be the first words on the first line.
- The date for the **Trigger** to fire must follow the keyword `DATE` in `DD MONTH YYYY` format.
- The time for the **Trigger** must follow the keyword `TIME`, with the time in `HH:MM:SS` 24-hour format.
- Other date and time formats (including the use of AM & PM) will result in an error.
- The first line can't be a comment line.

Declaring Schedule Triggers

A **Schedule Trigger** is defined by placing the appropriate command on the first line of an available **Trigger** in the ***Trigger List*** page. Once you have found the **Trigger** you wish to define, double click that row to edit the **Trigger** text. The **Trigger** definition is then the first line in the edit box.

The **Trigger** must be marked active for *JetStream Server* to process the **Action Commands** that follow it.

Schedule Trigger Format

Schedule Event Trigger

Sets a Trigger to occur at a particular date and time, with a recurring schedule.

Keyword	Keyword	Keyword	Date	Keyword	Time	Keyword	Value
trigger	schedule	date	*DD MONTH YYYY*	time	*HH:MM:SS*	repeat	# minutes | # hour | # day
trigger	schedule	date	*DD MONTH YYYY*	time	*HH:MM:SS*	repeat	*DAY (see below)*

- Any action commands can follow in the scheduled event, just like a **General Trigger**.
- The **Schedule Event Trigger** can be repeated at a desired time interval by using the keyword `REPEAT`.
- The time interval for the repeat can be specified by using a number followed by the keywords `MINUTES`, `HOURS`, or `DAYS`.
- The **Schedule Event Trigger** may also be repeated at the given time on any day of the week by using the keywords `SUNDAY`, `MONDAY`, `TUESDAY`, `WEDNESDAY`, `THURSDAY`, `FRIDAY`, or `SATURDAY`.
- The **Triggers** will begin responding from the date specified by the `DATE` keyword if it is used.

The following are examples of the **Schedule Event Trigger** for a specific date and time only:

```
trigger schedule date 15 JUNE 2008 time 15:30:00

trigger schedule date 22 SEPTEMBER 2008 time 09:00:00
```

The following are examples of the **Schedule Event Trigger** with repeat intervals:

```
trigger schedule date 15 JUNE 2008 time 15:30:00 repeat 10 minutes

trigger schedule date 22 SEPTEMBER 2008 time 09:00:00 repeat 2 hour

trigger schedule date 01 MAY 2008 time 22:30:00 repeat 1 day

trigger schedule date 09 AUGUST 2008 time 15:00:00 repeat SUNDAY

trigger schedule date 01 MAY 2008 time 08:00:00 repeat MONDAY FRIDAY
```

11 Procedures

Introduction

Procedures are groups of commands that are used like macros in word processing or subroutines in programming. Instead of repeating the same sets of commands in many different general triggers, the commands can be placed in a procedure. This procedure can then be called by a simple `CALL ProcedureName` command in many different **Triggers**. **Procedures** can also call other **Procedures**, allowing for even further functional grouping of commands.

Another advantage of **Procedures** is that the set of commands can be changed in one place in a **Procedure** instead of changing it in many different **Triggers**. When the commands in a **Procedure** are completed, the **Trigger** that called the **Procedure** will execute the next command after the `CALL ProcedureName` command.

TIP: It is good programming practice to use **Procedures** for code centralization and re-use. This reduces the number of lines of code, and makes future debugging and changes easier.

General Rules

- The keyword `PROCEDURE` must be the first word on the first line.
- The name of the **Procedure** should follow after the `PROCEDURE` keyword.
- The **Procedure** name cannot contain any spaces.
- The first line can't be a comment line.
- A **Procedure** need not contain **Action Commands** – it can be a definition only.

Declaring Procedures

A **Procedure** is defined by placing the appropriate **Procedure** command on the first line of an available **Procedure** in the ***Procedure List*** page. Once you have found the **Procedure** you wish to define, double click that row to edit the **Procedure** text. The **Procedure** definition is then the first line in the edit box.

The **Procedure** must be marked active for *JetStream Server* to process the **Action Commands** that follow it.

Procedure Format

Procedure

Used to group a set of commands together to be called in one unit from other Triggers or Procedures.

Keyword	Name
procedures	*ProcedureName (no spaces allowed)*

- See General Rules on previous page.

Writing Procedures

All of the commands and statements that can be used in **General Triggers** can be used in **Procedures**. **Procedures** look very much like **General Triggers** with the exception of the first line which must contain the required keyword `PROCEDURE` followed by a user defined unique `ProcedureName`.

The following is an example of a procedure that is used to turn off all the faders of a 6 channel console on **Audio Engine** 1 **Surface** 2:

```
procedure AE1Surface2AllFadersOff

     cmd ae1 surface2 channel1 bus0 off
     cmd ae1 surface2 channel2 bus0 off
     cmd ae1 surface2 channel3 bus0 off
     cmd ae1 surface2 channel4 bus0 off
     cmd ae1 surface2 channel5 bus0 off
     cmd ae1 surface2 channel6 bus0 off
```

If there were similar procedures for other surfaces, the following procedure could be used to turn off all the faders on all the surfaces:

```
procedure AE1AllSurfacesAllFadersOff

     call AE1Surface1AllOff
     call AE1Surface2AllOff
     call AE1Surface3AllOff
```

Using Procedures

The following is an example of a **Trigger** that needs to turn all faders on a given **Surface** off when a particular **Bridge Button** on the **Surface** is pressed. Other commands can occur before and after the **Procedure** call:

```
trigger ae1 surface2 bridge button12 on

     cmd ae1 surface2 bridge lamp12 on
     call AE1Surface2AllFadersOff
     cmd ae1 surface2 route s[CD 2] to d[Port 2 Fader 6]
```

Following is an example of a **Trigger** that needs to turn all faders on all **Surfaces** off when a particular **Bridge Button** on the **Surface** is pressed:

```
trigger ae1 surface2 bridge button 8 on

     cmd ae1 surface2 bridge lamp8 on
     call AE1AllSurfacesAllFadersOff
     cmd ae1 surface2 route s[CD 2] to d[Port 2 Fader 6]
     cmd ae1 surface2 route s[ISDN 3] to d[Port 2 Fader 1]
```

12 Action Commands

Introduction

Action Commands are any of the actions or events that an **Audio Engine** or *JetStream Server* is capable of performing. Each **Action Command** must start with the keyword `CMD` and must be completed on one line. **Action Commands** can be used in both **Triggers** and **Procedures**. The **Action Commands** are the events that are to happen when the **Trigger** fires or a **Procedure** is called.

Each of the listed commands includes a reference table listing the available options and usages of the command. Generally, alternative keywords are shown, however there are some equivalent keywords that are not shown.

The following keywords are equivalent and can be used interchangeably:

Keyword	**Alternative(s)**	
bus	relay	
delay	wait	timeout
exit	quit	
fader	chan	channel
lamp	light	
surf	surface	

For maximum readability, we suggest using the same keyword consistently throughout your **Trigger Table**. In some situations, you might want to use an alternative word for clarity, e.g. using relay instead of bus when pulsing or changing the state of a relay.

You will also note some command descriptions have different order of words. In most cases the order of keywords is not important – we suggest using the recommended order as per examples.

➥ *See Appendix A for a full list of allowed keywords and their equivalents.*

Commands that are directed at a **Source** and/or **Destination Device** number can be written using the bracket notation `s[Source Device]` and `d[Destination Device]`. The use of this notation is strongly encouraged as it is easier to read and allows more flexibility for future changes. "Hard coded" **Device** numbers can make it more difficult to manage your **Trigger Table** in the future.

➥ *See Chapter 6 for more information on the bracket notation for Device numbers.*

On, Off & Flash Commands

This group of commands turns channels, faders, devices, and lamps on or off.

Bus (Channel, Fader, Relay) On/Off

Turns the bus of a specified channel, fader or device on or off. Can be used to set bus assignments.

Keyword	Engine	Surface	Device	Bus	State
cmd	ae#	*Not Used*	device#### *(dest)*	bus# | relay#	on | off
cmd	ae#	surf# | surface#	device#### *(source)*	bus#	on | off
cmd	ae#	surf# | surface#	fader#	bus#	on | off
cmd	ae#	surf# | surface#	chan# | channel#	bus#	on | off
cmd	ae#	*Not Used*	d[*DeviceName*]	bus# | relay#	on | off
cmd	ae#	*Not Used*	d[LastRoute]	bus#	on | off
cmd	ae#	surf# | surface#	s[*DeviceName*]	bus#	on | off

- This command must include the **Audio Engine** number, **Surface** number, **Device** or **Fader** number, **Bus** number and the required keyword `ON` or `OFF`.
- One of the keywords `CHANNEL`, `FADER` or `DEVICE` is required.
- The required **Device** or **Fader** number may be given using the `s[Source Device]` or `d[Destination Device]` notation instead of using the `DEVICE` keyword.
- The special system destination `d[LastRoute]` will use the **Destination Device** number of the last **Route** command that was executed on the specified **Audio Engine**.
- The keywords `BUS` and `RELAY` can be used interchangeably, however for readability `RELAY` should only be used when the command is directed at an actual relay.
- If the **Device Number** is a **Source Device** or a named `s[Source Device]`, the command will scan faders on the specified surface from left to right until the **Source Device** is found. The specified **Bus** for that fader is then turned on or off. If the **Source Device** appears more than once on the **Surface**, only the first matching fader will be changed.

The following are examples of this command with a **Source Device**:

```
cmd ae1 surface1 device 0103 bus0 on
cmd ae3 surface1 s[CD 1] bus2 off
```

The following are examples of this command using a **Destination Device**:

```
cmd ae1 device 000b bus0 on
cmd ae3 surface1 fader2 bus2 on
cmd ae4 surface2 channel3 bus1 on
cmd ae1 d[Port2 Fader 4 In] bus0 off
cmd ae1 d[LastRoute] bus3 off
```

Button (Bridge or Softkey) On/Off

Turns a Bridge or Softkey button (not lamp) on or off.

Keyword	Engine	Surface	Device	Bus	State
cmd	ae#	surf# | surface#	bridge | softkey	button# | butt#	on | off

- The **Bridge** or **Softkey** button on a given **Surface** may be turned on or off.
- This command applies to the button itself, not to the lamp in the button.
- This command must include the **Audio Engine** number, **Surface** number and the required keyword `BRIDGE` or `SOFTKEY`, and `BUTTON`.
- The keyword `LAMP` must NOT be used.
- One of the keywords `ON` or `OFF` is also required.

The following are examples of the **Bridge** or **Softkey** button commands:

```
cmd ae1 surface1 bridge button11 on
cmd ae3 surface1 softkey button2 off
```

TIP: You can also set a Bridge or Softkey Button on or off using its Device Number and Bus Number, however it is much easier to use the notation above.

Lamp On/Off

Turns a bridge or softkey lamp on or off.

Keyword	Engine	Surface	Device	Bus	State
cmd	ae#	surf# | surface#	bridge | softkey	lamp#	on | off
cmd	ae#	surf# | surface#	channel #	bus#	on | off
cmd	ae#	*Not Used*	device####	bus#	on | off
cmd	ae#	*Not Used*	d[*Device Name*]	bus#	on | off

- The **Lamp** in a specified **Bridge Button**, **Softkey Button** or other **Destination Device** may be turned on or off.
- This command applies to the lamp in the button, not the button itself.
- This command must include the **Audio Engine** number, **Surface** number, and the required keyword `LAMP` or `BUS`.
- The keyword `BUTTON` must NOT be used.
- The keyword `BRIDGE` or `SOFTKEY` must be used for a **Bridge Lamp** or a **Softkey Lamp**.
- The **Destination Device** can be specified using keyword `DEVICE` or using the `d[Destination Device]` notation.
- When using a **Device** name/number or **Channel** number, `BUS` must used instead of `LAMP`.
- One of the keywords `ON` or `OFF` is also required.
- Do not use the keyword `PULSE` with regard to lamp commands.

The following are examples of the **Lamp** commands:

```
cmd ae1 surface1 bridge lamp11 on
cmd ae3 surface1 softkey lamp2 off
```

➥ *See the Relay Pulse command on page 70 to pulse a relay.*

Lamp Flash

Flashes a lamp with variable rate.

Keyword	Engine	Surface	Device	Bus	State	Flash Time	Rate
cmd	ae#	surf# | surface#	bridge | softkey	lamp#	flash	continuous | # times	slow | medium | fast
cmd	ae#	surf# | surface#	channel #	bus#	flash	continuous | # times	slow | medium | fast
cmd	ae#	*Not Used*	device####	bus#	flash	continuous | # times	slow | medium | fast
cmd	ae#	*Not Used*	d[*Device Name*]	bus#	flash	continuous | # times	

- This command applies to the lamp in the button, not the button itself.
- This command must include the **Audio Engine** number, **Surface** number, and the required keyword `LAMP` or `BUS`.
- The keyword `FLASH` is required. The keyword `BUTTON` must NOT be used.
- The keyword `BRIDGE` or `SOFTKEY` must be used for a **Bridge Lamp** or a **Softkey Lamp**.
- The **Destination Device** can be specified using keyword `DEVICE` or using the `d[Destination Device]` notation.
- When using a **Device** name/number or **Channel** number, `BUS` must used instead of `LAMP`.
- The number of times followed by the optional keyword `TIMES` is used to specify how long the lamp flashes for. The timeframe depends on flash speed.
- The keyword continuous is assumed if the flash time is not specified.
- On *Artisan*, *Mosaic* and *COM12* hardware, a flash time of `SLOW`, `MEDIUM` or `FAST` can be used.
- If the rate is not specified, `MEDIUM` is assumed. This is the rate supported by older hardware.
- `SLOW` flash is on for 1 second then off for 1 second
- `MEDIUM` flash is on for ½ second then off for ½ second
- `FAST` flash is on for ¼ second then off for ¼ second
- To flash for a certain amount of time, use this formula (based on flash speed and number)
  *(seconds on + seconds off) * No of Flashes*
- The keyword `FLAG` is no longer required, but is retained for compatibility.
- The keyword `CONTINUOUS` is no longer required, but is retained for compatibility.
- Do not use the keyword `PULSE` with regard to lamp commands.

The following are examples of the **Lamp** commands:

```
cmd ae4 surface2 bridge lamp2 flash once          ~ old format
cmd ae2 surf1 chan13 bus33 flash once 5           ~ old format
cmd ae1 device27 bus1 flash continuous 3 times    ~ old format
cmd ae1 device27 bus1 flash 3 times               ~ new format
cmd ae1 d[Ctrl – Surf1 GPI out] bus43 flash fast  ~ new format
cmd ae1 surf1 chan13 bus44 flash medium           ~ new format
```

Fader Commands

This group of commands is used to manipulate faders, including routes, levels, trim, mode, alias and effects.

Input Route

Assigns a specified Source Device to a Destination Device. Same as assigning input on surface.

Keyword	Engine	Surface	Keyword	Source	Keyword	Destination
cmd	ae#	surf# | surface#	route	device ####	{to}	chan# | channel # | fader #
cmd	ae#	*Not Used*	route	s[*Device Name*]	{to}	d[*Device Name*]

- This command must include the **Audio Engine** number, **Surface** number, **Source Device** number, **Destination Device** number and the required keyword `ROUTE`.
- The **Source Device** can be specified using keyword `DEVICE` or using the `s[Source Device]` notation.
- The **Destination Device** can be specified by using the keywords FADER or CHANNEL or using the `d[Destination Device]` notation.
- The use of the `TO` keyword is optional.
- The last routed **Destination Device** number is automatically stored in a special system **Destination Device** that can be referenced with the `d[LastRoute]` notation.

The following are examples of the **Route** command:

```
cmd ae1 surface1 route device 0103 to channel2
cmd ae3 route s[Port1 Program Out] to d[Port1 Fader7 In]
```

Set Device Alias

Assigns an 8 character alias to a Source Device, for display on Surfaces.

Keyword	Engine	Source Device	Keyword	Alias
cmd	ae#	device ####	{set} alias	" " *(8 characters)*
cmd	ae#	s[*Device Name*]	{set} alias	" " *(8 characters)*

- Each **Source Device** can have a user defined Alias name assigned to it. For example, the source (OB/Remote venue) of an ISDN line could be assigned as an alias name.
- **Alias** names are automatically displayed on all **Surfaces** in the label position above all **Faders** to which the **Device** is assigned.
- The displayed alias is changed anytime a **Route** is changed or the **Alias** itself is changed.
- **Alias** names are limited to a maximum of eight characters.
- This command must include the **Audio Engine** number, **Source Device** number, and the required keyword `ALIAS`.
- The keyword `SET` is optional and can be used for improved code readability.
- The **Source Device** can be specified using `DEVICE` or `s[Source Device]` notation.

The following are examples of the **Set Alias** command with 8 character strings:

```
cmd ae1 device 0103 set alias " Game 1 "
cmd ae3 s[ISDN 1] set alias " Ready  "
```

🎧 TIP: The Device Alias is an 8 character string on both 8 & 16 character **Audio Engines**.

Set Fader Level

Sets the Fader level of a specified fader or Destination Device. Same as moving fader on surface.

Keyword	Engine	Surface	Destination	Keyword	Value
cmd	ae#	surf# | surface#	chan# | channel#	{set} level	0 to 255 | -INFINITY to +10dB | OFF | INF
cmd	ae#	surf# | surface#	fader#	{set} level	0 to 255 | -INFINITY to +10dB | OFF | INF
cmd	ae#	*Not Used*	d[*Device Name*]	{set} level	0 to 255 | -INFINITY to +10dB | OFF | INF
cmd	ae#	*Not Used*	device ####	{set} level	0 to 255 | -INFINITY to +10dB | OFF | INF

- This command must include the **Audio Engine** number, **Surface** number, the fader or destination device number and the required keyword `LEVEL`.
- The keyword `SET` is optional and can be used for improved code readability.
- The **Destination Device** can be specified by using the keywords `FADER` or `CHANNEL`, or by using the `d[Destination Device]` notation.
- The **Fader Level** value may be expressed in dB or in internal **Audio Engine** units (0-255).
- A value of 195 is equal to unity gain, or 0 dB.
- The keyword `dB` is required for levels expressed in dB units.
- The allowable range of levels in dB is `-INFINITY` (off) to `+10 dB`.
- The keywords `INF` or `OFF` may be used to set the fader level to infinity.

The following are examples of the **Set Fader Level** command:

```
cmd ae1 surface1 set fader2 level 195
cmd ae2 surface2 channel6 level -10 db
cmd ae3 set d[Port1 Fader4 In] level 0 db
cmd ae1 d[Port1 Fader2 In] level off
```

Set Fader Mode

Sets the Fader mode of a specified fader or Destination Device. Same as setting mode on surface.

Keyword	Engine	Surface	Destination	Keyword	Value
cmd	ae#	surf# | surface#	chan# | channel#	{set} mode	STEREO | MONO | PHASE | LL | RR | Lx | xR
cmd	ae#	surf# | surface#	fader#	{set} mode	STEREO | MONO | PHASE | LL | RR | Lx | xR
cmd	ae#	*Not Used*	d[*Device Name*]	{set} mode	STEREO | MONO | PHASE | LL | RR | Lx | xR
cmd	ae#	*Not Used*	device ####	{set} mode	STEREO | MONO | PHASE | LL | RR | Lx | xR

- This command must include the **Audio Engine** number, **Surface** number, the **Fader** or **Destination Device** number and the required keyword `MODE`.
- The keyword `SET` is optional and can be used for improved code readability.
- The **Destination Device** can be specified by using the keywords `FADER` or `CHANNEL` or by using the `d[Destination Device]` notation.

The **Fader Mode** may be set to the following values:

STEREO	For stereo source, Left source to Left, Right source to Right For mono source, Left Right source average to Left and Right
MONO	Left Right source average to Left and Right
PHASE	Left source to Left, minus Right source to Right
LL	Left source to Left and Right
RR	Right source to Left and Right
Lx	Left source on Left, silence on Right
xR	Silence on Left, Right source on Right

The following are examples of the **Set Fader Mode** command:

```
cmd ae1 surface1 set fader2 mode mono
cmd ae2 surface2 channel6 mode stereo
cmd ae3 set d[Port1 Fader4 In] mode LL
cmd ae1 d[Port1 Fader2 In] mode xR
```

Set Fader Trim

Sets the Fader mode of a specified fader or Destination Device. Same as setting fader trim on surface.

Keyword	Engine	Surface	Destination	Keyword	Value
cmd	ae#	surf# | surface#	chan# | channel#	{set} trim	-10 to +10 (*dB*) | none
cmd	ae#	surf# | surface#	fader#	{set} trim	-10 to +10 (*dB*) | none
cmd	ae#	*Not Used*	d[*Device Name*]	{set} trim	-10 to +10 (*dB*) | none
cmd	ae#	*Not Used*	device ####	{set} trim	-10 to +10 (*dB*) | none

- This command must include the **Audio Engine** number, **Surface** number, the **Fader** or **Destination Device** number and the required keyword `TRIM`.
- The keyword `SET` is optional and can be used for improved code readability.
- The **Destination Device** can be specified by using the keywords `FADER` or `CHANNEL` or by using the `d[Destination Device]` notation.
- The allowable range of trim values is from -10 dB to +10 dB where 0 indicates 0 dB.
- The plus sign and the keyword `dB` are optional.
- The keyword `NONE` may be used to set 0 dB trim.

The following are examples of the **Set Fader Trim** command:

```
cmd ae1 surface1 channel2 set trim 10 db
cmd ae3 d[Port2 Fader6 In] trim 0
cmd ae2 surface2 fader7 set trim -3
cmd ae3 d[Port3 Fader1 In] trim none
```

Set Fader Pan

Sets the pan value of a specified fader or Destination Device. Same as setting fader pan on surface.

Keyword	Engine	Surface	Destination	Keyword	Value
cmd	ae#	surf# | surface#	chan# | channel#	{set} pan	-20 to +20 | center | left |right
cmd	ae#	surf# | surface#	fader#	{set} pan	-20 to +20 | center | left |right
cmd	ae#	*Not Used*	d[*Device Name*]	{set} pan	-20 to +20 | center | left |right
cmd	ae#	*Not Used*	device ####	{set} pan	-20 to +20 | center | left |right

- This command must include the **Audio Engine** number, **Surface** number, the **Fader** or **Destination Device** number and the required keyword `PAN`.
- The keyword `SET` is optional and can be used for improved code readability.
- The **Destination Device** can be specified by using the keywords `FADER` or `CHANNEL` or by using the `d[Destination Device]` notation.
- The allowable range of pan values is from -20 for full pan left to +20 for full right pan.
- A pan value of 0 indicates the channel center.
- The plus sign is optional.
- The keyword `LEFT` may be used for full pan left values and the keyword `RIGHT` may be used for full pan right values.
- The keyword `CENTER` may be used to set the pan value to the center of the channel.

The following are examples of the **Set Fader Pan** command:

```
cmd ae1 surface1 channel 2 set pan -20
cmd ae3 d[Port2 Fader6 In] pan  center
cmd ae2 surface2 fader 7 pan right 18
cmd ae3 d[Port 3 Fader 1 In] set pan 0
cmd ae1 d[Port 1 Fader 1 In] pan left 10
```

Set Fader Limiter

Sets the limiter parameters of a specified fader or Destination Device. Same as surface controls.

Keyword	Engine	Surface	Destination	Keyword	Value
cmd	ae#	surf# | surface#	chan# | channel#	{set} limiter	threshold # ratio # release #
cmd	ae#	surf# | surface#	fader#	{set} limiter	threshold # ratio # release #
cmd	ae#	*Not Used*	d[*Device Name*]	{set} limiter	threshold # ratio # release #
cmd	ae#	*Not Used*	device ####	{set} limiter	threshold # ratio # release #

- This command must include the **Audio Engine** number, **Surface** number, the **Fader** or **Destination Device** number and the required keyword `LIMITER`.
- The keyword `SET` is optional and can be used for improved code readability.
- The **Destination Device** can be specified by using the keywords `FADER` or `CHANNEL`, or by using the `d[Destination Device]` notation.
- This command can change one or more of the limiter parameters values by using the appropriate keywords `THRESHOLD`, `RATIO`, or `RELEASE`.
- Any parameters omitted are not changed.
- The attack value of the limiter is fixed at 100 microseconds and can not be changed.

The following are examples of the Set Fader Limiter command:

```
cmd ae1 surface1 channel2 set limiter threshold 8 ratio 3 release 20
cmd ae3 d[Port2 Fader6 In] set limiter threshold -8 ratio 4 release 30
cmd ae3 surface2 fader2 set limiter ratio 6
```

🎧 TIP: To programmatically turn the limiter/compressor (dynamics) on or off, send the appropriate bus on/off command to BUS22 of the appropriate destination. There is no separate Command Builder command to achieve this.

Set Fader Compression

Sets the compressor parameters of a specified fader or Destination Device. Same as surface controls.

Keyword	Engine	Surface	Destination	Keyword	Value
cmd	ae#	surf# | surface#	chan# | channel#	{set} compression	threshold # ratio # attack # release #
cmd	ae#	surf# | surface#	fader#	{set} compression	threshold # ratio # attack # release #
cmd	ae#	*Not Used*	d[*Device Name*]	{set} compression	threshold # ratio # attack # release #
cmd	ae#	*Not Used*	device ####	{set} compression	threshold # ratio # attack # release #

- This command must include the **Audio Engine** number, **Surface** number, the **Fader** or **Destination Device** number and the required keyword `COMPRESSION`.
- The keyword `SET` is optional and can be used for improved code readability.
- The **Destination Device** can be specified by using the keywords `FADER` or `CHANNEL`, or by using the `d[Destination Device]` notation.
- This command can change one or more of the compressor parameters by using the appropriate keywords `THRESHOLD`, `RATIO`, `ATTACK` or `RELEASE`.
- Any parameters omitted are not changed.

The following are examples of the **Set Fader Compression** command:

```
cmd ae1 surface1 channel2 set compression threshold 8 attack 10
cmd ae2 d[Port1 Fader6 In] set compression threshold 10 ratio 4
cmd ae3 surface2 fader2 set compression ratio 6 release 1500
```

🎧 TIP: To programmatically turn the limiter/compressor (dynamics) on or off, send the appropriate bus on/off command to BUS22 of the appropriate destination. There is no separate Command Builder command to achieve this.

Set Fader Equalization

Sets the equalization parameters of a specified fader or Destination Device. Same as surface controls.

Keyword	Engine	Surface	Destination	Keyword	Value
cmd	ae#	surf# | surface#	chan# | channel#	{set} eq	high | high mid | low mid | low freq # gain {db} # bandwidth #
cmd	ae#	surf# | surface#	fader#	{set} eq	*As above*
cmd	ae#	*Not Used*	d[*Device Name*]	{set} eq	*As above*
cmd	ae#	*Not Used*	device ####	{set} eq	*As above*

- This command must include the **Audio Engine number**, **Surface** number, the **Fader** or **Destination Device** number and the required keyword `EQ`.
- The keyword `SET` is optional and can be used for improved code readability.
- The **Destination Device** can be specified by using the keywords `FADER` or `CHANNEL`, or by using the `d[Destination Device]` notation.
- The equalization can be set in one of four bands that are designated using the keywords `HIGH`, `HIGH MID`, `LOW MID` and `LOW`.
- The frequency and gain can be set in any of the four bands using the keywords `FREQ` and `GAIN`.
- The bandwidth can be set only in the High Mid and Low Mid bands using the keyword `BANDWIDTH`.
- Gain values are assumed to be in dB.
- The `dB` keyword is optional.

The following are examples of the Set Fader Equalization command:

```
cmd ae1 surface2 fader1 set eq high freq 17000 gain 12
cmd ae1 surface1 channel2 set eq high mid freq 1000 gain -10
cmd ae3 d[Port2 Fader6 In] set eq low freq 1400 gain 10
cmd ae3 surface2 fader2 set eq low mid freq 2000 gain 12 bandwidth 1000
```

TIP: To programmatically turn the equalization on or off, send the appropriate bus on/off command to BUS23 of the appropriate destination. There is no separate Command Builder command to achieve this.

Set Mix Minus

Sets the output bus for a Mix Minus on a surface. These settings are not available on the surface.

Keyword	Engine	Surface	Keyword	Destination	Values
cmd	ae#	surf# | surface#	set mix minus | set mix-	#	bus # | {always} {addmic} {stereo}

This command overrides the default operation of the mix minus system for a particular engine and surface. The default mix minus settings are entered on the system page of the *AEConfig* program.

- This command must include the **Audio Engine** number, **Surface** number, keywords `SET MIX MINUS` (or `SET MIX-`) and the desired **Mix Minus** output number.
- The keyword `BUS` is used to change the mix bus used as the source of the mix minus.
- If the keyword `BUS` is omitted, the feature keywords `ALWAYS`, `ADDMIC` & `STEREO` are used.
- The **Set Mix Minus** command sets all three features if the `BUS` keyword is not used. If the keyword is present, the feature is turned on. If the keyword is absent, the feature is turned off.
- The keyword `ALWAYS` causes the **Mix Minus** output to be active, even when the Source Devices allocated to it are not in use. If `ALWAYS` is omitted, the output is silent unless an appropriate **Fader** is turned on.
- The keyword `ADDMIC` routes the talkback microphone to the **Mix Minus** output when the **Fader** allocated to the **Mix Minus** output is turned off.
- The keyword `STEREO` switches the mix minus output to stereo (if your configs support it).

The **Bus** number may be set to the following values:

BUS1 PGM mix bus
BUS2 CUE mix bus
BUS3 AUX1 mix bus
BUS4 AUX2 mix bus
BUS5 AUX3 mix bus
BUS6 AUX4 mix bus (available on Surface 1 only)
BUS9 AUX5 mix bus (available on Surface 1 only)
BUS10 AUX6 mix bus (available on Surface 1 only)
BUS11 AUX7 mix bus (available on Surface 1 only)
BUS12 AUX8 mix bus (available on Surface 1 only)

The following are examples of the **Set Mix Minus** command:

```
cmd ae1 surface1 set mix minus 3 bus 1
cmd ae2 surface3 set mix minus 4
cmd ae2 surface2 set mix minus 4 always
cmd ae4 surface1 set mix minus 2 addmic
cmd ae2 surface1 set mix minus 1 stereo
cmd ae1 surface1 set mix minus 3 always addmic stereo
```

External Communications

JetStream Server has the capability to act as a serial data router, as well as control external devices through serial and TCP/IP commands. *CommandBuilder* uses the following commands to control serial routing and send commands via serial, IP and GPI to external devices.

Text to UDP Port

Sends an ASCII text string to a specified UDP destination port, from the local port set in JetStream Server.

Keyword	Keyword	Destination	Keyword	Port	Data
cmd	udp	ip[0.0.0.0]	port	#	*"Data enclosed in Quote Marks"*
cmd	udp	u[ComputerName]	port	#	*"Data enclosed in Quote Marks"*

- The required keyword is `UDP`.
- The destination IP address must be specified in the command using the `u[UserName]` notation or `ip[100.100.1.1]` address bracket notation.
- The destination port number is specified using the required keyword `PORT`.
- The data to send must be enclosed in quotes.

The following are examples of the **Text to UDP Port** command:

```
cmd udp ip[192.100.67.1] port 10300 "Text to 192.100.67.1 Port 10300"
cmd udp u[PC-12345] port 10600 "Hello World"
```

Text to Com Port

Sends an ASCII text string out the specified Auxiliary Com Port.

Keyword	Keyword	Dest Port	Data
cmd	port	#	*"Data enclosed in Quote marks / Hex Bytes within <> marks"*

- The **Com Port** is assumed to have been configured and opened in *JetStream Server*.
- The required keyword is `PORT`.
- The destination port must be included. This is the **Com Port** number assigned in Windows.
- The data to send should be enclosed in quotes.
- <00> Numbers enclosed in < > are treated as hex bytes.
- Common ASCII code shortcuts can also be used, as listed below.

Hex	ASCII Code
<esc>	27
<nul>	0
<soh>	1
<stx>	2
<sub>	26
<eot>	4
<lf>	10
<cr>	13
<1c>	28

The following are examples of the **Text to Com Port** command:

```
cmd port 22 "This is Text to Port 22"
cmd port 16 "Hello World"
```

This command can be used to send text to a BetaBrite scrolling LED panel. This has been used at a number of **Logitek** sites around the world. The following example will send text to a BetaBrite sign:

```
cmd port 3 "<nul><nul><nul><eot><1c><2d>  G'Day Fellas"
```

➥ *For more information on the BetaBrite signs, see http://www.betabrite.com/ or see Chapter 23 for more detailed information.*

Relay Pulse

Used to pulse a relay on an Audio Engine or Control Surface.

Keyword	Engine	Destination	Bus	Keyword
cmd	ae#	*device####*	relay # | bus#	pulse
cmd	ae#	d[*Device Name*]	relay # | bus#	pulse

- This command must include the **Audio Engine** number, the **Destination Device** number and requires keywords `RELAY`, and `PULSE`.
- The **Destination Device** can be specified using the `d[Destination Device]` notation.
- The **Relay** number is equivalent to the **Bus** number. The two keywords can be interchanged.
- This command will also accept the **Destination** in the **Surface** and **Channel**/**Fader** notation, however for improved readability this usage is not recommended.

The following are examples of the **Relay Pulse** command:

```
cmd ae1 device0001 relay5 pulse
cmd ae1 d[Ctrl – Surf1 GPI out] relay12 pulse
```

➥ *See the On/Off command on Page 55 to turn a relay on or off instead of pulse it.*

Assembly Command

Used to send Audio Engine Protocol command messages to an Audio Engine (for advanced users).

Keyword	Engine	Keyword	Data
cmd	ae#	*asm*	*"02 xx xx xx xx"*

- This command sends a command to an **Audio Engine** using the **Audio Engine Protocol**.
- This command must include the **Audio Engine** number and the required keyword `ASM`.
- The command should follow as a hex code string with one space between each hex byte.
- See the **Audio Engine Protocol** or contact **Logitek** for more information on hex commands.

The following is an example of the **Assembly** command:

```
cmd ae1 asm "02 03 b2 0b 03"
```

Other Functions

This section covers some useful commands not listed in previous sections.

Talk Delay

Controls the Audio Engine Talk Delay when using SharcAttack DSP cards.

Keyword	Engine	Surface	Destination	Keyword	Command
cmd	ae#	surf# | surface#	chan# | channel#	talk	start | stop | off | dump | max #
cmd	ae#	*Not Used*	d[*Device Name*]	talk	start | stop | off | dump | max #
cmd	ae#	*Not Used*	device ####	talk	start | stop | off | dump | max #

- This command must include the **Audio Engine** number, **Surface** number, **Destination Device**, and required keyword `TALK`.
- One of the following commands is required, `START`, `STOP`, `OFF`, `DUMP` or `MAX` seconds.
- The value following `MAX` must be between 0 and 25.5 seconds, in increments of 0.1 seconds. An error will occur if the value is omitted or outside this range.
- The **Destination Device** can be specified by using the keyword `CHANNEL`, or by using the `d[Destination Device]` notation.
- The **Destination Device** is usually `d[Port1 Router 1 In]` or `d[Port2 Router 1 In]`, but some configurations are set for multiple delays on Port 1 and Port 2.
- The **Surface** display for the **Talk Delay** is linked to the Router 1 Crosspoint on Port 1 & 2.

The following commands can be sent to the **Talk Delay**:

START Begins buildup of delay, and turns on the Surface display.
STOP Ramps down to zero delay time.
OFF Immediately turns off delay, dumps any remaining delay and turns off the Surface display.
DUMP Dumps 4 seconds from the delay buffer.
MAX 0.0 Sets the maximum delay time, from 0 to 25.5 seconds in 0.1 second increments. The DSP card may impose a shorter delay maximum depending on number of delays used and available memory.

The following are examples of the **Talk Delay** command:

```
cmd ae1 d[Port1 Router 1 In] talk start
cmd ae1 d[Port1 Router 1 In] talk stop
cmd ae1 d[Port1 Router 1 In] talk off
cmd ae1 d[Port2 Router 1 In] talk dump
cmd ae1 d[Port1 Router 2 In] talk max 9.9
```

Execution Control

These statements are used for controlling the execution of **Triggers** and **Procedures**.

Quit/Exit

Stops JetStream Server from executing any further commands in the current trigger.

Keyword	Keyword
cmd	quit | exit

- The keyword `QUIT` or `EXIT` is required.
- This command is useful in **Triggers** that contain conditional test statements where a value has been changed earlier in the **Trigger**.

The following are examples of the **Quit** and **Exit** commands:

```
cmd quit
cmd exit
```

Call Procedure

Calls a Procedure from within a Trigger or another Procedure.

Keyword	Procedure
cmd	*ProcedureName*

- This command requires the keyword `CALL` followed by the name of the **Procedure**.
- All commands in the **Procedure** are executed before control returns to the calling code.
- Any **Procedure** can call other **Procedures** within itself.

The following is an example of the **Call Procedure** command:

```
call Surface1AllFadersOff
```

➥ *See Chapter 11 for more information on writing Procedures.*

Set Trigger Active/Not Active

Controls whether an existing Trigger will be executed by JetStream Server.

Keyword	Keyword	Keyword	Keyword	Trigger
cmd	set	trigger	active | notactive	*TriggerDefinition* | *(TriggerDefinition)*

- When a **General Trigger** has been set not active, it will not execute the specified action commands when the **Trigger** event occurs. It can be enabled again by using the **Set Trigger Active** command.
- The required keywords are `SET`, `TRIGGER`, and either `ACTIVE` or `NOTACTIVE`.
- The definition of the **General Trigger** that is to be set active or not active should follow the keywords.
- The **General Trigger** should exist before using the **Set Trigger Active/Not Active** command.
- The optional parenthesis around the **General Trigger** definition increases user readability.
- If the specified General Trigger that has been set not active is a **Toggle Trigger**, it may still have it's toggle state changed using a **Set Toggle State** command and its **Toggle** state tested using an **If Toggle Equal** statement.

The following are examples of the **Set Trigger Active/Not Active** command:

```
cmd set trigger notactive ae3 d[Port1 Fader 4 In] bus0 off
cmd set trigger notactive ( ae1 surface2 device 0105 bus0 on )
cmd set trigger active ae3 d[Port1 Fader 4 In] bus0 off
cmd set trigger active ( ae1 surface2 device 0105 bus0 on )
```

Set Toggle State

Programmatically sets the state of a Toggle Trigger.

Keyword	Keyword	Keyword	Value	Trigger
cmd	set	toggle =	1 | 2	*TriggerDefinition* | *(TriggerDefinition)*

- The specified **General Trigger** must be a **Toggle Trigger**.
- The allowable **Toggle** states are 1 (normal) or 2 (toggle).
- The required keywords are `TOGGLE`, the equal sign (=) and the desired toggle state (1 or 2).
- The keyword `SET` is optional.
- The specification of the **General Trigger** should follow the keywords.
- The **General Trigger** should exist before the **Set Toggle State** command is written.
- The optional parenthesis around the **General Trigger** increases user readability.
- The current state of a **Toggle Trigger** can be tested using the **If Toggle Equal** statement.

The following are examples of the **Set Toggle State** command:

```
cmd set toggle = 1 ( ae3 d[Port1 Fader 4 In] bus0 on )
cmd set toggle = 2 ( ae1 surface2 bridge button6 on )
```

Cancel Timer

Cancels a Conditional Trigger that was previously started by an If Timer statement.

Keyword	Keyword	Keyword	Timer
cmd	cancel	timer	#

- The required keywords are `CANCEL` and `TIMER` followed by the number of the **Timer** from specification of the If Timer statement.
- It is not an error to cancel a timer that is not running.

The following is an example of the Cancel Timer command:

```
cmd cancel timer 2
```

Clear Button

Clears a Conditional Trigger previously set using the If Button statement.

Keyword	Keyword	Keyword	Trigger
cmd	clear	button	*TriggerDefinition* | *(TriggerDefinition)*

- The required keywords are `CLEAR` and `BUTTON` followed by the definition of the **Conditional Trigger**.
- The optional parenthesis around the **Conditional Trigger** increases user readability.
- It is not an error to clear a **Conditional Trigger** that has not been set.

The following are examples of the **Clear Button** command:

```
cmd clear button ( ae3 surface1 softkey button3 on )
cmd clear button ( ae2 surface2 bridge button6 off )
```

Clear Trigger

Clear a Conditional Trigger previously set using the If Trigger statement.

Keyword	Keyword	Keyword	Trigger
cmd	clear	trigger	*TriggerDefinition* | *(TriggerDefinition)*

- The required keywords are `CLEAR` and `TRIGGER` followed by the definition of the Conditional Trigger.
- The optional parenthesis around the **Conditional Trigger** increases user readability.
- It is not an error to clear a **Conditional Trigger** that has not been set.

The following are examples of the **Clear Trigger** command:

```
cmd clear trigger ( ae3 surface1 d[Port1 Fader 3 In] on )
cmd clear trigger ( ae2 surface2 route device 0102 to channel3 )
```

➥ *See Chapter 9 for more information on Conditional Triggers.*

Clear When Off

Clears any commands that were stored by a previous When Off statement.

Keyword	Keyword	Keyword	Keyword	AE	Surface	Destination
cmd	clear	when	off	ae#	surf# | surface#	chan# | channel#
cmd	clear	when	off	ae#	surf# | surface#	chan#-# | channel#-# *(range)*
cmd	clear	when	off	ae#	surf# | surface#	fader#
cmd	clear	when	off	ae#	surf# | surface#	fader#-# *(range)*
cmd	clear	when	off	ae#	*Not Used*	d[*Device Name*]

- The **When Off** statement stores commands as "Pending" if a fader is on.
- Use **Clear When Off** to cancel a prior **When Off** command, before setting another.
- The required keywords are `CLEAR WHEN OFF`.
- The `CHANNEL` or `FADER` keywords may include a range of channels to be cleared.
- The **Destination Device** can be specified using the `d[Destination Device]` notation.
- The optional parenthesis can be used to enclose the specification of the **When Off** test statement.
- It is not an error to clear a fader that is not "Pending".

The following are examples of the Clear When Off command:

```
cmd clear when off ( ae3 d[Port1 Fader 4 In] )
cmd clear when off ( ae1 surface2 channel 1-6 )
cmd clear when off ( ae2 surface1 fader 6 )
```

➥ *See Chapter 17 for more information about the When Off Test Statement.*

13 Surface Text Commands

Introduction

Logitek Surfaces have the ability to display user generated text on their LCD screens. The type, size and amount of text varies widely between different generations of console. The methods for displaying text are similar, but, due to small differences each method is detailed below in a separate section.

The types of text that can be displayed include user defined text strings, a continuously updated time display and a continuously updated temperature display generated by an external device. The time display is obtained and formatted from the *JetStream Server* PC's current Windows time format. The temperature is formatted in the ***System Administrator ➢ External Devices*** tab in *JetStream Server*.

A number of menu based text display commands is also available for obtaining user selections.

All Surfaces

Event Timer

Most surfaces have one or more event timers that are controlled from dedicated buttons on the monitor or softkey module. Timer components are also available in vMix+ where they can operate independently or as slaves to the surface displays. This command provides control of event timers from triggers and procedures.

Keyword	AE	Keyword	ID	address	Action
cmd	ae#	circletimer	timer#	device # (hex)	Start Stop Reset Up Down

- The default timer number is 1.
- Device 2C = Surface1, Device 54 = Surface 2.

The following are examples of commands for a surface 1 timer**:**

```
cmd ae1 circletimer 1 device 2C start
cmd ae0 circletimer 2 device 2C reset
```

Artisan Screens

The *Artisan* has color LCD screens for each fader, which can be used to show fader information and labels. In addition, the **Meter Bridge**, Monitor module and Wide Softkey module have screens that support user defined text.

All screens on the *Artisan* are color.

The *Artisan* has user screen space similar to a *Mosaic* console. Like the *Mosaic*, the *Artisan* is designed to be used in conjunction with a new software application, *vScreen*, to show additional user defined information (including clocks, meters and text).

➥ *See the vTools Manual for further information on configuring vScreen.*

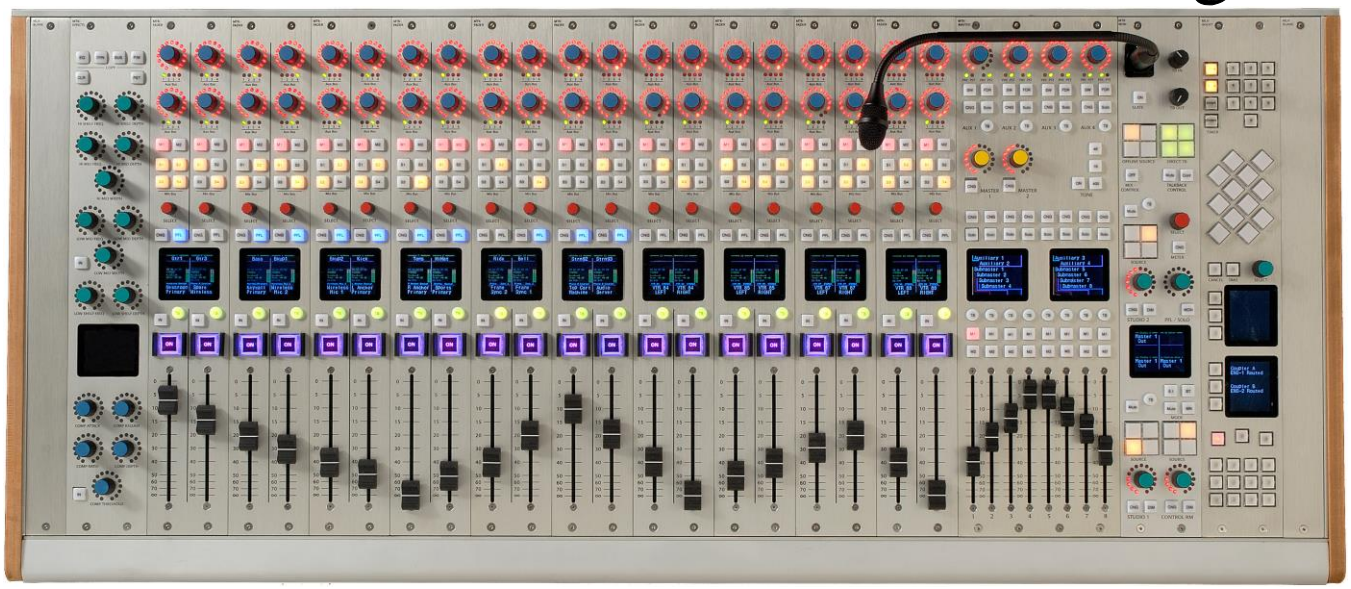


Figure 3 - Artisan Surface

> The *Artisan* introduces many new surface text functions, and therefore some commands requiring destination addresses will be affected by the *Artisan* layout your surfaces use. Please contact **Logitek Electronic Systems** or your reseller to obtain the latest information.

Artisan Fader Screen

The *Artisan* **Fader Module** (MTK-FADER) contains one screen (shared between two faders) that allows an **Alias** to be written. The **Alias** is an 8 character string in large font, as shown below.

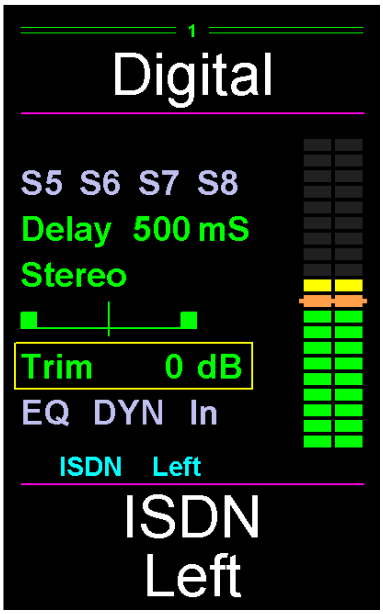


Figure 4 - Fader Screen

Artisan Label Text

This command is used to write a label to an input on a Fader Screens.

Keyword	AE	Surface	Screen	Keyword	Keyword	Text
cmd	ae#	surf# | surface #	chan# | channel#	text	label	"1234567890123456"
cmd	ae#	surf# | surface #	fader#	text	label	"1234567890123456"

- The label is large font text displayed in a label space directly above the fader information.
- This label may have a maximum of 8 characters.
- The command must contain the `CHANNEL` number and the keyword `LABEL`.
- The `LINE` or `POS` keywords are not required for this type of text.
- To clear the label, write a blank string.
- The label may be replaced by flashing text "PENDING" if a `WHEN OFF` command is used.
- The label is replaced if a `SET ALIAS` command is issued to the source device on that fader.
- The label is cleared when a route is performed on that fader.
- To center a label you must manually pad the text string with spaces.

The following are examples of channel label text commands written to the **Fader Screen**:

```
cmd ae1 surface1 channel1 text label "  MUTE  "
cmd ae1 surface1 channel6 text label "        "
```

➥ *To set an alias for an input, see the Set Device Alias command in Chapter 12.*

Artisan Effects Screen

The *Artisan* **Effects Module** (MTK-EFFECTS) contains one screen. This screen does not support user text.

➥ *For further details, see the Artisan Manual.*

Artisan Master Screen

The *Artisan* **Master Module** (MTK-MASTER) contains two screens that allow **Aliases** to be written. The **Alias** is an 8 character string in small font.

Artisan Master Text

This command is used to write a label to the Artisan Master Screen.

Keyword	AE	Surface	Destination	Keyword	Keyword	Text
cmd	ae#	*Not Used*	device#### *(dest)*	set	alias	"12345678"
cmd	ae#	*Not Used*	d[Device Name] *(dest)*	Set	alias	"12345678"

- The label is small font text displayed for each meter directly below the meter.
- This label may have a maximum of 8 characters.
- The command must contain the `SOURCE` number and the keyword `SET ALIAS`.
- The `LINE` or `POS` keywords are not required for this type of text.
- To clear the label, write a blank string.
- To center a label you must manually pad the text string with spaces.

The following are examples of meter text labels written to the **Master Screen**:

```
cmd ae1 s[Sub 1   Pre Fdr Out] set alias " Sub 1"
cmd ae1 s[Aux 1   Pre Fdr Out] set alias " Aux 1"
```

TIP: When the Artisan surface first starts up, it labels the meters for Aux 1-4 and Submasters 1-8. Then, when JetStream Server starts up, it looks to see if there are any aliases for every source on the system, and if there is no alias, it sends out a blank string for every device with an alias. This means that once JetStream Server starts, all of the labels for Aux 1-4 and Submasters 1-8 on the Artisan are overwritten by the blank aliases.

The following example contains the neccessary commands for inclusion in an **Init Trigger** or **Procedure** called from the **Init Trigger** or **Surface Reset Trigger** to ensure that meter text labels are written to the **Master Screen** on an *Artisan* connected to **Audio Engine** 1:

```
cmd ae1 s[Sub 1    Pre Fdr Out] set alias "  Sub 1"
cmd ae1 s[Sub 2    Pre Fdr Out] set alias "  Sub 2"
cmd ae1 s[Sub 3    Pre Fdr Out] set alias "  Sub 3"
cmd ae1 s[Sub 4    Pre Fdr Out] set alias "  Sub 4"
cmd ae1 s[Sub 5    Pre Fdr Out] set alias "  Sub 5"
cmd ae1 s[Sub 6    Pre Fdr Out] set alias "  Sub 6"
cmd ae1 s[Sub 7    Pre Fdr Out] set alias "  Sub 7"
cmd ae1 s[Sub 8    Pre Fdr Out] set alias "  Sub 8"
cmd ae1 s[Aux 1    Pre Fdr Out] set alias "  Aux 1"
cmd ae1 s[Aux 2    Pre Fdr Out] set alias "  Aux 2"
cmd ae1 s[Aux 3    Pre Fdr Out] set alias "  Aux 3"
cmd ae1 s[Aux 4    Pre Fdr Out] set alias "  Aux 4"
```

➥ *For further details, see the Artisan Manual.*

Artisan Monitor Screen

The *Artisan* **Monitor Module** (MTK-MON) has one screen. This screen does not support user text.

➥ *For further details, see the Artisan Manual.*

Artisan Wide Softkey Screen

The *Artisan* **Wide Softkey Module** (MLX-WSOFT) has two screens for user text. Each screen has three buttons to the left, for initiating **Triggers**.

Both screens can display user defined text. In addition they can be used with **Route Select** or **Variable Select Triggers** to present menu or router choices to the user. The options are selected by the user with the ***Select*** knob and ***Cancel*** and ***Take*** buttons.

When using **Route Select**, the selected route can tallied next to the button as shown on the right.

Figure 6 - Route Select tally

Figure 5 - Route Select menu

Artisan Wide Softkey Text

This command is used to write text to the Fader Screens.

Keyword	AE	Surface	Keyword	Screen	Options	Line	Pos	Color	Text
cmd	ae#	surf# | surface #	text	device52	{insert}	line#	pos#	*See below*	"14 chars"
cmd	ae#	surf# | surface #	text	device51	{insert}	line#	pos#	*See below*	"14 chars"

- Text is displayed on as 11 lines of 14 characters.
- Device 52 is the top screen; device 51 is the bottom screen for the surface. Channel number is not supported.
- The command must contain the `DEVICE` or `FADER` number and `LINE` number.
- To clear a line, write a single space string to that line, without the `INSERT` keyword.
- Use the keyword `INSERT` with a position reference to overwrite written text occupying that position and keep exisitng text on the line.
- The text color can be specified by adding a predetermined value to the postion value. White is assumed if no value is added to the postion value.

Artisan Text Color values

The following values are added to the postion value to give the text a specific color:

White	+0
Blue	+16
Green	+32
Red	+48
Cyan	+64
Magenta	+80
Yellow	+96
50% Gray	+112

The following are examples of small font text commands written to the **Wide Softkey Screen:**

```
cmd ae1 surface1 text device52 line2 pos1 "This is Text"
cmd ae1 surface1 text device51 line6 pos1 " "
```

Artisan Wide Softkey Route Select

This command is described in full later in this chapter. When using the **Route Select** command, the selection can be written to the screen next to the relevant button using these locations:

Button 1	device52 line2 pos1
Button 2	device52 line6 pos1
Button 3	device52 line10 pos1
Button 4	device51 line2 pos1
Button 5	device51 line6 pos1
Button 6	device51 line10 pos1

Artisan Meter Bridge Screens

The *Artisan* **Wide Meter Bridge** (MLX-WBRIDGE) has six screens. The Artisan Wide Bridge supports a clock, a timer and text on any of the 6 LCD screens. You cannot send auxiliary meter displays to the bridge in an Artisan configuration as those meters are already displayed on the Master module by default.

The *Artisan* **Narrow Meter Bridge** (MLX-BRIDGE) has two screens. The Artisan Narrow Bridge supports a clock, a timer and text on any of the 2 LCD screens. You cannot send auxiliary meter displays to the bridge in an Artisan configuration as those meters are already displayed on the Master module by default.

The screens of a Wide Bridge are shown below:

Figure 7 - Artisan Meter Bridge

In the above example, the clock and the timer are shown on LCD 1 and LCD 5 respectively. They can however, be located on any other screen through the use of triggers. Therefore, any screen not being used for a clock or a timer may also be used for text.

Each screen has 8 lines of text available, which must be addressed using the following channel & line numbers:

Artisan Wide Meter Bridge Screens

Screen	Purpose	Device / Channel	Lines	Notes
Screen 1	Clock	device53	1-8	
Screen 2	User text	device53	65-72	
Screen 3	User text	device53	129-136	
Screen 4	Master 2 Meter	device54	1-8	Text can go on lines 7 & 8 below the meter
Screen 5	Timer	device54	65-72	
Screen 6	User text	device54	129-136	
LCD	Main meter label	device53	16	16 characters available

Artisan Narrow Meter Bridge Screens

Screen	Purpose	Device / Channel	Lines	Notes
Left	Monitor meter & user text	device53	1-8	
Right	Monitor meter & user text	device53	1-8	
LCD	Main meter label	device53	16	16 characters available

Figure 8 - Artisan Meter Bridge Screens 1-3

Artisan Clock

This command is used to write a clock display to the Meter Bridge Screens.

Keyword	AE	Surface	Keyword	Screen	Line	Options
cmd	ae#	surf# | surface #	clock	chan# | channel#	line#	{clear}

- This command enables a large clock display on screen 1 of the *Artisan* **Meter Bridge**.
- For the *Artisan* **Meter Bridge**, use `DEVICE53 LINE15` to set the clock.
- A standard text clock can also be written to other screens if desired.
- The display is continuously updated until a `CLOCK CLEAR` command is used.
- The **Clock Clear** command must contain the same **Audio Engine**, **Surface**, line and position numbers as the original **Clock** command.

The following are examples of how to display and clear the clock:

```
cmd ae1 surface1 clock device53 line15
cmd ae1 surface1 clock clear device53 line15
```

Artisan Meter Bridge Text

This command is used to write text to the Meter Bridge Screens.

Keyword	AE	Surface	Keyword	Screen	Options	Line	Pos	Color	Text
cmd	ae#	surf# | surface #	text	device#	{insert}	line#	pos#	*See below*	"18 chars"
cmd	ae#	surf# | surface #	text	chan# | channel#	{insert}	line#	pos#	*See below*	"18 chars"

- Text is displayed on Screens 1-6 as 8 lines of 18 characters.
- See page 118 for the screen addresses & lines.
- This command can also be used to set the text on the LCD display below the main meter.
- The command must contain the `DEVICE` or `FADER` number and `LINE` number.
- To clear a line, write a single space string to that line, without the `INSERT` keyword.
- Use the keyword `INSERT` to replace only the written text and keep other text on the line.
- The text color can be specified by adding a predetermined value to the postion value. White is assumed if no value is added to the postion value.

Artisan Text Color values

The following values are added to the postion value to give the text a specific color:

White	+0
Blue	+16
Green	+32
Red	+48
Cyan	+64
Magenta	+80
Yellow	+96
50% Gray	+112

The following are examples of small font text commands written to the **Meter Bridge:**

```
cmd ae1 surface1 text device53 line66 pos1 green "This is Text"
cmd ae1 surface1 text device53 line16 pos1 " "
```

Figure 9 - Artisan Meter Bridge Screens 4-6

Artisan Master 1/2 Meter

The **Master 1/2 Meter** is set to follow the **Master 1/2** source, or depending on your **Audio Engine** configuration, another source device. This is set by the **Audio Engine**, and is not dependent upon *CommandBuilder*.

Lines 7 & 8 on Screen 1 & 4 can be used for additional text displays, as per the *Artisan* **Meter Bridge Text** command detailed on the previous page.

Artisan Timer

The *Artisan* **Timer** is controlled automatically on the surface. There are currently no **Trigger** commands to alter the timer functionality. When running in "auto" mode, the timer will reset and display the source name each time a new fader is started (provided that input is set to "timer reset" enabled).

Route 3 Text

The *Route 3* display screen has two modes of operation in which text can be displayed, **Normal Mode** and **Message Mode**.

Normal Mode is used to display labels below a route selection, for example to label a record device name. The label is displayed on the bottom line, with the top line showing the selected route. **Message Mode** is used to prompt the user for a response with the **CH1** and **CH2** buttons on the *Route 3's* front panel. When in **Message Mode** the route displays are temporarily hidden.

Before you can send text to a *Route 3*, you need to know its surface addressing. Addresses are set with thumbwheels on the back of the unit. There are two ways that *Route 3* units can be addressed:

1. Using a standard **Surface Address**, with the thumbwheels set to 01, 04, 07, 10 etc (multiple units can be daisy chained on one **Audio Engine Port** and will use three channels per unit).
2. Using an **Output Address**, under DSP v3 and higher, with the thumbwheels set to the first of three sequential output routes to be controlled by the **Route 3**. These device addresses are allocated by *AEConfig* in the ***Output Settings*** page, and will be hex 6E or higher. In this scenario it doesn't matter which port the *Route 3* is connected to, or where it is in a chain.

The recommended address style depends on the **Port** usage of **Control Surfaces** at your facility.

TIP: Older Route 3 units may require a firmware update to support Output Addressing.

Figure 10 - Route 3

Route 3 Normal Mode Text

This command is used to write a label underneath a router selection on the Route 3.

Keyword	AE	Surface	Channel	Command	Keyword	Keyword	Text
cmd	ae#	surf# | surface #	chan# | channel#	route3	text	label | default	"123456789012"
cmd	ae#	*Not Used*	d[*DeviceName*]	route3	text	label | default	"123456789012"

- With **Surface Addressing**, use a **Surface** number and **Channel** number.
- **Channel** number is 1-3 for the first *Route 3* on a **Port**, 4-6 for the second, and so on.
- With **Output Addressing**, the device number is required.
- This command supports the `d[Destination Device]` notation.
- Required keywords are `ROUTE3 TEXT` and `LABEL` or `DEFAULT`.
- The screen display in **Normal Mode** is divided into three columns of two lines each.
- The upper line in each column is the **Source Device** name for the *Route 3* channel.
- When a new **Source Device** is selected, this line changes automatically to the new **Source Device** name. This upper line can not be changed in any other way.
- The lower line in each column is a user defined text label for that channel.
- This label may have up to 12 characters and is left justified.
- Leading blanks are required to center short labels.
- The user defined text label can be permanently stored as the default label within the *Route 3*. Whenever a new **Source Device** is selected for a channel, that channel's default label is displayed on the lower line overwriting the current label.
- When the default label is permanently stored it is also displayed on lower line for the specified channel. Use the keyword `DEFAULT` in place of `LABEL` to set default text.

The following are examples of temporarily changing the label on a *Route 3* display:

```
cmd ae1 surface2 route3 text label channel1 "Temp 1"
cmd ae1 surface2 route3 text label channel2 "Temp 2"
cmd ae1 surface2 route3 text label channel3 "Temp 3"
cmd ae1 d[OutputName] text line5 "Temp 3"
```

The following are examples of changing the stored default labels and at the same time changing the lower line on a *Route 3* display:

```
cmd ae1 surface2 route3 text default channel1 "Chan 1"
cmd ae1 surface2 route3 text default channel2 "Chan 2"
cmd ae1 surface2 route3 text default channel3 "Chan 3"
```

TIP: The recommended location to set *Route 3* labels is in the Init Trigger.

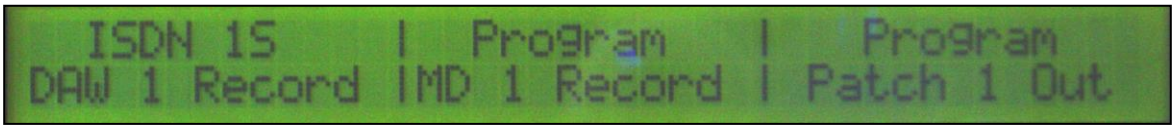


Figure 11 - Route 3 Normal Mode text

Route 3 Set Message Mode

This command is used to set the text mode of a Route 3.

Keyword	AE	Surface	Channel	Command	Keyword	Mode
cmd	ae#	surf# | surface #	chan# | channel#	route3	set	message mode | normal mode
cmd	ae#	*Not Used*	d[*DeviceName*]	route3	set	message mode | normal mode

- With **Surface Addressing**, use a **Surface** number and **Channel** number.
- With **Output Addressing**, the `d[Destination Device]` notation is recommended.
- This command does not support a **Destination Device** number.
- **Channel** number is 1-3 for the first *Route 3* on a **Port**, 4-6 for the second, and so on.
- Required keywords are `ROUTE3 SET NORMAL MODE` or `ROUTE3 SET MESSAGE MODE`.

The following is an example of setting the *Route 3* screen to **Message Mode**:

```
cmd ae1 surf1 channel1 route3 set message mode
```

The following is an example of restoring the *Route 3* screen to **Normal Mode**:

```
cmd ae1 d[MiniDisc Record] route3 set normal mode
```

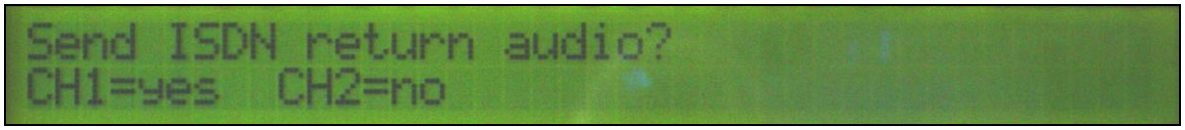


Figure 12 - Route 3 in Message Mode

Route 3 Message Mode Text

This command is used to display messages or questions to a Route 3.

Keyword	AE	Surface	Channel	Command	Keyword	Line	Text
cmd	ae#	surf# | surface #	chan# | channel#	route3	text	line#	"40 characters"
cmd	ae#	*Not Used*	d[*DeviceName*]	route3	text	line#	"40 characters"

- With **Surface Addressing**, use a **Surface** number and a **Channel** number.
- **Channel** number is 1-3 for the first *Route 3* on a **Port**, 4-6 for the second, and so on.
- With **Output Addressing**, the device number is required.
- This command does not support the `d[Destination Device]` notation.
- Required keywords are `ROUTE3 TEXT` and `LINE1` or `LINE2`.
- The line always starts at the far left, so leading blanks are required to center short text lines.
- Use an **If Trigger** for the **CH1** and **CH2** buttons to get responses to desired questions.
- This button press can only be detected when the *Route 3* is in **Message Mode**.
- **Normal Mode** is automatically restored when the **CH1** or **CH2** button is pressed.

The following is an example of sending two lines of text to a *Route 3* display after setting it to **Message Mode**. A response is expected by pressing **CH1** or **CH2**. Adding an **If Timer** statement so that the message display times out to a No (**CH2**) response, improves usability.

```
trigger ae1 d[Port 1 Fader 1 In] route any

     cmd ae1 d[Port 1 Fader 1 In] route3 set message mode
     cmd ae1 d[Port 1 Fader 1 In] route3 text line1 "Send ISDN return audio?"
     cmd ae1 d[Port 1 Fader 1 In] route3 text line2 "CH1=yes  CH2=no"

     if timer 1 wait 20
      cmd ae1 d[Port 1 Fader 1 In] route3 set normal mode
      cmd ae1 d[Port 1 Fader 1 In] route3 text label "No ISDN TB"
     endif
```

The following is an example of a **Trigger** and **Commands** that could be associated with the **CH1** button press after the message was displayed:

```
trigger ae1 d[Port 1 Fader 1 In] route3 button1 on

     cmd cancel timer 1
     cmd ae1 route s[ISDN 1] to d[Port 1 Fader 1 In]
     cmd ae1 d[Port 1 Fader 1 In] route3 text label channel 1 "ISDN TB Set"
```

The following is an example of a **Trigger** and **Commands** that could be associated with the **CH2** button press after the message was displayed:

```
trigger ae1 d[MD 1 Record] route3 button2 on

     cmd cancel timer 1
     cmd ae1 d[Port 1 Fader 1 In] route3 text label channel 1 "No ISDN TB"
```

➥ *See Chapter 7 for information on Route 3 Button Triggers.*

Text Select Functions

These functions are special implementations of the **Selection Screen** commands. They allow the user to select options from the **Selection Screen**. The **Route Select** function can also be used to present a list of inputs for a particular router **Crosspoint** (e.g. a record selector).

The functions are available on the *Numix II*, *Remora, Artisan* & *Mosaic (v1)* **Surfaces**.

> ⚠ Current **ROC** and **Mosaic** consoles use the DISPLAY keyword and not a console specific keyword. This is documented in the current **ROC** and **Mosaic** manual.

Variable Select

- This set of commands allows the value of a **User Variable** to be set by selecting from a list. This value can then be immediately tested to execute sets of **Action Commands**.
- The displayed selection list text is also user defined.
- The **Select Knob** is used to move the highlight bar to the desired line and then the **Accept** button is pressed to make that selection.
- The **Cancel** button can be pressed to exit **Selection Mode** without making a selection.

The examples for the individual **Variable Select** commands follow the command descriptions.

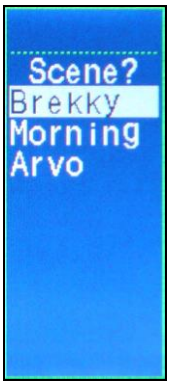


Figure 13 - Variable Select menu (Remora shown)

Variable Select Set Mode

This command is used to set the Variable Select mode on a surface.

Keyword	AE	Surface	Type	Variable	Keywords
cmd	ae#	surf# | surface #	numix1 | numix2 | remora | artisan | mosaic | display	*vUserVariable*	set selection mode

- This command sets the surface display to **Selection Mode**.
- The **Audio Engine** number, **Surface** number and type, and the keywords `SELECTION MODE` are required.
- The surface must be `NUMIX1`, `NUMIX2`, `REMORA`, `ARTISAN` or `MOSAIC`.
- If using a current generation ROC or Mosaic console, use the `DISPLAY` keyword in place of the surface type keyword.
- If no surface type is given, then `NUMIX1` is assumed.
- The keyword `SET` is optional.
- The selection value is saved in the **User Variable** specified with `vUserVariable` notation.
- The **User Variable** is required, and must be uniquely assigned. The **User Variable** cannot be used by another **Variable Select** or **Route Select Trigger**, or unpredictable results will occur.
- An `IF CANCEL` statement is required.

Variable Select Text

This command is used to enter selection text for the Variable Select mode on a surface.

Keyword	Keyword	Keyword	Line	Text
cmd	text	selection	*# (line)*	"12345678"

- This command is used to set the title and selection text lines for the list.
- The **Audio Engine** number and **Surface** are not required in these commands and are assumed to be those specified in the `SET SELECTION MODE` command.
- The keywords `TEXT SELECTION TITLE` are required for the title line of the selection list.
- The keywords `TEXT SELECTION` and the line number are required for each line in the list.

Variable Select If Accept

This Test Statement is used to test for an accept selection from the user.

Keyword	Keyword	Selection Number	Keyword
if	accept	selection = #	{then}

- This **Test Statement** tests for the selection that is accepted by the user and contains the **Action Commands** to be executed.
- There is usually one group of commands for each value of selection line in the list.
- The keywords `IF ACCEPT SELECTION =` and the selection line value are required.
- The keyword `THEN` is optional.
- The **Audio Engine** and **Surface** numbers are not required and are assumed to be those specified in the above `SET SELECTION MODE` command.
- The commands to be executed must contain the required **Audio Engine** and **Surface**.

- The commands to be executed must be followed by the required `ENDIF` line.
- The selection is stored in the **User Variable** from the `SET SELECTION MODE` command.

Variable Select If Cancel

This Test Statement is used to test for a cancel selection from the user.

Keyword	Keyword	Keyword	Time	Keyword
if	cancel	delay	*# (seconds)*	{then}

- This **Test Statement** contains commands to be executed when **Cancel** is pressed.
- This is required even if there are no commands to be executed when **Cancel** is pressed.
- The keywords `IF CANCEL` are required.
- The keyword `THEN` is optional.
- The **Audio Engine** and **Surface** numbers are not required and are assumed to be those specified in the `SET SELECTION MODE` command.
- The commands to be executed must contain the required **Audio Engine** and **Surface**.
- The commands to be executed must be followed by the required `ENDIF` line.
- The statement contains an internal timer that will execute the specified commands when it expires. The delay time is set using the keyword `DELAY` followed by the number of seconds
- The `IF CANCEL` automatically sets its timer to 60 seconds.
- When the **Cancel** button is pressed or the internal timer expires, the value of the **User Variable** is not changed and the **Selection Mode** display on the surface is cleared.

The following is a complete example of a **Variable Select** function:

```
cmd ae1 surface1 remora vCurrentShow set selection mode
cmd text selection title " Scene? "
cmd text selection 1 "Brekky"
cmd text selection 2 "Morning"
cmd text selection 3 "Arvo"

if accept selection = 1 then
 cmd ae1 surface1 route s[Host Mic] to d[Port1 Fader 1 In]
 cmd ae1 surface1 route s[Guest Mic 1] to d[Port1 Fader 3 In]
 cmd ae1 surface1 route s[Guest Mic 2] to d[Port1 Fader 4 In]
endif

if accept selection = 2 then
 cmd ae1 surface1 route s[Host Mic] to d[Port 1 Fader 4 In]
endif

if accept selection = 3 then
 cmd ae1 surface1 route s[Host Mic] to d[Port 1 Fader 1 In]
endif

if cancel delay 30 then
 cmd ae1 surface1 bridge lamp4 off
endif
```

Current **ROC** and **Mosaic** consoles use the DISPLAY keyword and not a console specific keyword. This is documented in the current **ROC** and **Mosaic** manual. Follow the example in the console manual and not the example above.

Route Select

- The **Route Select** function is similar to the **Variable Select** function.
- The user selects a **Source Device** from a displayed list.
- The items in the selection list will be the available **Source Device** names for a specified **Destination Device** (set in *AEConfig*), similar to a **Fader** input select.
- The **Select Knob** is used to move the highlight bar to the desired **Source Device** and then the **Accept** button is pressed to make that selection.
- The **Cancel** button can be pressed to exit **Selection Mode** without making a selection.

The examples for the individual **Route Select** commands follow the command descriptions.

Figure 14 - Route Select menu (Remora shown)

> Current **ROC** and **Mosaic** consoles use the DISPLAY keyword and not a console specific keyword. This is documented in the current **ROC** and **Mosaic** manual.

Route Select Set Mode

This command is used to set the Route Select mode on a surface.

Keyword	AE	Surface	Type	Variable	Keywords	Destination
cmd	ae#	surf# | surface #	numix1 | numix2 | remora| artisan| mosaic | display	*vUserVariable*	set route selection mode	d[*DeviceName*]

- This command sets the surface display to **Route Selection Mode**.
- The **Audio Engine** number, **Surface** number and type, and the keywords `ROUTE SELECTION MODE` are required.
- The surface type must be `NUMIX1`, `NUMIX2`, `REMORA`, `ARTISAN` or `MOSAIC`.
- If using a current generation ROC or Mosaic console, use the keyword `DISPLAY` instead of the surface type keywords above and refer to the console manual for further information.

- If no surface type is given, then `NUMIX1` is assumed. The selection list appears in slightly different displays on each type of surface, so it is important that the correct type is given.
- The keyword `SET` is optional.
- The selected **Source Device** value is saved in the required **User Variable** specified with `vUserVariable` notation.
- The **User Variable** is required, and must be uniquely assigned. The **User Variable** cannot be used by another **Variable Select** or **Route Select Trigger**, or unpredictable results will occur.

Route Select Text

This command is used to enter the title for the Route Select mode on a surface.

Keyword	Keyword	Keyword	Line	Text
cmd	text	selection	title	"1234567890123" *(13 characters)*

- This command is used to set the title for the list.
- The **Audio Engine** number and **Surface** are not required in these commands and are assumed to be those specified in the `SET SELECTION MODE` command.
- The keywords `TEXT SELECTION TITLE` are required for the title line of the selection list.
- The items in the selection list will be the available **Source Device** names for the specified **Destination Device**.
- An `IF CANCEL` statement is required. In May 2005 versions and later, an error will be generated if no `IF CANCEL` follows a **Route Select** command.

Route Select If Accept

This Test Statement is used to test for an accept selection from the user.

Keyword	Keyword	Selection	Keyword	Destination
if	accept	selection	{text}	{device#### | d[*DeviceName*]}

- This **Test Statement** contains commands to be executed when **Accept** is pressed.
- The **Source Device** that is highlighted when the **Accept** button is pressed will be routed to the **Destination Device** specified in the `SET ROUTE SELECTION MODE` command.
- The keyword `THEN` is optional.
- The **Audio Engine** and **Surface** numbers are not required and are assumed to be those specified in the above `SET SELECTION MODE` command.
- The commands to be executed must contain the required **Audio Engine** and **Surface**.
- The commands to be executed must be followed by the required `ENDIF` line.
- The selected **Source Device** number is stored in the **User Variable** specified in the `SET ROUTE SELECTION MODE` command.
- The selected **Source Device** name can also be displayed (tallied) on the surface by using the optional keyword `TEXT` followed by the desired location.
- The text tally destination can be specified by **Device** number or `d[Destination Device]` notation.
- The text tally destination does not support the `SOFTKEY` keyword. Use `DEVICE28` to write to the **Softkey** screen on a *Numix II*.
- The text tally does not support channel numbers, use the channel's **Device Number** to write to the **Numix**, **Remora**, **Artisan** and **Mosaic** fader screens.

Route Select If Cancel

This Test Statement is used to test for a cancel selection from the user.

Keyword	Keyword	Keyword	Time	Keyword
if	cancel	delay	*# (seconds)*	{then}

- This **Test Statement** contains commands to be executed when **Cancel** is pressed.
- This is required even if there are no commands to be executed when **Cancel** is pressed.
- The keywords `IF CANCEL` are required.
- The keyword `THEN` is optional.
- The **Audio Engine** and **Surface** numbers are not required and are assumed to be those specified in the `SET SELECTION MODE` command.
- The commands to be executed must contain the required **Audio Engine** and **Surface**.
- The commands to be executed must be followed by the required `ENDIF` line.
- The statement contains an internal timer that will execute the specified commands when it expires. The delay time is set using the keyword `DELAY` followed by the number of seconds.
- The `IF CANCEL` automatically sets its timer to 60 seconds if no time is specified.
- When the **Cancel** button is pressed or the internal timer expires, the value of the **User Variable** is not changed and the **Selection Mode** display on the surface is cleared.

The following is a complete example of the **Route Select** function:

```
cmd ae1 surface1 bridge lamp 3 on
cmd ae1 numix2 vFader1Route set route selection mode d[Port1 Fader 1 In]

cmd text route selection title "Fader 1"

if accept selection
 cmd ae1 surface1 bridge lamp 3 off
 cmd ae1 surface1 d[Port 1 Fader 1 In] bus 0 on
endif

if cancel delay 30 then
 cmd ae1 surface 1 bridge lamp 3 off
endif
```

Current **ROC** and **Mosaic** consoles use the DISPLAY keyword and not a console specific keyword. This is documented in the current **ROC** and **Mosaic** manual. Follow the example in the console manual instead of the sample above.

14 Additional Surface Commands

Introduction

Some **Logitek Surfaces** have additional features to enhance the operation and usability. This section details **Surface** specific feature commands.

Artisan/Mosaic Features

The *Artisan* and *Mosaic* consoles are fitted with a variety of different button lamps.

All programmable **Softkey** buttons support 16 brightness levels, which can be used to give the operator further feedback (for example, an Intercom lamp may glow brightly when the station is calling, but dimly when it is in use from another studio).

The on & off lamps in the faders (no off button for the *Artisan*) can be changed to many different colors using an RGB color value. This can be used to denote “ready” or “EOM” states on a particular source, or the main microphone color could be changed to make it easy to find.

In addition, all lamps support a three speed flash which can be used to denote different events.

Figure 15 - Mosaic Surface

Artisan Set Channel Color

Sets the color of an Artisan Channel On lamp.

Keyword	Engine	Surface	Device	Bus	Keyword	Value	Option
cmd	ae#	surf# | surface#	chan# | channel# | fader#	chon	set color | artisan | tally	RGB000015	slow | medium | fast
cmd	ae#	surf# | surface#	s[*Device Name*]	chon	set color | artisan | tally	RGB001500	slow | medium | fast
cmd	ae#	*Not Used*	device####	chon	set color | artisan | tally	RGB000015	slow | medium | fast
cmd	ae#	*Not Used*	d[*Device Name*]	chon	set color | artisan | tally	RGB151515	slow | medium | fast

- The **Channel On** buttons on the *Artisan* support a variety of colors and flash rates.
- To set fixed colors for specific faders, use the **Init Trigger** and send the color command using a `SURFACE` and `CHANNEL` or `FADER`; `DEVICE` number, or `d[Device]` notation.
- To change colors based on a particular input, establish a **Route Trigger** and send the appropriate color command using the `s[Source]` notation.
- The color value is set with individual Red, Green and Blue values from 0-15.
- Values higher than 15 for each color are not allowed and will result in an error.
- Due to the inherent properties of the three colored LEDs, many color combinations may be dull or unattractive. The values below are a recommended starting point.
- The keyword `ARTISAN` must be used.
- The keyword `TALLY` must be used. This allows the setting of color based on a High (ON) or Low (OFF) tally.
- An optional keyword `SLOW`, `MEDIUM` or `FAST` can be used to set a flash rate on the lamp.
- A flash rate of 0 is assumed if the keyword is omitted, and equates to solid illumination.
- The lamp can be set to flash even when its current state is off.
- It is not possible to set the flash rate without setting color values.
  Colors and flash rates are stored in the lamp and apply when it next comes on. Turning the lamp or switch off does not reset the stored values.

Red	RGB150000	**Purple**	RGB030015
Green	RGB001500	**Light Aqua**	RGB021411
Blue	RGB000015	**Dark Aqua**	RGB000915
White	RGB151515	**Orange**	RGB151200
Crimson	RGB040000	**Yellow**	RGB091500
Blue White	RGB041215	**Magenta**	RGB150015

The following is an example of an **Artisan Set Channel Color** command to change the lamp color to GREEN for the low (OFF) tally:

```
trigger ae1 device0b bus0 off

     cmd ae1 surf1 device0b artisan set color chon RGB001500 tally
```

The following is an example of an **Artisan Set Channel Color** command to change the lamp color to RED for the high (ON) tally:

```
trigger ae1 device0b bus0 on

     cmd ae1 surf1 device0b artisan set color chon RGB150000 tally
```

Mosaic Set Channel Color

Sets the color of a Mosaic Channel On or Channel Off lamp.

Keyword	Engine	Surface	Device	Bus	Keyword	Value	Option
cmd	ae#	surf# | surface#	chan# | channel# | fader#	choff | chon	set color | mosaic	RGB000015	slow | medium | fast
cmd	ae#	surf# | surface#	s[*Device Name*]	choff | chon	set color | mosaic	RGB001500	slow | medium | fast
cmd	ae#	*Not Used*	device####	choff | chon	set color | mosaic	RGB000015	slow | medium | fast
cmd	ae#	*Not Used*	d[*Device Name*]	choff | chon	set color | mosaic	RGB151515	slow | medium | fast

- The **Channel On** and **Off** buttons on the *Mosaic* support a variety of colors and flash rates.
- To set fixed colors for specific faders, use the **Init Trigger** and send the color command using a `SURFACE` and `CHANNEL` or `FADER`; `DEVICE` number, or `d[Device]` notation.
- To change colors based on a particular input, establish a **Route Trigger** and send the appropriate color command using the `s[Source]` notation.
- The color value is set with individual Red, Green and Blue values from 0-15.
- Values higher than 15 for each color are not allowed and will result in an error.
- Due to the inherent properties of the three colored LEDs, many color combinations may be dull or unattractive. The values below are a recommended starting point.
- The keyword `MOSAIC` must be used.
- An optional keyword `SLOW`, `MEDIUM` or `FAST` can be used to set a flash rate on the lamp.
- A flash rate of 0 is assumed if the keyword is omitted, and equates to solid illumination.
- The lamp can be set to flash even when its current state is off.
- It is not possible to set the flash rate without setting color values.
- The flash rate option is only supported in *Mosaic* v2.0 and above.
- `CHOFF` refers to the lower button and `CHON` the upper button, even if their function has been swapped using a Mosaic feature command.
- Colors and flash rates are stored in the lamp and apply when it next comes on. Turning the lamp or switch off does not reset the stored values.
- This command pertains to both the current Mosaic console and Mosaic version 1 (manufactured before December 2013)

Red	RGB150000
Green	RGB001500
Blue	RGB000015
White	RGB151515
Crimson	RGB040000
Blue White	RGB041215
Purple	RGB030015
Light Aqua	RGB021411
Dark Aqua	RGB000915
Orange	RGB151200
Yellow	RGB091500
Magenta	RGB150015

The PDF version of this manual shows each of the above colors from left to right.

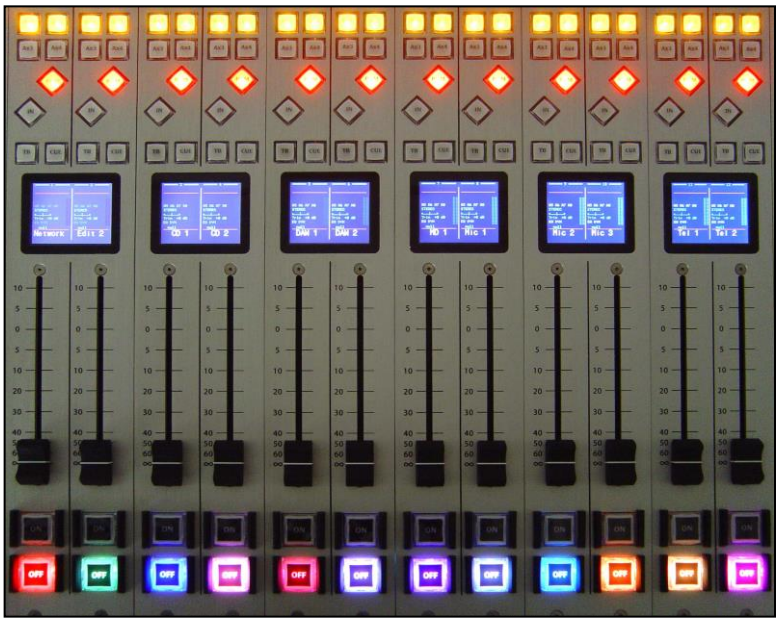


Figure 16 – Artisan/Mosaic Fader lamps colors

Artisan Set Lamp Intensity

Sets the intensity of an Artisan small button lamp.

Keyword	Engine	Surface	Device	Bus	Keyword	Value
cmd	ae#	surf# | surface#	chan# | channel#	bus# | lamp#	set intensity | artisan	0-15 | DIM | MID(DLE) | BRIGHT
cmd	ae#	*Not Used*	device####	bus#	set intensity | artisan	0-15 | DIM | MID(DLE) | BRIGHT
cmd	ae#	*Not Used*	d[*Device Name*]	bus#	set intensity | artisan	0-15 | DIM | MID(DLE) | BRIGHT

- The small and large softkey buttons on the Artisan support 16 brightness levels.
- Brightness levels are stored by the lamp and do not change until a new value is set or the **Surface** is power cycled. All lamps default to maximum brightness on **Surface** power-up.
- This command uses the required keywords `SET INTENSITY`.
- This command must include the **Audio Engine** number, **Device** or **Destination**, and the required keyword `LAMP` or `BUS`.
- The **Device** can be specified using the `d[Destination]` notation.
- If using keyword `LAMP`, a *Mosaic* specific lamp address must also be used.
- This command applies to the lamp in the button, not the button itself.
- The keyword `ARTISAN` must be used.
- The keyword `BUTTON` must NOT be used.
- The intensity level is set with a value from 0-15, or the keywords `DIM` (=0), `MIDDLE` (=8) or `BRIGHT` (=15). Only one intensity keyword should be used.
- The difference in value between adjacent brightness levels is quite subtle, particularly on the brighter (higher) values). Values 0-11 are pictured below.

The following are examples of the **Set Intensity** commands:

```
cmd ae1 device27 bus88 artisan set intensity 0
cmd ae1 surf1 chan13 bus44 artisan set intensity dim
cmd ae1 d[Ctrl – Surf1 GPI out] bus99 artisan set intensity bright
```

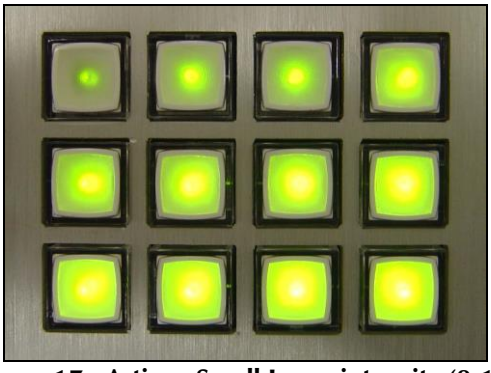


Figure 17 - Artisan Small Lamp intensity (0-11)

15 Legacy Consoles

These consoles are no longer in production but still compatible with a JetStream system.

Remora Screens

The *Remora* has one large color screen for each wedge. Though there are not separate **Selector** and **Fader Wedges** in the *Remora*, it can still perform the **Selection** and **Question Screen** functions.

Text and labels can be displayed in white on these screens. In addition, a colored **Message Arrow** can be displayed.

Remora Selector Screens

The *Remora* has a small section of the **Selector Wedge** (*Remora 4* module) devoted to selector screens. These are the **Selections Screen** and the **Question Screen**. Each of these screens has different capabilities and requires different style text commands.

Due to its smaller profile, the *Remora* does not have a separate **Softkeys Screen** for displaying text.

The main module of a *Remora* console (the *REM4*) has four faders, plus some additional space for monitoring selections & menus as shown below.

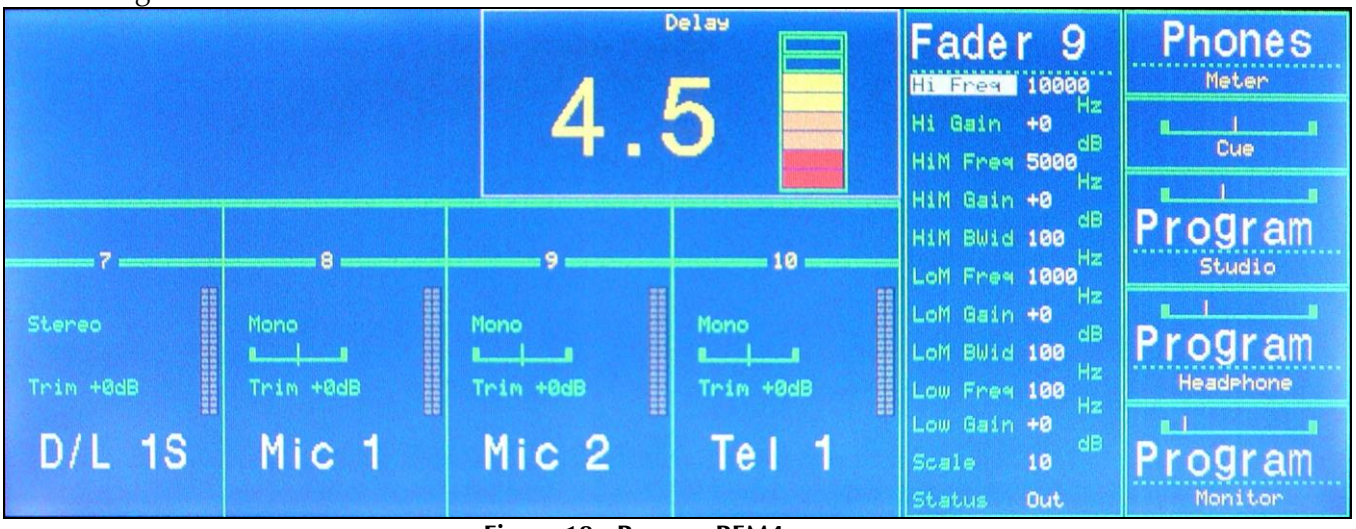


Figure 18 - Remora REM4 screen

TIP: The **Logitek** vScreen program can be used on a PC in the studio to display text messages from **Triggers**. This can be used to emulate a **Surface Softkey Screen**.

Remora Questions Screen Text

These commands are used to write text to the Remora Question Screen.

Keyword	AE	Surface	Keyword	Screen	Line	Pos	Text/Keyword
cmd	ae#	surf# | surface #	set	question			mode
cmd	ae#	surf# | surface #	text	question	line# *(1-2)*	pos# *(1-29)*	"29 characters"

- The **Question Screen** will display 2 lines of 29 characters each.
- This screen is used to ask a question, with the user able to select the **ACCEPT** or **CANCEL** buttons.
- Before text can be written to the **Question Screen**, it must be placed in **Question Mode**, using the keywords `QUESTION MODE`.
- Placing the **Question Screen** in this mode clears all existing text from the **Question Screen**.
- Text commands in **Question Mode** are displayed immediately when received by the surface.
- Use the `IF BUTTON CANCEL` and `IF BUTTON ACCEPT` **Conditional Triggers** to react to the user selection.
- You must use `CLEAR ALL` to clear the **Question Screen** when **ACCEPT** or **CANCEL** is pressed, otherwise the screen will not clear.

The following is an example of placing the **Question Screen** into the **Question Mode** and then displaying several lines of text:

```
cmd ae1 surface2 set question mode
cmd ae1 surface2 text question line1 pos1 "Assign Studio A to air?"
if ae1 surf1 button cancel delay30
 cmd ae1 question text clear all
endif
```

➥ *See Chapter 9 for more information on Conditional Triggers to accept or cancel.*

Remora Selections Screen Text

The **Selections Screen** can be used to display a menu of options to the user. These can be a user defined list, or a list of inputs for a router Crosspoint (e.g. record source selector). This command is available on the *Numix I*, *Numix II* and *Remora*, and therefore is documented in a separate section.

➥ *See Text Selection Screens at the end of this chapter for more information.*

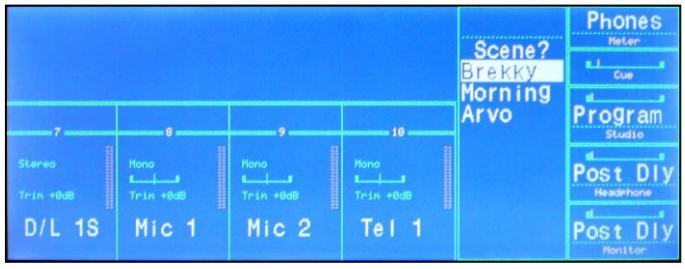


Figure 20 - Remora Selections screen

Figure 19 - Remora Question Screen

Remora Fader Screens

The fader section of the screen on the *Remora* is capable of displaying three types of text. These include small and large font text to the upper half of the screen and large font text to the label section of a fader. This screen will also display a special **Message Arrow** in the upper half of the screen. The *JetStream Server* computer clock and an external temperature device can also be displayed.

Each **Fader Wedge** can display two text sizes. Four lines of large text or twelve lines of small text can be displayed. When in small text mode, one line of large text can be displayed at the top of the screen.

Whilst the *Remora* has color screens, the majority of text functions will write text in white only. However, the **Message Arrow** function is able to display messages in a number of colors.

Figure 21 - Remora Fader Screen showing small text

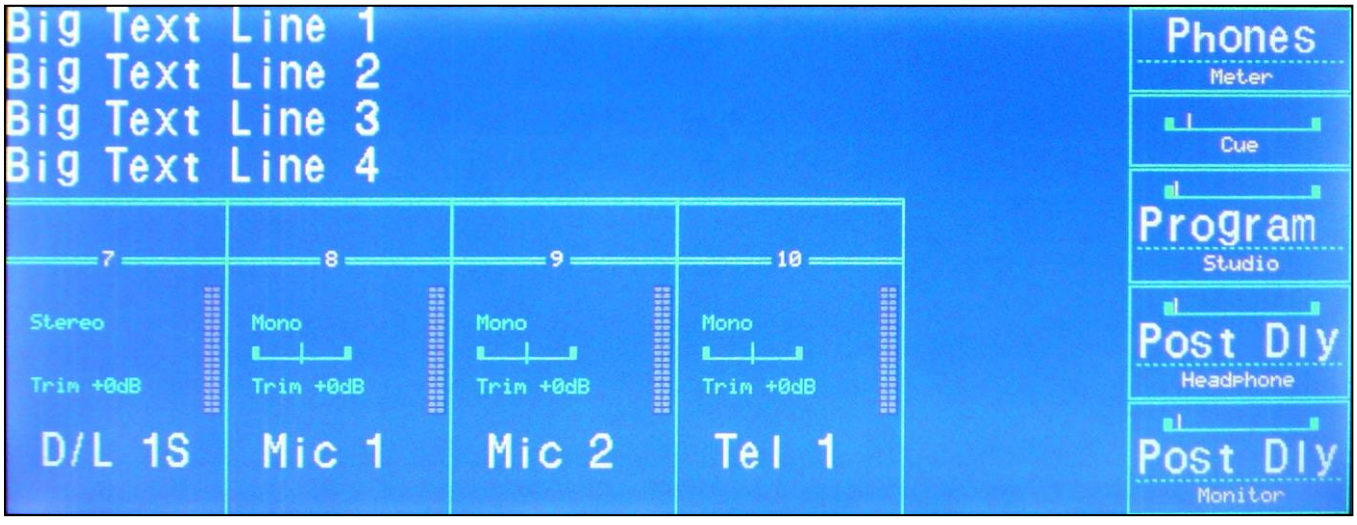


Figure 22 - Remora Fader Screen showing large text

Remora Small Font Text

This command is used to write a small font text to the upper half of the Fader Wedge Screens.

Keyword	AE	Surface	Keyword	Screen	Options	Line	Pos	Text
cmd	ae#	surf# | surface #	text	chan# | channel#	{insert}	line#	pos#	"36 chars"
cmd	ae#	surf# | surface #	text	fader#	{insert}	line#	pos#	"36 chars"

- Small text is displayed on the upper half of the screen as 12 lines of 36 characters.
- This text is divided into two columns of six lines each.
- There are two lines for each channel on the surface
- The first three channels are the left column; the second three channels are the right column.
- Channels 1-6 are on the first wedge, 7-12 on the second, and so on.
- The command must contain the `CHANNEL` or `FADER` number and `LINE1` or `LINE2`.
- To clear a line, write a single space string to that line, without the `INSERT` keyword.
- Use the keyword `INSERT` to replace only the written text and keep other text on the line.

The following are examples of small font text commands written to the **Fader Wedge:**

```
cmd ae1 surface1 text channel1 line1 pos1 "This is Text"
cmd ae1 surface1 text channel6 line2 pos1 " "
```

Remora Large Font Text

This command is used to write a large font text to the upper half of the Fader Wedge Screens.

Keyword	AE	Surface	Keyword	Screen	Options	Line	Pos	Text
cmd	ae#	surf# | surface #	text	chan# | channel#	{insert}	line#	pos#	"52 chars"
cmd	ae#	surf# | surface #	text	fader#	{insert}	line#	pos#	"52 chars"

- Large text is displayed on the upper half of the screen as 4 lines of 52 characters on a REM6 module. On a REM4 module each line can display 35 characters.
- Large text can be displayed as 1 line of 52 or 35 characters in conjunction with small text below.
- The command must contain a `CHANNEL` or `FADER` number of the **Fader Wedge**, the keyword `BIG` and can be written to `LINE 1`, `LINE 2`, `LINE 3` or `LINE 4`.
- Use the first channel number of the desired *Remora* wedge to access that screen.
- To clear a line, write a single space string to that line, without the `INSERT` keyword.
- Use the keyword `INSERT` to replace only the written text and keep other text on the line.

The following are examples of large font text commands written to the **Fader Wedge**:

```
cmd ae1 surface1 text big channel1 line1 pos1 "This is Text"
cmd ae1 surface1 text big channel6 line3 pos21 " "
```

Remora Label Text

This command is used to write a label to an input on the Fader Wedge Screens.

Keyword	AE	Surface	Screen	Keyword	Keyword	Text
cmd	ae#	surf# | surface #	chan# | channel#	text	label	"12345678"
cmd	ae#	surf# | surface #	fader#	text	label	"12345678"

- The label is large font text displayed in a label space directly above the fader information.
- This label may have a maximum of 8 characters.
- The channel label text lines must contain the `CHANNEL` number on the **Fader Wedge** and the keyword `LABEL`.
- The `LINE` or `POS` keywords are not required for this type of text.
- To clear the label write a blank string of 8 characters
- The label may be replaced by flashing text "PENDING" if a `WHEN OFF` command is used.
- The label is replaced if a `SET ALIAS` command is issued to the source device on that fader.
- The label is cleared when a route is performed on that fader.
- To center a label you must manually pad the text string with spaces.

The following are examples of channel label text commands written to the **Fader Wedge**:

```
cmd ae1 surface1 channel1 text label " Active "
cmd ae1 surface1 channel6 text label "        "
```

Figure 23 - Remora Fader Screen showing label & message

➥ *To set an alias for an input, see the Set Device Alias command in Chapter 12.*

Remora Message Arrow

This command is used to display message above an input on the Fader Wedge Screens.

Keyword	AE	Surface	Destination	Keyword	Color	State	Text
cmd	ae#	*Not Used*	device#### *(dest)*	message	*See*	on | off | flash | pulse	"12345678"
cmd	ae#	*Not Used*	device#### *(source)*	message	*List*	on | off | flash | pulse	"12345678"
cmd	ae#	surf# | surface#	fader#	message	*Below*	on | off | flash | pulse	"12345678"
cmd	ae#	surf# | surface#	chan# | channel#	message	*For*	on | off | flash | pulse	"12345678"
cmd	ae#	*Not Used*	d[*DeviceName*]	message	*Color*	on | off | flash | pulse	"12345678"
cmd	ae#	*Not Used*	d[LastRoute]	message	*Key*	on | off | flash | pulse	"12345678"
cmd	ae#	*Not Used*	s[*DeviceName*]	message	*Words*	on | off | flash | pulse	"12345678"

- A special **Message Arrow** can be displayed above a fader to display an alert to the user.
- An 8 character user defined text string is displayed above the **Message Arrow**.
- More than one **Message Arrow** can be displayed at a time on a *Remora* **Surface**.
- The **Message Arrow** will temporarily clear all but the top text line from the screen.
- All text lines will be restored when the **Message Arrow** is turned off.
- This command must include the **Audio Engine** number, **Surface** number, **Channel**, **Fader** or **Device** number and the required keyword `MESSAGE`.
- One of the keywords `CHANNEL`, `FADER` or `DEVICE` is required.
- The required **Device** or **Fader** number may be given using the `s[Source Device]` or `d[Destination Device]` notation instead of using the `DEVICE` keyword.
- The special system destination `d[LastRoute]` will use the **Destination Device** number of the last **Route** command that was executed on the specified **Audio Engine**.
- If the **Device Number** is a **Source Device** number or a named `s[Source Device]`, the command will affect all instances of that **Source Device** on the specified **Audio Engine**. Unlike the **On/Off** command, a **Message Arrow** is not directed at a particular **Surface** when using a **Source Device**.
- The **Message Arrow** may be turned on or off, set to flash continuously, or made to flash once only.
- One of the keywords `ON`, `OFF`, `PULSE` or `FLASH` is needed to set the mode of operation.
- If the mode keywords are missing, the message will default to `ON`.
- The keyword `PULSE` will cause the **Message Arrow** to go on and off only once.
- The keyword `FLAG` is no longer required, but will not cause harm if present.
- Since the *Remora* has a color screen, the color of the **Message Arrow** and text can be set by using one of the optional keywords `WHITE`, `RED`, `YELLOW`, `GREEN`, `AQUA`, or `MAGENTA`.
- If no color keyword is selected the **Message Arrow** and text are displayed in white.

The following are examples of **Message Arrow** command written to the **Fader Wedge**:

```
cmd ae1 surface1 channel1 message green on "On-Air"
cmd ae4 surface2 d[Port1 Fader 2 In] message off
cmd ae4 surface2 d[LastRoute] message off
cmd ae6 surface3 fader7 message aqua flash "On-Air"
cmd ae2 s[CD1] message red flash "On-Air"
cmd ae2 device0100 message yellow pulse "On-Air"
cmd ae2 device000b message green on "On-Air"
```

Remora Clock

This command is used to write a clock display to the Softkeys or Fader Wedge Screens.

Keyword	AE	Surface	Keyword	Size	Screen	Line	Pos	Options
cmd	ae#	surf# | surface #	clock	{big}	chan# | channel#	line#	pos#	{clear} | {insert}
cmd	ae#	surf# | surface #	clock	{big}	fader#	line#	pos#	{clear} | {insert}

- This command is similar to the text command but uses the keyword `CLOCK` instead of the keyword `TEXT`. See **Fader Wedge** and **Softkeys** commands for line/pos numbers.
- The keyword `SOFTKEY` can not be used with a *Remora* as it has no **Softkey** Screen.
- A quoted string is not required and will be ignored if present.
- The keyword `INSERT` can be used to insert the time display text into an existing line.
- The display is continuously updated until a `CLOCK CLEAR` command is used.
- The **Clock Clear** command must contain the same **Audio Engine**, **Surface**, line and position numbers as the original **Clock** command.

The following are examples of how to display and clear the clock:

```
cmd ae1 surface2 clock softkey insert line3 pos6
cmd ae3 surface1 clock big channel7 line1 pos1
cmd ae1 surface2 clock clear softkey line3 pos6
```

Remora Temperature

This command is used to write a temperature display to the Softkeys or Fader Wedge Screens.

Keyword	AE	Surface	Keyword	Size	Screen	Line	Pos	Options
cmd	ae#	surf# | surface #	temperature	{big}	chan# | channel#	line#	pos#	{clear} | {insert}
cmd	ae#	surf# | surface #	temperature	{big}	fader#	line#	pos#	{clear} | {insert}

- This command is similar to the text command but uses the keyword `TEMPERATURE` instead of the keyword `TEXT`. See **Fader Wedge** and **Softkeys** commands for line/pos numbers.
- The keyword `SOFTKEY` can not be used with a *Remora* as it has no **Softkey** Screen.
- A quoted string is not required and will be ignored if present.
- The keyword `INSERT` is used to insert the temperature display text into an existing line.
- The display is continuously updated until a `TEMPERATURE CLEAR` command is used.
- The **Temperature Clear** command must contain the same **Audio Engine**, **Surface**, line and position numbers as the original **Temperature** command.

The following are examples of how to display and clear the temperature:

```
cmd ae1 surface2 temperature softkey insert line3 pos6
cmd ae3 surface1 temperature big channel7 line1 pos1
cmd ae1 surface2 temperature clear softkey line3 pos6
```

Mosaic (v1) Screens

Information in this section pertains to Mosaic (v1) consoles manufactured before December 2013. Current Mosaic information is in the previous chapter and the current Mosaic manual.

The *Mosaic (v1)* has color LCD screens for each fader, which can be used to show fader information and labels. In addition, the **Meter Bridge**, Monitor module and Wide Softkey module have screens that support user defined text.

All screens on the *Mosaic (v1)* are color.

The *Mosaic (v1)* has less user screen space than a *Numix* or *Remora* console. However, the *Mosaic (v1)* is designed to be used in conjunction with *vScreen*, to show additional user defined information (including clocks, meters and text).

➥ *See the vTools Reference Manual for further information on configuring vScreen.*

Figure 24 - Mosaic (v1) Surface

The *Mosaic* introduces many new surface text functions, and therefore commands requiring destination addresses will be affected by the *Mosaic* layout your surfaces' use. Please contact **Logitek Electronic Systems** or your reseller to obtain the latest information.

Mosaic (v1) Fader Screen

The *Mosaic (v1)* **Fader Module** (MLX-20/MLX-FADER) contains one screen (shared between two faders) that allows an **Alias** to be written. The **Alias** is an 8 character string in large font, as shown below.

Figure 25 - Mosaic (v1) Fader Alias (8 character)

Mosaic Label Text

This command is used to write a label to an input on a Fader Screens.

Keyword	AE	Surface	Screen	Keyword	Keyword	Text
cmd	ae#	surf# | surface #	chan# | channel#	text	label	"1234567890123456"
cmd	ae#	surf# | surface #	fader#	text	label	"1234567890123456"

- The label is large font text displayed in a label space directly above the fader information.
- This label may have a maximum of 8 characters.
- The command must contain the `CHANNEL` number and the keyword `LABEL`.
- The `LINE` or `POS` keywords are not required for this type of text.
- To clear the label, write a blank string.
- The label may be replaced by flashing text "PENDING" if a `WHEN OFF` command is used.
- The label is replaced if a `SET ALIAS` command is issued to the source device on that fader.
- The label is cleared when a route is performed on that fader.
- To center a label you must manually pad the text string with spaces.

The following are examples of channel label text commands written to the **Fader Screen**:

```
cmd ae1 surface1 chan1 text label "  MUTE  "
cmd ae1 surface1 chan6 text label "        "
```

➥ *To set an alias for an input, see the Set Device Alias command in Chapter 12.*

Mosaic (v1) Wide Softkey Screen

The *Mosaic (v1)* **Wide Softkey Module** (MLX-40/MLX-WSOFT) has two screens for user text. Each screen has three buttons to the left, for initiating **Triggers**.

Both screens can display user defined text. In addition they can be used with **Route Select** or **Variable Select Triggers** to present menu or router choices to the user. The options are selected by the user with the ***Select*** knob and ***Cancel*** and ***Take*** buttons.

When using **Route Select**, the selected route can tallied next to the button as shown on the right.

Figure 27 - Route Select tally

Figure 26 - Route Select menu

Mosaic Wide Softkey Text

This command is used to write text to the Fader Screens.

Keyword	AE	Surface	Keyword	Screen	Options	Line	Pos	Color	Text
cmd	ae#	surf# | surface #	text	device27	{insert}	line#	pos#	*See below*	"14 chars"
cmd	ae#	surf# | surface #	text	device28	{insert}	line#	pos#	*See below*	"14 chars"

- Text is displayed on as 11 lines of 14 characters.
- Device 28 is the top screen; device 27 is the bottom screen for surface on Port 1. Use device 50 and 4f for surface on Port 2. Channel number is not supported.
- The command must contain the `DEVICE` or `FADER` number and `LINE` number.
- To clear a line, write a single space string to that line, without the `INSERT` keyword.
- Use the keyword `INSERT` with a position reference to overwrite written text occupying that position and keep exisitng text on the line
- The text color can be specified by adding a predetermined value to the postion value. White is assumed if no value is added to the postion value.

Mosaic (v1) Text Color values

The following values are added to the postion value to give the text a specific color:

White	+0
Blue	+16
Green	+32
Red	+48
Cyan	+64

Magenta	+80
Yellow	+96
50% Gray	+112

The following are examples of small font text commands written to the **Wide Softkey Screen:**

```
cmd ae1 surface1 text device28 line2 pos1 "This is Text"
cmd ae1 surface1 text device28 line6 pos1 " "
```

Mosaic (v1) Wide Softkey Route Select

This command is described in full later in this chapter. When using the **Route Select** command, the selection can be written to the screen next to the relevant button using these locations:

Button 1	device28 line2 pos1	**Button 4**	device27 line2 pos1
Button 2	device28 line6 pos1	**Button 5**	device27 line6 pos1
Button 3	device28 line10 pos1	**Button 6**	device27 line10 pos1

Mosaic (v1) Monitor Screen

The *Mosaic (v1)* **Monitor Module** (MLX-34/MLX-MON) has one screen. It does not support user text.

Figure 28 - Mosaic (v1) Monitor Screen

Mosaic (v1) Meter Bridge Screens

The *Mosaic (v1)* **Wide Meter Bridge** (MLX-WBRIDGE) has six screens, as shown below:

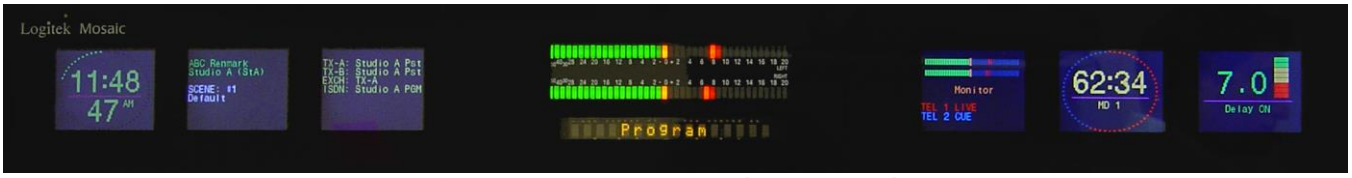


Figure 29 - Mosaic (v1) Wide Meter Bridge

CommandBuilder does not have any keywords to address these screens directly. Each screen has 8 lines of text available, which must be addressed using the following channel & line numbers:

Mosaic (v1) Wide Meter Bridge Screens

Screen	Purpose	Device / Channel	Lines	Notes
Screen 1	Clock & Temperature display	device2b/chan43	1-8	Not available if using clock
Screen 2	User text	device2b/chan43	65-72	
Screen 3	User text	device2b/chan43	129-136	
Screen 4	Monitor meter & user text	device2c/chan44	1-8	Lines 7 & 8 below the monitor meter
Screen 5	Timer	device2c/chan44	65-72	
Screen 6	Delay status & user text	device2c/chan44	129-136	Lines 7 & 8 (135 & 136) below the delay tally
LCD	Main meter label	device2b/chan43	16	16 characters available

Mosaic (v1) Narrow Meter Bridge Screens

Screen	Purpose	Device / Channel	Lines	Notes
Left	Monitor meter & user text	device2b/chan43	1-8	
Right	Monitor meter & user text	device2c/chan44	1-8	
LCD	Main meter label	device2b/chan43	16	16 characters available

Figure 30 - Mosaic (v1) Wide Meter Bridge Screens 1-3

Mosaic (v1) Clock

This command is used to write a clock display to the Meter Bridge Screens.

Keyword	AE	Surface	Keyword	Screen	Line	Options
cmd	ae#	surf# | surface #	clock	chan# | channel#	line#	{clear}

- This command enables a large clock display on screen 1 of the *Mosaic (v1)* **Meter Bridge**.
- For the *Mosaic (v1)* **Wide Meter Bridge**, use `CHAN33 LINE15` to set the clock.
- A standard text clock can also be written to other screens if desired.
- The display is continuously updated until a `CLOCK CLEAR` command is used.
- The **Clock Clear** command must contain the same **Audio Engine**, **Surface**, line and position numbers as the original **Clock** command.

The following are examples of how to display and clear the clock:

```
cmd ae1 surface1 clock chan33 line15
cmd ae1 surface1 clock clear chan33 line15
```

Mosaic (v1) Meter Bridge Text

This command is used to write text to the Meter Bridge Screens.

Keyword	AE	Surface	Keyword	Screen	Options	Line	Pos	Color	Text
cmd	ae#	surf# | surface #	text	device#	{insert}	line#	pos#	*See below*	"18 chars"
cmd	ae#	surf# | surface #	text	chan# | channel#	{insert}	line#	pos#	*See below*	"18 chars"

- Text is displayed on Screens 1-6 as 8 lines of 18 characters.
- See page 118 for the screen addresses & lines.
- This command can also be used to set the text on the LCD display below the main meter.
- The command must contain the `DEVICE` or `FADER` number and `LINE` number.
- To clear a line, write a single space string to that line, without the `INSERT` keyword.
- Use the keyword `INSERT` with a position reference to overwrite written text occupying that position and keep exisitng text on the line.
- The text color can be specified by adding a predetermined value to the postion value. White is assumed if no value is added to the postion value.

Mosaic (v1) Text Color values

The following values are added to the postion value to give the text a specific color:

White	+0
Blue	+16
Green	+32
Red	+48
Cyan	+64
Magenta	+80
Yellow	+96
50% Gray	+112

The following are examples of small font text commands written to the **Meter Bridge:**

```
cmd ae1 surface1 text chan33 line66 pos33 "This is Text"
cmd ae1 surface1 text chan33 line16 pos1 " "
```

Figure 31 - Mosaic (v1) Wide Meter Bridge Screens 4-6

Mosaic (v1) Monitor Meter

The **Monitor Meter** is set to follow the **Monitor** source, or depending on your **Audio Engine** configuration, another source device. This is set by the **Audio Engine**, and is not dependent upon *CommandBuilder*.

Lines 7 & 8 on Screen 4 can be used for additional text displays, as per the *Mosaic (v1)* **Meter Bridge Text** command detailed on the previous page.

Mosaic (v1) Timer

The *Mosaic (v1)* **Timer** is controlled automatically on the surface. There are currently no **Trigger** commands to alter the timer functionality. When running in "auto" mode, the timer will reset and display the source name each time a new fader is started (provided that input is set to "timer reset" enabled).

Mosaic (v1) Delay Display

The **Talk Delay** display will automatically show on Screen 6 on a *Mosaic* **Wide Meter Bridge**, when the *SharcAttack* **Talk Delay** is turned on.

The time display and bar graph are sent by *JetStream Server* when this function is enabled in *JetStream Server*.

Lines 7 & 8 are available for additional delay text tally (as shown above). The standard *Mosaic* **Meter Bridge Text** command can be used for this.

➥ *See page 72 for the Talk Delay command set.*

16 User & System Variables

Introduction

Variables are memory places in *JetStream Server* to store numeric values. **User Variables** can be set inside **Triggers**. **System Variables** are internal to *JetStream Server* and are set upon certain system events occurring.

CommandBuilder has **Test Statements** that allow the value of both types of variables to be tested, similar to most programming languages. These **Test Statements** can be nested to provide very powerful and complex logic tests.

➥ *See Chapter 17 for more details on Test Statements.*

User Variables

User Variables can contain user defined integer numeric values, and **Device** numbers assigned to a **Faders** and **Routes**.

There can be a maximum of 320 user defined variables. The variables are not assigned to a particular **Audio Engine** and are accessible from all **Triggers** and **Procedures**. The changing or setting of the **User Variable** value is itself a **General Trigger**.

➥ *See Chapter 7 for more details on using a User Variable set in a General Trigger.*

System Variables

System Variables store numeric values generated by *JetStream Server* itself. The values of these variables are read only, and can only be changed by *JetStream Server* in response to certain events. They are intended to provide the ability to respond to status and error events in the system.

The changing or setting of a **System Variable** value (by *JetStream Server*) is itself a **General Trigger**.

➥ *See Chapter 7 for more details on using a System Variable set in a General Trigger.*

Defining Variables

User Variables are defined on the ***System Page*** in *CommandBuilder* program. The variable name can contain a maximum of 30 alphanumeric characters. Spaces are not allowed in variable names.

A **User Variable** is referred to using the `vMyVariableName` notation. This notation is the **User Variable** name with the prefix v. The `v` prefix is not part of the variable definition.

➥ *See Chapter 5 for more information on defining User Variables.*

Setting User Variables

Set Variable

This command is used to set a User Variable.

Keyword	Keyword	Variable	Assignment	Value
cmd	set	*vVariableName*	=	*# (integer) | vOtherVariable | zSystemVariable*

- The value of a **User Variable** may be set in any **Trigger** or **Procedure**.
- The variable value is set using the keywords `CMD SET` followed by the variable name using the `vMyVariableName` notation, followed by the equal sign.
- The value of the variable may be set equal to a number, or set equal to the current value of another **User Variable** or **System Variable**.

The following are examples of setting the value of **User Variables**:

```
cmd set vStudio5Phone = 3
cmd set vStudio5Phone = vStudio3Phone
```

Set Variable Talk Time

This command is used to set a User Variable from the current delay time of a Talk Delay Crosspoint.

Keyword	Keyword	Variable	Assignment	Engine	Surface	Device	Keywords
cmd	set	*vVariableName*	=	ae#	*Not Used*	device ####	talk time
cmd	set	*vVariableName*	=	ae#	surf# | surface#	chan# | channel#	talk time
cmd	set	*vVariableName*	=	ae#	*Not Used*	d[*Device Name*]	talk time

- The value of a **User Variable** may be set in any **Trigger** or **Procedure**.
- The variable value is set using the keywords `CMD SET` followed by the variable name using the `vMyVariableName` notation, followed by the equal sign.
- The value of the variable may be set equal to the current time of a **Talk Delay** Crosspoint.
- The keywords `TALK TIME` are required for this command.
- A valid **Audio Engine** and **Destination Device** is required, along with the keywords `TALK TIME`.
- The **Destination Device** may be in `DEVICE####`, `d[Destination Device]` or `SURF# CHAN#` notation.
- The delay time (in tenth of seconds) is multiplied by ten (delay time 1.1 seconds = 11).

The following is an example of setting the value of **User Variables** from a Talk Delay time:

```
cmd set vStudio5DelayTime = ae1 d[Port1 Delay CP1 In] talk time
```

Route Variables

A **User Variable** can also be used to store and recall a **Source Device** route. This function is useful for copying **Source Devices** from one **Destination Device** to another. It can be combined with a `TRIGGER ROUTE ANY` to copy a route to a **User Variable** and other devices.

Route Store

This command is used to store a current route to User Variable.

Keyword	Keyword	Keyword	AE	Surf	Destination	Keyword	Variable
cmd	store	route	(ae#	surf# | surface #	chan# | channel #)	{to}	*vVariableName*
cmd	store	route	(ae#	surf# | surface #	fader #)	{to}	*vVariableName*
cmd	store	route	(ae#	*Not Used*	device####)	{to}	*vVariableName*
cmd	store	route	(ae#	*Not Used*	d[*DeviceName*])	{to}	*vVariableName*

- The **Source Device** number is saved in the variable using the keywords `CMD STORE ROUTE` followed by the **Audio Engine** number and destination, then the **User Variable** name using the `vMyVariableName` notation.
- The parenthesis and the keyword `TO` are optional.

Route Recall

This command is used to recall a route value from a User Variable.

Keyword	Keyword	AE	Surf	Destination	Keyword	Variable
cmd	store	ae#	surf# | surface #	chan# | channel #)	{to}	*vVariableName*
cmd	store	ae#	surf# | surface #	fader #)	{to}	*vVariableName*
cmd	store	ae#	*Not Used*	d[*DeviceName*])	{to}	*vVariableName*

- The **Source Device** number can be recalled from a **User Variable** and assigned to a **Destination Device**.
- The **Destination Device** does not need to be the same as the device used to store the route.
- The route is recalled from the variable using the keywords `CMD RECALL ROUTE` followed by the **Audio Engine** number and destination, then the **User Variable** name using the `vMyVariableName` notation.
- The parenthesis and the keyword `FROM` are optional.
- Device numbers are usually different across **Audio Engines**, even when the inputs are named the same. For this reason a **Route Recall** should only be performed from a **User Variable** that had a matching **Route Store** on the same **Audio Engine**.

The following is an example of storing and recalling a route in a **User Variable**:

```
cmd store  ae1 d[Port1 Fader 1 In] to vPort1Fader1Route
cmd recall vPort1Fader1Route to ae1 d[Port1 Fader 1 In]
```

17 Test Statements

Introduction

Test Statements provide a means to execute different sets of **Action Commands** depending on the outcome of the **Test Statement**. The available Test Statements include **If Toggle Equals**, **If Variable**, **If State Equals**, **If State Scan** and **When Off**.

The use of **Test Statements** allows for very powerful **Trigger** logic. If you have prior programming experience, you will no doubt be familiar with `IF/END IF` type tests.

The **Logitek Scripting Language** has two primary types of **Test Statements** – `IF` and `WHEN`. The group of Action Commands enclosed in a **Test Statement** must be completed with a corresponding `ENDIF` or `ENDWHEN`.

Like most programming languages, *CommandBuilder* supports the nesting of `IF` statements. This allows for even further depth to the logic tests that can be performed.

If Toggle

Tests the current state of a Toggle Trigger.

Keyword	Keyword	Comparator	Value	Keyword
if	toggle	=	1 | 2	{then}

- If the **Toggle State** is equal, the commands enclosed in the **If Toggle** block are executed.
- This **Test Statement** can be used only with **Toggle Triggers**.
- Appropriate sets of commands can be executed depending on the current state.
- The desired commands for a specified state must appear between the IF TOGGLE = statement and its corresponding required ENDIF keyword.
- **Toggle Triggers** start in **Toggle State** 1, and alternate between 1 and 2 on each execution.
- The keyword THEN is optional, but recommended for improved code readability.

The following example illustrates the use of the **If Toggle Equals** statement:

```
if toggle = 1 then
 cmd ae3 surface1 fader2 bus2 on
 cmd ae4 surface2 channel3 bus1 on
 cmd ae1 d[Port2 Fader 4 In] bus 0 on
endif

if toggle = 2 then
 cmd ae3 surface1 fader2 bus 2 off
 cmd ae4 surface2 channel3 bus 1 off
 cmd ae1 d[Port 2 Fader 4 In] bus 0 off
endif
```

➥ *See Chapter 12 for information on setting the Toggle State manually.*

If Variable

Compares the current value of a User or System Variable to another value, variable or device.

Keyword	Variable	Comparator	Value	Keyword
if	*vUserVariable* | *zSystemVariable*	= (*equal*)	*vUserVariable*	{then}
		<> (*not equal*)	*zSystemVariable*	
		< (*less than*)	#	
		> (*greater than*)	ae# s[*DeviceName*]	
		<= (*less/equal*)	ae# d[DeviceName]	
		>= (*greater/equal*)	ae# device####	

- **Variable** values can be compared as equal (=); not equal (<>); less than (<); greater than (>); less than or equal (<=); or greater than or equal (>=).
- If the comparison is true, the enclosed block of **Action Commands** is executed.
- The **Action Commands** to be executed if true must appear between the **If Variable** statement and its corresponding required `ENDIF` keyword.
- A **System Variable** is specified using the `zSystemVariableName` notation.
- A **User Defined** variable is specified using the `vMyVariableName` notation.
- The **Variable** value can be compared to integer values; the current value of other **Variables**; **Source Device** numbers using the `DEVICE` keyword or `s[Source Device]` notation; or **Destination Devices** using the keywords `AE`, `SURF`, and `CHANNEL` or `FADER`, or the `d[Destination Device]` notation.
- The keyword THEN is optional, but recommended for improved code readability.

The following example illustrates the use of the **If Variable** statement:

```
if vPhoneLine = 1 then
 cmd ae3 surface1 route s[Outside Phone 1] to d[Port1 Fader 1 In]
 cmd ae3 surface1 fader1 bus0 on
endif

if vPhoneLine <> vCurrentStudio then
 cmd ae3 surface1 route s[Outside Phone 1] to d[Port2 Fader 2 In]
 cmd ae3 surface1 fader2 bus0 on
endif

if vPhoneLine = s[Outside Phone 1] then
 cmd ae3 surface1 route s[Outside Phone 1] to d[Port1 Fader 3 In]
 cmd ae3 surface1 fader3 bus0 on
endif

if zSupervisorANetworkFail = 1 then
 cmd ae3 surface1 route s[Outside Phone 1] to d[Port1 Fader 7 In]
endif

if zSupervisorASerialFail = 0 then
 cmd ae3 surface1 route s[Outside Phone 1] to d[Port2 Fader 12 In]
endif
```

If Variable And

A variation of If Variable for multiple tests, which must all evaluate as true to execute code block.

Keyword	Test 1	Comparator	Value	Operator	Additional Tests	Keyword
if	(*VariableName*	= | < | > | < | > | <= | >=	*Value*)	{AND}	(…) *optional*	{then}

- The **If Variable Test Statement** can be combined with itself to test values of many variables.
- If all of the individual **Test Statements** are true, the block of commands is executed.
- Each separate statement must be enclosed in parenthesis and joined using the keyword `AND`.
- Spaces are required either side of any **Variable** name to ensure the variable is found.
- Individual **Test Statements** follow the same rules as the **If Variable Test Statement** above.
- The keywords `AND` and `OR` can't be mixed in the same multiple **Test Statement**.

The following are examples of the **If Variable** statement using multiple `AND` test conditions:

```
if ( vPhoneLine = 1 ) and ( vAirStudio = 3 ) and ( vScene = 4 ) then
 cmd ae3 surface1 route s[Outside Phone 1] to d[Port2 Fader 2 In]
 cmd ae3 surface1 fader2 bus0 on
endif

if (vPhoneLine = s[CD 2] ) and ( zJetStream ServerASerialFail = 0 ) then
 cmd ae3 surface1 route s[Outside Phone 2] to d[Port 3 Fader 4 In]
endif
```

If Variable Or

A variation of If Variable for multiple tests, of which one must evaluate as true to execute code block.

Keyword	Test 1	Comparator	Value	Operator	Additional Tests	Keyword
if	(*VariableName*	= | < | > | < | > | <= | >=	*Value*)	{OR}	(…) *optional*	{then}

- The **If Variable Test Statement** can be combined with itself to test values of many variables.
- If one of the individual **Test Statements** is true, the block of commands is executed.
- Each separate statement must be enclosed in parenthesis and joined using the keyword `OR`.
- Spaces are required either side of any **Variable** name to ensure the variable is found.
- Individual **Test Statements** follow the same rules as the **If Variable Test Statement** above.
- The keywords `AND` and `OR` can't be mixed in the same multiple **Test Statement**.

The following are examples of the **If Variable** statement using multiple `OR` test conditions:

```
if ( vPhoneLine = 1 ) or ( vAirStudio = 3 ) or ( vScene = 4 ) then
 cmd ae3 surface1 route s[Outside Phone 1] to d[Port2 Fader 2 In]
 cmd ae3 surface1 fader2 bus0 on
endif

if (vPhoneLine = s[CD 2] ) or ( zSupervisorASerialFail = 0 ) then
 cmd ae3 surface1 route s[Outside Phone 2] to d[Port 3 Fader 4 In]
endif
```

If State

Tests the state of a set of conditions in a specified Audio Engine.

Keyword	Keyword	Test Condition	Keyword
if	state {=}	(*test condition statement* ON | OFF | ROUTE | NOT ROUTE)	{then}

- If the conditions are true, then the command block is executed.
- The **Action Commands** for a specified set of conditions must appear between the `IF STATE` **Test Statement** and its corresponding required `ENDIF` keyword.
- The required state keyword must be `ON`, `OFF`, `ROUTE`, or `NOT ROUTE`.
- The keyword `THEN` is optional, but recommended for improved code readability.
- The EQUALS sign is not required.
- The parentheses are optional when there is only one test condition.
- The parentheses are required for multiple sets of conditions (see following).

The following are examples of the **If State** statement with only one test condition:

```
if state = ( ae1 surface3 channel3 bus0 on ) then
 cmd ae1 surface3 route s[ISDN 2] to d[Port1 Fader3 In]
endif

if state = ( ae1 surface2 route device 010C to channel 6 ) then
 cmd ae1 surface2 d[Port1 Fader3 In] bus0 off
endif

if state = ( ae3 surface2 not route s[CD 1] to d[Port2 Fader5 In] ) then
 cmd ae3 surface2 d[Port1 Fader5 In] bus0 on
endif
```

If State And

Tests the state of multiple conditions in a specified Audio Engine. All must evaluate to true.

Keyword	Keyword	Test Condition	Operator	Additional Tests	Keyword
if	state {=}	(*first test condition*)	{AND}	(*second test condition*) …	{then}

- The **If State Test Statement** can be combined with itself to test multiple states.
- If all of the individual **Test Statements** are true, the block of commands is executed.
- Each separate statement must be enclosed in parenthesis and joined using the keyword `AND`.
- Individual **Test Statements** follow the same rules as the **If State Test Statement** above.
- The keywords `AND` and `OR` can't be mixed in the same multiple test statement.
- The EQUALS sign is not required.

Following is an example of the **If State** test with multiple test conditions using the `AND` condition:

```
if state = (ae1 surf3 chan3 bus0 on) and (ae1 surf3 chan4 bus0 on) then
 cmd ae1 surf3 route s[ISDN 2] to d[Port1 Fader 3 In]
endif
```

If State Or

Tests the state of multiple conditions in a specified Audio Engine. One must evaluate to true.

Keyword	Keyword	Test Condition	Operator	Additional Tests	Keyword
if	state {=}	(*first test condition*)	{OR}	(*second test condition*) …	{then}

- The **If State Test Statement** can be combined with itself to test multiple states.
- If one of the individual **Test Statements** is true, the block of commands is executed.
- Each separate statement must be enclosed in parenthesis and joined using the keyword `OR`.
- Individual **Test Statements** follow the same rules as the **If State Test Statement** above.
- The keywords `AND` and `OR` can't be mixed in the same multiple test statement.
- The EQUALS sign is not required.

Following is an example of the **If State** test with multiple test conditions using the `OR` condition:

```
if state = (ae2 chan1 bus0 off) or (ae2 chan2 bus0 off) then
 cmd ae2 route s[CD 2] to d[Port 1 Fader 4 In]
endif
```

If State Scan

Tests the state of a Source Device wherever it appears in a specified Audio Engine.

Keyword	Keyword	AE	Surface	Source	Bus	State	Keyword
if	state scan	ae#	surf# | surface #	s[Device]	bus#	ON | OFF	{then}
if	state scan	ae#	surf# | surface #	device####	bus#	ON | OFF	{then}

- **If State Scan** is a special case of **If State**, which includes a **Source Device**.
- The faders on the specified **Surface** are checked from left to right until the specified **Source Device** is found. If the **Source** appears more than once, only the first instance is evaluated.
- If the condition is true for that fader, then the commands in the code block are executed.
- The **Source Device** may be specified by device number or `s[Source Device]` notation.
- The **Action Commands** for a specified set of conditions must appear between the `IF STATE SCAN` **Test Statement** and its corresponding required `ENDIF` keyword.
- The required action keyword must be `ON` or `OFF`.
- The keyword `THEN` and parentheses are optional.
- Only one set of conditions may be specified in a single **If State Scan Test Statement**.

The following are examples of the **If State Scan Test Statement**:

```
if state scan ( ae1 surface3 s[CD 1] bus0 on ) then
 cmd ae1 surface3 route s[ISDN 2] to d[Port1 Fader3 In]
endif

if state scan ( ae2 device0104 bus1 off ) then
 cmd ae2 route s[CD 2] to d[Port1 Fader4 In]
endif
```

If Variable Talk Time

Compares the current delay time value of a Talk Delay Crosspoint in a specified Audio Engine.

Keyword	Engine	Surface	Device	Keywords	Comparator	Keyword
if	ae#	*Not Used*	device ####	talk time	= (*equal*) <> (*not equal*) < (*less than*) > (*greater than*) <= (*less/equal*) >= (*greater/equal*)	{then}
if	ae#	surf# | surface#	chan# | channel#	talk time	*As Above*	{then}
if	ae#	*Not Used*	d[*Device Name*]	talk time	*As Above*	{then}

- **Talk Delay** time can be compared as equal (=); not equal (<>); less than (<); greater than (>); less than or equal (<=); or greater than or equal (>=).
- If the comparison is true, the enclosed block of **Action Commands** is executed.
- The **Action Commands** to be executed if true must appear between the **If Talk Time** statement and its corresponding required `ENDIF` keyword.
- A valid **Audio Engine** and **Destination Device** is required, along with the keywords `TALK TIME`.
- The **Destination Device** may be in `DEVICE####`, `d[Destination Device]` or `SURF# CHAN#` notation.
- The delay time (in tenth of seconds) is multiplied by ten (delay time 1.1 seconds = 11).
- The keyword THEN is optional, but recommended for improved code readability.
- This **Test Statement** cannot be combined with other **Test Statements** using `AND / OR`.

The following examples illustrate the use of the **If Talk Time** statement:

```
if ae1 d[Port1 Delay CP1 In] talk time = 70 then
 cmd ae1 surface1 bridge lamp12 on
endif

if ae1 surface1 chan38 talk time < 70 then
 cmd ae1 surface1 bridge lamp12 off
endif
```

When Off

Executes a set of commands when a specified fader is turned off.

Keyword	Keyword	AE	Surface	Destination
when	off	ae#	surf# | surface #	chan# | channel#
when	off	ae#	surf# | surface #	fader#
when	off	ae#	*Not Used*	d[*Device Name*]
when	off	ae#	*Not Used*	device ####

- The **When Off** statement tests a specified **Surface Channel** or **Destination Device** to determine if its **Bus 0** (main on/off switch) is currently off.
- If **Bus 0** is off, the **Action Commands** associated with the **When Off Test Statement** are executed immediately.
- If **Bus 0** is not off, the **Action Commands** are stored and executed when **Bus 0** does go off.
- This **Test Statement** is often used to route a new **Source Device** to a **Fader** when the **Fader** is turned off (preventing it the source from changing while the **Fader** is on-air or in use).
- While the commands are stored, the flashing text "Pending" is displayed over the **Fader**.
- One of the keywords `CHANNEL`, `FADER` or `DEVICE` is required.
- The required **Device** or **Fader** number may be given using the `d[Destination Device]` notation instead of using the `DEVICE` keyword.
- The keyword `BUS` is not used since the **Test Statement** always checks **Bus 0**.
- The **Action Commands** to be executed must appear between the `WHEN OFF` **Test Statement** and its corresponding required `ENDWHEN` keyword.
- The stored commands and "Pending" text can be cleared using the **Clear When** command.

The following examples illustrate the use of the **When Off** statement:

```
when off ae1 surface1 d[Port1 Fader2 In]
 cmd ae1 route s[Phone 1] to d[Port1 Fader 2 In]
endwhen

when off ae2 surface2 fader4
 cmd ae2 surface2 fader4 route s[Phone 1]
endwhen
```

Part C: Examples and application notes

The final section of the manual contains example Triggers and application notes to help you achieve more complex tasks. These examples are based on some of the best Trigger applications used at Logitek sites around the world.

18 Trigger Layout

The layout of **Triggers** is entirely up to the user, and does not affect the way *JetStream Server* operates. However, a logical layout will greatly aid future maintenance and additions to the table.

The recommended layout is to allocate a "stub" **Trigger** for every possible **Trigger Event**, and then code against the **Triggers** that are to be used. This makes it easy to find **Triggers** in the list, and see which buttons, GPIs and other events are allocated. In addition, this layout also makes it easy to add **Triggers** in place in the future.

A "stub" **Trigger** is essentially a placeholder. It contains the **Trigger** definition and description, without any code. It is left "Not Active" until code is added.

If required, **Logitek** can supply example **Trigger** files to assist with your file layout and design.

Allocating Stub Triggers

We suggest allocating blocks of **Triggers** for related **Trigger Events**. The following blocks are suggested:

- Summary (documentation) & Init Triggers
- Audio Engine GPIs
- Surface GPIs
- Surface Bridge Buttons
- Surface Softkey Buttons
- Surface Button Wedge Buttons
- Surface Miscellaneous Triggers

For maximum readability, each block should start at the next "hundred", e.g. 101, 201, etc.

Within each block, allocate the ON and OFF **Triggers** for each relevant **Engine** or **Surface**, e.g. **Audio Engine** GPI ONs at #101, then **Audio Engine** GPI OFFs at #121. This layout and spacing makes it very easy to find related Triggers.

Example Stub Triggers Allocation

Following is a suggested **Trigger** layout for a system with one **Audio Engine**, and one **Surface** with a **Button Wedge**.

01	Site Information (documentation – no active code)
02	Summary of Triggers (documentation – no active code)
03	Summary of Timers (documentation – no active code)
04	Init Trigger
11-100	Audio Engine Triggers (if used)
101-115	Audio Engine GPI On Triggers
121-135	Audio Engine GPI Off Triggers

[repeat this sequence for each Audio Engine]

201-212	Surface GPI On Triggers (use 201-225 for *Mosaic*)
221-232	Surface GPI Off Triggers (use 231-255 for *Mosaic*)

[repeat this sequence for each Surface]

301-312	Surface Bridge Button On Triggers
321-332	Surface Bridge Button Off Triggers

[repeat this sequence for each Surface]

401-412	Surface Softkey Button On Triggers
421-432	Surface Softkey Button Off Triggers

[repeat this sequence for each Surface]

501-524	Surface Button Wedge On Triggers
531-554	Surface Button Wedge Off Triggers

[repeat this sequence for each Surface / Button Wedge]

601+	Miscellaneous Triggers (e.g. fader, input, route, etc)

Each block should be adjusted to the next available block of one hundred where more **Audio Engines** and **Surfaces** are required.

Different **Surfaces** have different buttons and will influence the allocation of **Triggers**. For example, the *Numix II* surface has a **Softkey** panel, whereas the *Remora* does not. The *Mosaic* **Surface** has many different button panels, depending on the modules ordered.

➥ ***Consult the relevant Surface manual for more information on available buttons.***

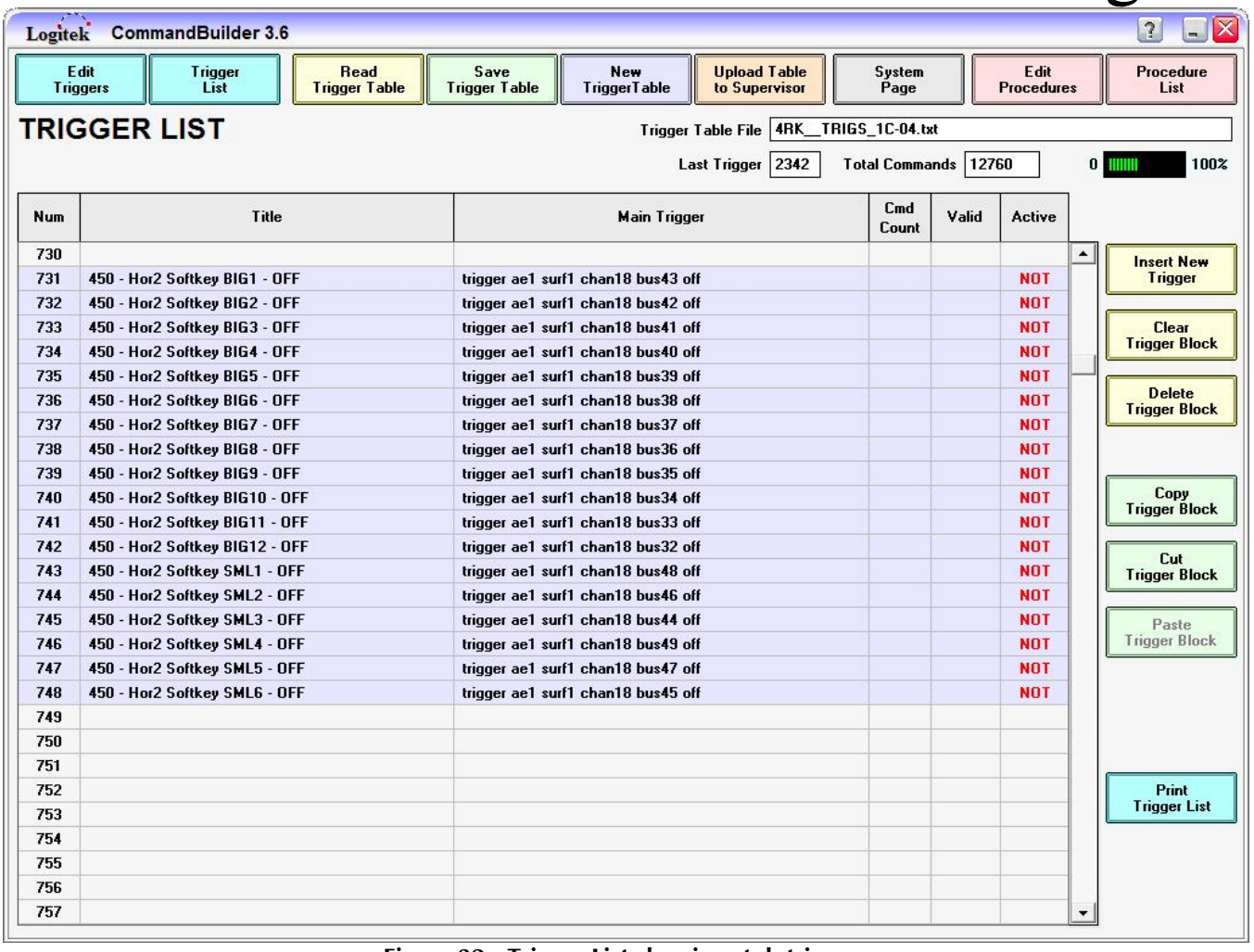


Figure 32 - Trigger List showing stub triggers

The above screenshot shows a section of "stub" **Triggers** following our suggested layout.

Trigger Naming

The naming of **Triggers** is entirely up to the user. Examples from the recommended naming are:

- AE1 - Engine GPI 1 - ON
- St1 - Surface GPI 1 - ON
- St1 - Bridge BUTT1 - ON - Scene 1 "Talk"
- St1 - Softkey BUTT5 - OFF
- St1 - NW24 BUTT1 - ON

In each case, add a short description of the Trigger where relevant, e.g.:

- AE1 Engine GPI 15 - ON - St1 Comm Break Pulse
- St1 - Bridge BUTT1 - ON - Scene 1 "Talk"

A **Trigger** must not have the word `Trigger` as the first word in its name, otherwise *CommandBuilder* will delete its name from the table.

19 Basic Examples

This chapter includes some examples of basic **Trigger** functions that perform useful functions. These are based on popular functions used by **Logitek** stations around the world.

CD Preview

Functionality Description

This is a hotkey function to put the CD fader on CUE and send a GPI to the CD player to preview the start of the cued track.

It requires a CD player that supports a GPI to provide this function (either GPI held high when in preview, or GPI to start and GPI to recue). The example may need to be adjusted to suit the equipment you use.

Components

This function has an ON (button pressed) and OFF (button released) **Trigger**.

Button On Trigger (pressed)

- Check if CD player is OFF (don't execute function while it is playing to air).
- Turn CUE channel ON for CD.
- Turn button lamp ON.
- Turn GPI ON to CD player.
- Enabled OFF trigger (so the OFF component only runs following a successful ON).

Button Off Trigger (released)

- Turn CUE channel OFF for CD.
- Turn button lamp OFF.
- Turn GPI OFF to CD player.

Further Information

- If State Scan – Page 130
- Bus On/Off – Page 55
- Set Trigger Active – Page 74

Script Examples

```
trigger ae1 surf1 chan14 bus37 on

~=======================================^=======================================^=================================~
~ PROCEDURE: CD Preview                                                          REVISED: 18th May 2005
~ DATA/ID: CD1 Preview - ON
~ DESCRIPTION: Puts CD on/off cue. Sends GPI to CD.
~ NOTES/DEPENDENCIES:
~ DEBUG STATUS:
~=======================================^================~===================^=================================~

      if state (ae1 scan surf1 s[StA CD 1] bus00 off)      ~ is CD 1 off?
       cmd ae1 surf1 s[StA CD 1] bus02 on                  ~ Cue on
       cmd ae1 surf1 chan13 bus37 on                       ~ turn lamp on
       cmd ae1 device0001 bus03 on                         ~ GPI on
       cmd set active trigger ae1 surf1 chan14 bus37 off
      endif



trigger ae1 surf1 chan14 bus37 off

~=======================================^=======================================^=================================~
~ PROCEDURE: CD Preview                                                          REVISED: 18th May 2005
~ DATA/ID: CD1 Preview - OFF
~ DESCRIPTION: Puts CD on/off cue. Sends GPI to CD.
~ NOTES/DEPENDENCIES:
~ DEBUG STATUS:
~=======================================^================~===================^=================================~

      cmd ae1 surf1 s[StA CD 1] bus02 off                  ~ Cue off
      cmd ae1 surf1 chan13 bus37 off                       ~ turn lamp off
      cmd ae1 device0001 bus03 off                         ~ GPI off
      cmd set notactive trigger ae1 surf1 chan14 bus37 off
```

Quick Record

Functionality Description

This function will change desired faders (eg microphones and telephone channels) to a record bus, and change the monitoring to suit. Pressing the button again will restore the channels to PGM and monitoring back to original state.

Optionally, this function could also be used to send a GPI to put a device into record mode.

Components

This is a toggle (two-state) button function.

Toggle State = 1 (go into Quick Record)

- Flash the Quick Record button lamp.
- Check if Mic 1-4 are off: If Off, turn off PGM bus and turn on AX1 bus.
- Check if Tel 1 is off: If Off, turn off PGM bus and turn on AX1 bus, set mix minus to AX1, label the Tel 1 fader to show the return path.
- Repeat previous step for Tel 2.
- Store the current Monitor, Operator and Guest Headphone device routes into variables.
- Route AX1 to the Monitor, Operator and Guest Headphones.

Toggle State = 2 (exit Quick Record)

- Turn off the Quick Record button lamp.
- Check if Mic 1-4 PGM is Off: If Off, turn on PGM bus and turn off AX1 bus, turn off channel.
- Check if Tel 1 PGM is Off: If Off, turn on PGM bus, turn off its AX1 bus, turn off channel, set mix minus to PGM, label the Tel 1 fader to show the return path.
- Repeat previous step for Tel 2.
- Recall the previous Monitor, Operator and Guest Headphone routes from their variables.

Further Information

- If Toggle – Page 38
- Bus On/Off – Page 55
- If State Scan – Page 130
- Set Fader Label – Page 61
- Device Store/Recall – Page 124

Script Examples

```
trigger ae1 surf1 chan14 bus35 on toggle                                    ~ <--- put your required button here!

~=====================================^=====================================^==================================~
~ PROCEDURE: Remote record with tally                                       REVISED: 18th May 2005
~ DATA/ID: Quick Record - ON
~ DESCRIPTION: Puts mics and tel on to AX1 ON & AX1 OFF. Change Tel mix minus to AX1. Monitoring to AX1.
~ NOTES/DEPENDENCIES: Toggle function - all changes reversed on toggle. Faders on / PGM are not touched.
~ DEBUG STATUS:
~=====================================^================~====================^==================================~

~== Toggle On ========================================================================================================~
if toggle = 1
 cmd ae1 bridge lamp05 flash continuous              ~ lamp tally on

 if state (ae1 scan surf1 s[St1 Mic 1] bus00 off)    ~ is Mic 1 off?
  cmd ae1 surf1 s[St1 Mic 1] bus01 off               ~ PGM off
  cmd ae1 surf1 s[St1 Mic 1] bus03 on                ~ AX1 on
 endif

 if state (ae1 scan surf1 s[St1 Mic 2] bus00 off)    ~ is Mic 2 off?
  cmd ae1 surf1 s[St1 Mic 2] bus01 off               ~ PGM off
  cmd ae1 surf1 s[St1 Mic 2] bus03 on                ~ AX1 on
 endif

 if state (ae1 scan surf1 s[St1 Mic 3] bus00 off)    ~ is Mic 3 off?
  cmd ae1 surf1 s[St1 Mic 3] bus01 off               ~ PGM off
  cmd ae1 surf1 s[St1 Mic 3] bus03 on                ~ AX1 on
 endif

 if state (ae1 scan surf1 s[St1 Mic 4] bus00 off)    ~ is Mic 4 off?
  cmd ae1 surf1 s[St1 Mic 4] bus01 off               ~ PGM off
  cmd ae1 surf1 s[St1 Mic 4] bus03 on                ~ AX1 on
 endif

 if state (ae1 scan surf1 s[St1 Telephone 1] bus00 off)  ~ is Tel 1 off?
  cmd ae1 surf1 s[St1 Telephone 1] bus01 off         ~ PGM off
  cmd ae1 surf1 s[St1 Telephone 1] bus03 on          ~ AX1 on
  cmd ae1 set surf1 mix- 4 bus4                      ~ set MM4 to AX2
  cmd ae1 surf1 s[St1 Telephone 1] text label "Rtn: AX2"
 endif

 if state (ae1 scan surf1 s[St1 Telephone 2] bus00 off)  ~ is Tel 2 off?
  cmd ae1 surf1 s[St1 Telephone 2] bus01 off         ~ PGM off
  cmd ae1 surf1 s[St1 Telephone 2] bus04 on          ~ AX2 on
  cmd ae1 set surf1 mix- 5 bus4                      ~ set MM5 to AX2
  cmd ae1 surf1 s[St1 Telephone 2] text label "Rtn: AX2"
 endif

 cmd store ae1 d[Port1 Monitor In] route to vMonSt1MonAmp      ~ store Monitor Amp crosspoint setting
 cmd store ae1 d[Port1 Headphones In] route to vMonSt1OprHP    ~ store Opr Headphones crosspoint setting
 cmd store ae1 d[Port1 Studio In] route to vMonSt1GstHP       ~ store Gst Headphones crosspoint setting

 cmd ae1 route s[Port1 Aux 2 Out] to d[Port1 Monitor In]       ~ route AX2 to Monitor
 cmd ae1 route s[Port1 Aux 2 Out] to d[Port1 Headphones In]    ~ route AX2 to headphones
 cmd ae1 route s[Port1 Aux 2 Out] to d[Port1 Studio In]        ~ route AX2 to guest headphones

endif
```

```
~== Toggle Off ==========================================================================================================~
if toggle = 2
 cmd ae1 bridge lamp05 off                              ~ lamp tally off

 if state (ae1 scan surf1 s[St1 Mic 1] bus01 off)      ~ is Mic 1 off PGM?
  cmd ae1 surf1 s[St1 Mic 1] bus01 on                  ~ PGM on
  cmd ae1 surf1 s[St1 Mic 1] bus03 off                 ~ AX1 off
  cmd ae1 surf1 s[St1 Mic 1] bus00 off                 ~ channel off
 endif

 if state (ae1 scan surf1 s[St1 Mic 2] bus01 off)      ~ is Mic 2 off PGM?
  cmd ae1 surf1 s[St1 Mic 2] bus01 on                  ~ PGM on
  cmd ae1 surf1 s[St1 Mic 2] bus03 off                 ~ AX1 off
  cmd ae1 surf1 s[St1 Mic 2] bus00 off                 ~ channel off
 endif

 if state (ae1 scan surf1 s[St1 Mic 3] bus01 off)      ~ is Mic 3 off PGM?
  cmd ae1 surf1 s[St1 Mic 3] bus01 on                  ~ PGM on
  cmd ae1 surf1 s[St1 Mic 3] bus03 off                 ~ AX1 off
  cmd ae1 surf1 s[St1 Mic 3] bus00 off                 ~ channel off
 endif

 if state (ae1 scan surf1 s[St1 Mic 4] bus01 off)      ~ is Mic 4 off PGM?
  cmd ae1 surf1 s[St1 Mic 4] bus01 on                  ~ PGM on
  cmd ae1 surf1 s[St1 Mic 4] bus03 off                 ~ AX1 off
  cmd ae1 surf1 s[St1 Mic 4] bus00 off                 ~ channel off
 endif

 if state (ae1 scan surf1 s[St1 Telephone 1] bus01 off)     ~ is Tel 1 off PGM?
  cmd ae1 surf1 s[St1 Telephone 1] bus01 on           ~ PGM on
  cmd ae1 surf1 s[St1 Telephone 1] bus03 off          ~ AX1 off
  cmd ae1 surf1 s[St1 Telephone 1] bus00 off          ~ channel off
  cmd ae1 set surf1 mix- 4 bus1                       ~ set MM4 to PGM
  cmd ae1 surf1 s[St1 Telephone 1] text label "Rtn: PGM"
 endif

 if state (ae1 scan surf1 s[St1 Telephone 2] bus01 off)     ~ is Tel 2 off PGM?
  cmd ae1 surf1 s[St1 Telephone 2] bus01 on           ~ PGM on
  cmd ae1 surf1 s[St1 Telephone 2] bus03 off          ~ AX1 off
  cmd ae1 surf1 s[St1 Telephone 2] bus00 off          ~ channel off
  cmd ae1 set surf1 mix- 5 bus1                       ~ set MM5 to PGM
  cmd ae1 surf1 s[St1 Telephone 2] text label "Rtn: PGM"
 endif

 cmd recall ae1 surf1 d[Port1 Monitor In] from vMonSt1MonAmp                ~ recall Monitor Amp
 cmd recall ae1 surf1 d[Port1 Headphones In] from vMonSt1OprHP              ~ recall Opr Headphones
 cmd recall ae1 surf1 d[Port1 Studio In] from vMonSt1GstHP                  ~ recall Gst Headphones

endif
```

Route Select

Functionality Description

This function is used to select a route from a list. It is commonly used for record selection on a surface. The *Mosaic* surface **Wide Softkey** module contains two text screens which support up to 6 **Route Select** buttons. The *Numix* and *Remora* surfaces do not have button specifically designed for **Route Selects**, but do support the command.

Components

This function is contained within a single **Trigger**, however sets **If Accept** and **If Cancel** Conditional **Triggers**. The **Route Select** will:

- Set selection mode using a variable (the Source Device is stored to this variable).
- Turns the button lamp on.
- Labels the screen with the selection options.
- If accepted then perform the route and label the screen with the selected source.
- If cancelled or no action after 30 seconds then turn the lamp off.

If the same route is to be controlled elsewhere in the system, you will need some form of **Route Trigger** to update the text on the surface screen, otherwise they will be out of sync.

Further Information

- Bus On/Off – Page 55
- Route Select – Page 97

Script Examples

```
trigger ae1 d[Ctrl - Surf1 GPI In] bus81 on                                    ~ Mosaic Wide Softkey, Text Button 3

~=====================================^=====================================^==================================~
~ PROCEDURE: Route Select v1.0                                                 REVISED: 24th February 2005
~ DATA/ID: AE1 Port1 (StA) Wide Softkey TEXT3
~ DESCRIPTION: Turn on Route Select Mode
~ NOTES/DEPENDENCIES:
~ DEBUG STATUS:
~=====================================^================~====================^==================================~

     cmd ae1 remora vRoute_StA_MD1 set route selection mode d[StA MD1]         ~ turn on route select mode
     cmd ae1 d[Ctrl - Surf1 GPI Out] bus81 on                                  ~ turn on lamp 3

     cmd text route selection title "MD 1"               ~ display text title to route screen

     if accept selection text ae1 device28 line10 pos1   ~ display routed source to route screen
      cmd ae1 d[Ctrl - Surf1 GPI Out] bus81 off          ~ turn off lamp 3
     endif

     if cancel delay 30 then                             ~ after 30 seconds of inactivity...
      cmd ae1 d[Ctrl - Surf1 GPI Out] bus81 off          ~ turn off lamp 3
     endif
```

Mic Mute

Functionality Description

This function can be used to mute a microphone. The example uses a console button to activate, however, could easily be adapted to be triggered from an external GPI (eg external cough mute panel).

Components

This function has an ON (button pressed) and OFF (button released) **Trigger**.

Button On Trigger (pressed)

- Check if the Mic 1 fader is turned On (if Off, no more code will be executed).
- Turn Bus 8 (Cough Mute Bus) On for that source.
- Label the fader with the text "MUTE".
- Turn the button lamp on.

Button Off Trigger (released)

- Turn Bus 8 (Cough Mute Bus) Off.
- Clear the label from the Fader.
- Turn the lamp off.

Note the **Off Trigger** does not check to see if the fader is off, therefore will execute whenever the button is released. As it is just clearing the label and turning off the mute (Bus 8), there is no need to check if the fader is on.

Further Information

- Bus On/Off – Page 55
- If State Scan – Page 130
- Set Fader Label – Page 61

Script Examples

```
trigger ae1 d[Ctrl - Surf1 GPI In] bus86 on                                    ~ Mosaic Wide Softkey, Misc Button #2

~======================================^=======================================^=================================~
~ PROCEDURE: Cough Mute                                                        REVISED: 22nd February 2005
~ DATA/ID: AE1 Surf1 (StA) Mic 1 Mute ON
~ DESCRIPTION: Activate Mic Mute if Mute button pressed. Cancel Mic Mute if Mute button released.
~ NOTES/DEPENDENCIES: Activates/Deactivates Mute bus for Mic input.
~ DEBUG STATUS:
~======================================^================~======================^=================================~

if state scan ( ae1 surf1 s[StA Mic 1] bus0 on ) then
 cmd ae1 s[StA Mic 1] bus8 on                          ~ cough mute on
 cmd ae1 s[StA Mic 1] text label "  MUTE  "            ~ tally label to fader
 cmd ae1 d[Ctrl - Surf1 GPI Out] bus86 on             ~ mute lamp on
endif



trigger ae1 d[Ctrl - Surf1 GPI In] bus86 off                                   ~ Mosaic Wide Softkey, Misc Button #2

~======================================^=======================================^=================================~
~ PROCEDURE: Cough Mute                                                        REVISED: 22nd February 2005
~ DATA/ID: AE1 Surf1 (StA) Mic 1 Mute OFF
~ DESCRIPTION: Activate Mic Mute if Mute button pressed. Cancel Mic Mute if Mute button released.
~ NOTES/DEPENDENCIES: Activates/Deactivates Mute bus for Mic input.
~ DEBUG STATUS:
~======================================^================~======================^=================================~

cmd ae1 s[StA Mic 1] bus8 off                         ~ cough mute off
cmd ae1 s[StA Mic 1] text label " "                   ~ tally label to fader
cmd ae1 d[Ctrl - Surf1 GPI Out] bus86 off             ~ mute lamp off
```

12 x 1 Router

Functionality Description

This function uses a group of buttons to control a destination output, making the panel a 12x1 router. It could easily be adapted to any other number of buttons, e.g. to make the bottom row of a *COM12* panel a 6x1 router.

Components

This function uses an **On Trigger** when the button is pressed, to execute the command. A **Procedure** is used to turn all lamps in the group off, rather than repeat that code for each button. This makes it easier to change the buttons in the group in the future.

Button On Trigger

- Call procedure to turn OFF all lamps in group.
- Turn appropriate lamp ON.
- Route audio.

Procedure

- Turn OFF all lamps in group.

Further Information

- Call Procedure – Page 53
- Bus On/Off – Page 55
- Route – Page 59

Script Examples

```
trigger ae1 d[PA Monitor Amp] bus62 on

~=======================================^=======================================^==================================~
~ PROCEDURE: 12x1 Router                                                        REVISED: 23rd May 2005
~ DATA/ID: AE1 Port4 (PA) COM-12 Button 12
~ DESCRIPTION: Routes audio, lights tally lamps.
~ NOTES/DEPENDENCIES:
~ DEBUG STATUS:
~=======================================^================~=====================^==================================~

     call COM12_PA_LampsOff                                                     ~ procedure to turn off Lamps 7-12
     cmd ae1 d[PA Monitor Amp] bus42 on                                         ~ turn on lamp 12
     cmd ae1 route s[Sat Net NERR] to d[PA Monitor Amp]                         ~ routes NERR to PA Intercom

procedure COM12_PA_LampsOff

~=======================================^=======================================^==================================~
~ PROCEDURE: PA COM12 Lamps Off                                                 REVISED: 23rd May 2005
~ DATA/ID: PA - COM12 Lamps
~ DESCRIPTION: Turns off all lamps on Production Assistant COM12
~ NOTES/DEPENDENCIES:
~ DEBUG STATUS:
~=======================================^================~=====================^==================================~

      cmd ae1 d[PA Monitor Amp] bus31 off                                       ~ turn off lamp 1
      cmd ae1 d[PA Monitor Amp] bus32 off                                       ~ turn off lamp 2
      cmd ae1 d[PA Monitor Amp] bus33 off                                       ~ turn off lamp 3
      cmd ae1 d[PA Monitor Amp] bus34 off                                       ~ turn off lamp 4
      cmd ae1 d[PA Monitor Amp] bus35 off                                       ~ turn off lamp 5
      cmd ae1 d[PA Monitor Amp] bus36 off                                       ~ turn off lamp 6
      cmd ae1 d[PA Monitor Amp] bus37 off                                       ~ turn off lamp 7
      cmd ae1 d[PA Monitor Amp] bus38 off                                       ~ turn off lamp 8
      cmd ae1 d[PA Monitor Amp] bus39 off                                       ~ turn off lamp 9
      cmd ae1 d[PA Monitor Amp] bus40 off                                       ~ turn off lamp 10
      cmd ae1 d[PA Monitor Amp] bus41 off                                       ~ turn off lamp 11
      cmd ae1 d[PA Monitor Amp] bus42 off                                       ~ turn off lamp 12
```

Record Start with Tally

Functionality Description

This function is used to start a device recording. It utilizes a record tally back from the device to update the status of the button, which is a Record/Stop toggle. If the device is put into record locally, the button will illuminate and change into a stop function. Likewise, is the device is put into record from the remote button, but stopped locally, the lamp will turn off and button state reset.

Components

This function uses an **On Trigger** when the button is pressed, to execute the commands. GPI **Triggers** from the device update the lamp status and button state. When the record tally goes high, the lamp changes from flashing (set by the button) to solid, to indicate the recording did start.

Trigger Toggle State = 1 (Record Start)

- Send pulses to MiniDisc.
- Flash lamp in button.

Toggle State = 2 (Record Stop)

- Send pulses to MiniDisc.
- Turn off lamp in button.

GPI On (Tally High)

- Reset toggle state on button trigger (device has gone into record, change button to “stop”).
- Tally lamp ON (solid).

GPI Off (Tally Low)

- Reset toggle state on button trigger (device stopped recording, change button to “record”).
- Tally lamp OFF.

Further Information

- If Toggle – Page 126
- Relay Pulse – Page 70
- Lamp Flash – Page 57
- Bus On/Off – Page 55
- Set Toggle State – Page 74

Script Examples

```
trigger ae1 surf1 chan14 bus34 on toggle

~=====================================^=====================================^=================================~
~ PROCEDURE: Remote record with tally                                        REVISED: 27th April 2005
~ DATA/ID: MD1 Record - ON
~ DESCRIPTION: Sends Record GPI to MD1
~ NOTES/DEPENDENCIES: Relies on a record tally GPI from device to update lamp and toggle state
~ DEBUG STATUS:
~=====================================^================~====================^=================================~

    if toggle = 1 then                                  ~ do this if the button was off
     cmd ae1 d[Ctrl - Surf1 GPI Out] bus07 pulse        ~ set MiniDisc 1 to record (record GPI)
     cmd ae1 d[Ctrl - Surf1 GPI Out] bus05 pulse        ~ set MiniDisc 1 to record (play GPI)
     cmd ae1 surf1 chan13 bus34 flag flash continuous   ~ flash lamp
    endif

    if toggle = 2 then                                  ~ do this if the button was on
     cmd ae1 d[Ctrl - Surf1 GPI Out] bus06 pulse        ~ set MiniDisc 1 to stop (stop GPI)
     cmd ae1 surf1 chan13 bus34 off                     ~ turn off lamp
    endif



trigger ae1 d[Ctrl - Surf1 GPI In] bus07 on

~=====================================^=====================================^=================================~
~ PROCEDURE: Record Tally                                                    REVISED: 28th February 2005
~ DATA/ID: MD1 Record - ON
~ DESCRIPTION: Updates status of record button lamp
~ NOTES/DEPENDENCIES:
~ DEBUG STATUS:
~=====================================^================~====================^=================================~

     cmd set toggle=2 ae1 surf1 chan14 bus34 on         ~ set toggle state
     cmd ae1 surf1 chan13 bus34 on                      ~ turn lamp on



trigger ae1 d[Ctrl - Surf1 GPI In] bus07 off

~=====================================^=====================================^=================================~
~ PROCEDURE: Record Tally                                                    REVISED: 28th February 2005
~ DATA/ID: MD1 Record - OFF
~ DESCRIPTION: Updates status of record button lamp
~ NOTES/DEPENDENCIES:
~ DEBUG STATUS:
~=====================================^================~====================^=================================~

     cmd set toggle=1 ae1 surf1 chan14 bus34 on         ~ set toggle state
     cmd ae1 surf1 chan13 bus34 off                     ~ turn lamp off
```

Console Scenes

A “scene” is a way of quickly assigning certain sources to a set of faders. This is useful in many situations, such as a studio changeover from one program to another.

Functionality Description

Instead of the operator having to change the sources on every fader, a button can be programmed to complete a series of events in one “snapshot”. In the example given below, the commands written into this **Trigger** can reassign 6 faders with the push of just one button.

Components

A scene in usually programmed to a single button in a group, or from a menu activated by a button. In our example, a button within a group is used. This **On Trigger** will:

- Set a variable to track active scene (useful for other functions that reference current scene).
- Updates lamp group to turn selected scene lamp ON, and all others OFF.
- Clear Pending routes stored by previous scenes (so new pending routes can be set).
- Write text to display scene number and name to one of the surface screens.
- Turn PGM bus ON for all faders (except any faders that are turned ON).
- Perform specific routes for each fader, unless the route is already active. If the fader is currently ON, the WHEN OFF is used to store that route as PENDING).
- Set the Mix Minus bus to PGM on Telephone channels. Label these with `Rtn: PGM`.

Whilst this **Trigger** appears quite long, it is mostly just the same commands repeated for each fader, with the relevant sources for that fader.

Note: the check as to whether a source is already routed is not absolutely necessary. However, this prevents an on-air fader being labeled `PENDING` when it would not actually be changed.

Further Information

- Set Variable Equals - Page 124
- Bus On/Off – Page 55
- Clear When Off – Page 76
- Text – Chapter 13 (see commands for specific surface).
- If State / If State Not – Page 129
- When – Page 131
- Route – Page 59

Script Example

```
trigger ae1 d[Ctrl - Surf1 GPI In] bus32 on

~=====================================^=====================================^==================================~
~ PROCEDURE: Load Scene V2.4 PD                                              REVISED: 29th April 2005
~ DATA/ID: AE1 Port1 (StA) Wide Softkey Scene Button 1
~ DESCRIPTION: Loads scene to console. Delays fader change if it is on (pending). A1, A2 & A3 assignments unchanged.
~              PGM assignment unchanged if fader (BUS0) is turned on.
~ NOTES/DEPENDENCIES: Monitoring unchanged in scenes - delegation/delay does this.
~
~ DEBUG STATUS:
~=====================================^================~====================^==================================~

     cmd set vSceneStA = 1                                ~ update current scene

     cmd ae1 d[Ctrl - Surf1 GPI Out] bus32 on             ~ update Softkey Scene Lamp 1
     cmd ae1 d[Ctrl - Surf1 GPI Out] bus33 off            ~ update Softkey Scene Lamp 2
     cmd ae1 d[Ctrl - Surf1 GPI Out] bus34 off            ~ update Softkey Scene Lamp 3
     cmd ae1 d[Ctrl - Surf1 GPI Out] bus35 off            ~ update Softkey Scene Lamp 4
     cmd ae1 d[Ctrl - Surf1 GPI Out] bus36 off            ~ update Softkey Scene Lamp 5
     cmd ae1 d[Ctrl - Surf1 GPI Out] bus37 off            ~ update Softkey Scene Lamp 6

     cmd clear when ae1 surf1 chan01-10                   ~ clear when/pending - Faders 1 to 12

     cmd ae1 device2b text big line69 pos1 "SCENE: #1"                          ~ set text on Bridge screen 2
     cmd ae1 device2b text big line70 pos1 "Default"                            ~ set text on Bridge screen 2

     if state (ae1 surf1 chan01 bus0 off)
      cmd ae1 surf1 chan01 bus1 on                        ~ Fader 01 - PGM on (if channel is off)
     endif

     if state (ae1 surf1 chan02 bus0 off)
      cmd ae1 surf1 chan02 bus1 on                        ~ Fader 02 - PGM on (if channel is off)
     endif

     if state (ae1 surf1 chan03 bus0 off)
      cmd ae1 surf1 chan03 bus1 on                        ~ Fader 03 - PGM on (if channel is off)
     endif

     if state (ae1 surf1 chan04 bus0 off)
      cmd ae1 surf1 chan04 bus1 on                        ~ Fader 04 - PGM on (if channel is off)
     endif

     if state (ae1 surf1 chan05 bus0 off)
      cmd ae1 surf1 chan05 bus1 on                        ~ Fader 05 - PGM on (if channel is off)
     endif

     if state (ae1 surf1 chan06 bus0 off)
      cmd ae1 surf1 chan06 bus1 on                        ~ Fader 06 - PGM on (if channel is off)
     endif

     if state (ae1 surf1 chan07 bus0 off)
      cmd ae1 surf1 chan07 bus1 on                        ~ Fader 07 - PGM on (if channel is off)
     endif

     if state (ae1 surf1 chan08 bus0 off)
      cmd ae1 surf1 chan08 bus1 on                        ~ Fader 08 - PGM on (if channel is off)
     endif

     if state (ae1 surf1 chan09 bus0 off)
      cmd ae1 surf1 chan09 bus1 on                        ~ Fader 09 - PGM on (if channel is off)
     endif

     if state (ae1 surf1 chan10 bus0 off)
      cmd ae1 surf1 chan10 bus1 on                        ~ Fader 10 - PGM on (if channel is off)
     endif
```

```
if state not (ae1 surf1 chan01 route device s[Sat Net])
 when ae1 surf1 chan01 off
  cmd ae1 route surf1 chan01 device s[Sat Net]                    ~ Fader 01 - Network (if not already selected)
 endwhen
endif

if state not (ae1 surf1 chan02 route device s[Ed2 PGM St *N*])
 when ae1 surf1 chan02 off
  cmd ae1 route surf1 chan02 device s[Ed2 PGM St *N*]             ~ Fader 02 - Edit 2 PGM (if not already
            selected)
 endwhen
endif

if state not (ae1 surf1 chan03 route device s[StA CD1])
 when ae1 surf1 chan03 off
  cmd ae1 route surf1 chan03 device s[StA CD1]                    ~ Fader 03 - CD 1 (if not already selected)
 endwhen
endif

if state not (ae1 surf1 chan04 route device s[StA CD2])
 when ae1 surf1 chan04 off
  cmd ae1 route surf1 chan04 device s[StA CD2]                    ~ Fader 04 - CD 2 (if not already selected)
 endwhen
endif

if state not (ae1 surf1 chan05 route device s[StA MD1])
 when ae1 surf1 chan05 off
  cmd ae1 route surf1 chan05 device s[StA MD1]                    ~ Fader 05 - MD 1 (if not already selected)
 endwhen
endif

if state not (ae1 surf1 chan06 route device s[StA Mic 1])
 when ae1 surf1 chan06 off
  cmd ae1 route surf1 chan06 device s[StA Mic 1]                  ~ Fader 06 - Mic 1 (if not already selected)
 endwhen
endif

if state not (ae1 surf1 chan07 route device s[StA Mic 2])
 when ae1 surf1 chan07 off
  cmd ae1 route surf1 chan07 device s[StA Mic 2]                  ~ Fader 07 - Mic 2 (if not already selected)
 endwhen
endif

if state not (ae1 surf1 chan08 route device s[StA Mic 3])
 when ae1 surf1 chan08 off
  cmd ae1 route surf1 chan08 device s[StA Mic 3]                  ~ Fader 08 - Mic 3 (if not already selected)
 endwhen
endif

if state not (ae1 surf1 chan09 route device s[StA DAW 1])
 when ae1 surf1 chan09 off
  cmd ae1 route surf1 chan09 device s[StA DAW 1]                  ~ Fader 09 - DAW 1 (if not already selected)
 endwhen
endif

~ no "if state not" for telephone - we want this to run every time to update the MM, even if already selected
when ae1 surf1 chan10 off
 cmd ae1 route surf1 chan11 device s[StA Telephone 1]             ~ Fader 10 - Telephone 1 (if not already)
 cmd ae1 set surf1 mix- 4 bus1                                    ~ set MM4 to PGM
 cmd ae1 surf1 chan10 text label "Rtn: PGM"
endwhen
```

Mosaic Monitor Hotkeys

Functionality Description

This function is designed for the *Mosaic's* **Monitor** hotkeys. It routes the selected source and updates the lamp statuses.

In addition, we have a function that will respond to a route change and update the *Mosaic* hotkey button lamps. This is necessary in case the route was changed externally (e.g. by another Trigger).

Components

This function has an On Trigger (when the button is pressed), plus a **Procedure** and a **Route Trigger**. The latter two allow for a source to be changed externally, with the appropriate tally.

Button On Trigger

- The button on Trigger performs the route.

Procedure

- The Procedure is used to check the existing route, and update the lamps if the route matches a known preset button.
- This is called from a Route Trigger, or any other Trigger that updates the monitor routing (remember, Triggers do not cascade, so the Route Trigger will be called if we do a route from inside another Trigger).

Route Trigger

- Calls the procedure to check the current route.
- Note, this Trigger will not execute if a Trigger performs a route to this destination (prevents endless loops).

Further Information

- Bus On/Off – Page 55
- Input Route – Page 59
- Call Procedure – Page 51
- If State – Page 129

Notes

The *Mosaic* has a function called "Follow Monitor" for the **Headphones** and **Guest/Studio** sends. This function is automatically turned off when a **Monitor Hotkey** is updated from a **Trigger**. An improvement to this function would first check whether "Follow Mon" is selected, and if so, ignore the change.

Script Examples

```
trigger ae1 surf1 chan27 bus16 on                                                ~ Mosaic Monitor Hotkey #1

~=====================================^=====================================^=================================~
~ PROCEDURE: Monitor Routing                                                     REVISED: 22nd February 2005
~ DATA/ID: AE1 Port1 (StA) Phones USER 1
~ DESCRIPTION: If on, route source to announcer headphones.
~ NOTES/DEPENDENCIES:
~ DEBUG STATUS:
~=====================================^================~====================^=================================~

     cmd route ae1 s[Port1 Program Out] to d[Port1 Headphones In]                ~ routes PGM to announcer h/p


trigger ae1 d[Port1 Monitor In] route any

~=====================================^=====================================^=================================~
~ PROCEDURE: Monitor Route Change with Tally                                     REVISED: 27th April 2005
~ DATA/ID: AE1 Port 1
~ DESCRIPTION: Updates Monitor Hot Key Lamps.
~ NOTES/DEPENDENCIES:
~ DEBUG STATUS:
~=====================================^================~====================^=================================~

     call MosaicMonitor_StA_Monitor                        ~ Call Monitor Tally (external route was performed)


procedure MosaicMonitor_StA_Monitor

~=====================================^=====================================^=================================~
~ PROCEDURE: Mosaic Monitor Hotkey Update                                        REVISED: 27th April 2005
~ DATA/ID: StA - Monitor
~ DESCRIPTION: Updates monitor hotkey lamps based on current route
~ NOTES/DEPENDENCIES:
~ DEBUG STATUS:
~=====================================^================~====================^=================================~

     cmd ae1 d[Port1 Monitor In] bus16 off                ~ turn off lamps for JetStream Server
     cmd ae1 d[Port1 Monitor In] bus17 off                ~ turn off lamps for JetStream Server
     cmd ae1 d[Port1 Monitor In] bus18 off                ~ turn off lamps for Supervisor
     cmd ae1 d[Port1 Monitor In] bus19 off                ~ turn off lamps for Supervisor
     cmd ae1 d[Port1 Monitor In] bus20 off                ~ turn off lamps for Supervisor

     if state = ( ae1 route s[Port1 Program Out] to d[Port1 Monitor In] ) then   ~ Hotkey Source 1 is selected
      cmd ae1 d[Port1 Monitor In] bus16 on                                       ~ Turn lamp 1 on
     endif

     if state = ( ae1 route s[Port1 Aux 1 Out] to d[Port1 Monitor In] ) then     ~ Hotkey Source 2 is selected
      cmd ae1 d[Port1 Monitor In] bus17 on                                       ~ Turn lamp 2 on
     endif

     if state = ( ae1 route s[Port3 CP1 TXA Stereo] to d[Port1 Monitor In] ) then~ Hotkey Source 3 is selected
      cmd ae1 d[Port1 Monitor In] bus18 on                                       ~ Turn lamp 3 on
     endif

     if state = ( ae1 route s[5MV 1062 Rx] to d[Port1 Monitor In] ) then         ~ Hotkey Source 4 is selected
      cmd ae1 d[Port1 Monitor In] bus19 on                                       ~ Turn lamp 4 on
     endif

     if state = ( ae1 route s[Port1 Delay CP1 Out] to d[Port1 Monitor In] ) then ~ Hotkey Source 5 is selected
      cmd ae1 d[Port1 Monitor In] bus20 on                                       ~ Turn lamp 5 on
     endif
```

20 Delay Control Examples

CommandBuilder can be used to build sophisticated delay control logic to assist operators get in and out of delay. The examples in this chapter us the **Logitek** *SharcAttack* internal delay, however could be easily adapted to external profanity delay units, using GPI control.

There are many different ways to enter and exit delay. These examples show methods that have been used at different stations around the world.

Delay Start

Functionality Description

This function starts the internal **Talk Delay**.

Components

The Delay Start button has an **On Trigger**, which performs these tasks, plus a **Procedure** to update display tallies.

Button On Trigger

- Check the value of a variable (if = 0, then continue). This prevents going into delay again.
- Disables Post Swap mode, turn off Post Swap Lamp and reset toggle (see later in this chapter).
- Send Delay Off command to Talk Delay (just to be sure).
- Route Program to Monitor Amp and Headphones.
- Call procedure to update Mosaic monitor hotkey lamps (see Chapter 19 for more info).
- Send Delay On command to Talk Delay.
- Call a procedure a to update lamps and text.
- Turn on “Dump” lamp.
- Update variables to track status.

Tally Procedure

- Turns on appropriate lamps in the studio.
- Writes text to screen.
- Can also be used to display status lamps elsewhere in the facility.

Further Information

- If Variable – Page 127
- Set Variable – Page 124
- Bus On/Off – Page 55
- Set Toggle State – Page 74
- Talk Delay Off/Start – Page 72
- Route – Page 59
- Call Procedure – Page 53

- Text – Chapter 13.

Notes

This function uses a variable to track delay status. Because of this, the *vDelay* application should not be used to activate the delay, or the studio functions will not be in sync. However, you may choose not to use variables, in which case *vDelay* can be used (but the lamps in the studio will still be out of sync if *vDelay* starts or stops the delay).

Script Examples

```
trigger ae1 surf1 chan14 bus40 on                                                ~ Mosaic Narr Softkey, Button 9

~=====================================^=====================================^=================================~
~ PROCEDURE: Delay Start v2.0 (SharcAttack)                                      REVISED: 27th April 2005
~ DATA/ID: AE1 Port1 (StA) Narr Softkey BIG9
~ DESCRIPTION: If off, send start to SharcAttack Delay. If on, ignore button press.
~ NOTES/DEPENDENCIES: Variable status determines whether we are in delay or not.
~ DEBUG STATUS:
~=====================================^================~====================^=================================~

     if vDelayStAOn = 0                                  ~ no-one is on delay
      cmd set vDelayStAPostSwap = 0                      ~ set post swap mode off
      cmd ae1 d[Ctrl - Surf1 GPI Out] bus87 off          ~ lamp FUNC3 off
      cmd set toggle = 1 ( ae1 d[Ctrl - Surf1 GPI In] bus87 )                    ~ reset toggle to off state
      cmd ae1 d[Port1 Delay CP1 In] talk off                                     ~ force off to clear delay (PGM)
      cmd route ae1 s[Port1 Program Out] to d[Port1 Monitor In]                  ~ route Program to Monitor Amp
      cmd route ae1 s[Port1 Program Out] to d[Port1 Headphones In]               ~ route Program to Opr Headphones
      call MosaicMonitor_StA_Monitor                     ~ update monitor hotkey lamps
      call MosaicMonitor_StA_Headphones                  ~ update monitor hotkey lamps
      cmd ae1 d[Port1 Delay CP1 In] talk start           ~ start delay (PGM)
      call DelayTally_StA_On                             ~ update lamps and text
      cmd set vDelayStAOn = 1                            ~ update variable
      cmd set vDelayStASafe = 0                          ~ update variable
     endif


procedure DelayTally_StA_On

~=====================================^=====================================^=================================~
~ PROCEDURE: Delay Text Tally                                                    REVISED: 21st February 2005
~ DATA/ID: Studio A (StA) - Delay On
~ DESCRIPTION: Updates delay status on Mosaic surfaces.
~ NOTES/DEPENDENCIES:
~ DEBUG STATUS:
~=====================================^================~====================^=================================~

     cmd ae1 surf1 chan13 bus40 on                      ~ turn on "Delay On" lamp in this studio
     cmd ae1 surf1 chan13 bus43 on                      ~ turn on "Dump" lamp in this studio
     cmd ae1 surf1 chan13 bus41 off                     ~ turn off "Delay Off" lamp in this studio

     cmd ae1 device2C line135 text pos33 "     Delay ON   "
```

Delay Dump

Functionality Description

This function dumps the Talk Delay chain, but keeps Talk Delay ON. The delay will then rebuild.

Components

The Delay Dump button has an **On Trigger**, which performs these tasks, plus an **Off Trigger**, which handles status updates when the button is released.

Button On Trigger

- Check the value of status variable. This prevents dumping when not in delay.
- Send Dump command twice to dump full delay (2 lots of 4 seconds).
- Write text to screen.

Button Off Trigger

- Sets a timer that will clear the text display 3 seconds after the button is released.

Further Information

- If Variable – Page 127
- Talk Delay Dump – Page 72
- Text – Chapter 13
- If Timer – Page 48

Script Examples

```
trigger ae1 surf1 chan14 bus43 on                                          ~ Mosaic Narr Softkey, Butt 12 (press)

~=====================================^=====================================^==================================~
~ PROCEDURE: Delay Dump v2.0 (SharcAttack)                                 REVISED: 22nd February 2005
~ DATA/ID: AE1 Port1 (St1) Narrow Softkey BIG 12
~ DESCRIPTION: If on, send a "dump" to delay. If off, do nothing.
~ NOTES/DEPENDENCIES: Variable status determines whether we are in delay or not.
~ DEBUG STATUS:
~=====================================^===============~====================^==================================~

     if vDelayStAOn = 1                                ~ we are on delay at the moment
      cmd ae1 d[Port1 Delay CP1 In] talk dump           ~ send dump to delay (PGM)
      cmd ae1 d[Port1 Delay CP1 In] talk dump           ~ send dump to delay (PGM)
      cmd ae1 device2C line137 text pos49 " ***Delay DUMP***"
     endif

trigger ae1 surf1 chan14 bus43 off                                         ~ Mosaic Narr Softkey Butt 12 (release)

~=====================================^=====================================^==================================~
~ PROCEDURE: Delay Dump v2.0 (SharcAttack)                                 REVISED: 22nd February 2005
~ DATA/ID: AE1 Port1 (StA) Narrow Softkey BIG 12
~ DESCRIPTION: If on, send a "dump" to delay. If off, do nothing.
~ NOTES/DEPENDENCIES: Variable status determines whether we are in delay or not.
~ DEBUG STATUS:
~=====================================^===============~====================^==================================~

     if timer 112 wait 3 then
      cmd ae1 device2C line137 text " "
     endif
```

Post Monitor

Functionality Description

This function allows the operator to switch to post-delay monitoring temporarily to verify they are correctly in delay. The button is a toggle function, and will flash while active. This example also updates the *Mosaic's* hotkey buttons (this can be omitted on *Remora* and *Numix* surfaces).

Components

The Post Mon has an **On Toggle Trigger**, which performs these tasks.

Button On Toggle Trigger (State 1)

- Checks toggle state (1 = normal, 2 = toggle).
- Updates variable (optional) – this may be referenced by other triggers.
- Stores current Monitor and Headphones routes to variables.
- Routes Post Delay Crosspoint output to Monitor and Headphones.
- Flashes the Post Monitor button lamp.
- Call procedure to update Mosaic monitor hotkey lamps (see Chapter 19 for more info).

Button On Toggle Trigger (State 2)

- Checks toggle state (1 = normal, 2 = toggle).
- Updates variable (optional) – this may be referenced by other triggers.
- Restores routes to Monitor and Headphones from variables.
- Turns the Post Monitor button lamp off.
- Call procedure to update Mosaic monitor hotkey lamps (see Chapter 19 for more info).

Further Information

- If Toggle – Page 126
- Set Variable – Page 124
- Device Store/Recall – Page 124
- Lamp Flash – Page 57
- Call Procedure – Page 53

Notes

If you use this function, you need to consider the implications of changing the monitor sources from other Triggers. For example, our Delay On and Delay Exit Triggers will update the monitor source, and therefore these Triggers also update the toggle state and button lamp.

These updates ensure the operation of the console is consistent and predictable, and avoids operator confusion.

Script Examples

```
trigger ae1 d[Ctrl - Surf1 GPI In] bus87 on toggle                              ~ Mosaic Wide Softkey Function Button 3

~=====================================^=====================================^==================================~
~ PROCEDURE: Pre/Post swap                                                     REVISED: 27th April 2005
~ DATA/ID: AE1 Port1 (StA) Wide Softkey FUNC3
~ DESCRIPTION: Swaps delay between pre/post monitoring. Swaps between prior setting, and "PostDelay".
~ NOTES/DEPENDENCIES: Toggle and restore can also be done in the delay "direct" trigger.
~ DEBUG STATUS:
~=====================================^================~====================^==================================~

    if toggle = 1                                       ~ normal state
     cmd set vDelayStAPostSwap = 1                      ~ set post swap mode
     cmd store ae1 d[Port1 Monitor In] route to vMonStAMonAmp                   ~ store Monitor Amp source device
     cmd store ae1 d[Port1 Headphones In] route to vMonStAOprHP                 ~ store Opr Headphones source device
     cmd route ae1 s[Port1 Delay CP1 Out] to d[Port1 Monitor In]                ~ route post delay to Monitor Amp
     cmd route ae1 s[Port1 Delay CP1 Out] to d[Port1 Headphones In]             ~ route post delay to Opr Headphones
     cmd ae1 d[Ctrl - Surf1 GPI Out] bus87 flag flash continuous               ~ flash lamp FUNC3
     call MosaicMonitor_StA_Monitor                     ~ update monitor hotkey lamps
     call MosaicMonitor_StA_Headphones                  ~ update monitor hotkey lamps
    endif

    if toggle = 2                                       ~ toggled state
     cmd set vDelayStAPostSwap = 0                      ~ set post swap mode off
     cmd recall ae1 d[Port1 Monitor In] from vMonStAMonAmp                      ~ recall Monitor Amp
     cmd recall ae1 d[Port1 Headphones In] from vMonStAOprHP                    ~ recall Opr Headphones
     cmd ae1 d[Ctrl - Surf1 GPI Out] bus87 off          ~ lamp FUNC3 off
     call MosaicMonitor_StA_Monitor                     ~ update monitor hotkey lamps
     call MosaicMonitor_StA_Headphones                  ~ update monitor hotkey lamps
    endif
```

Delay Exit – Method A (Timer Based)

Functionality Description

This function turns off Talk Delay. When the operator presses "Delay Off", their monitoring is changed to Post Delay and a timer is set based on the current delay time. The Delay Off button lamp will flash, and once the delay has emptied the delay is turned off. At this time, the lamps and text are also updated.

This method provides a fairly simple control method for operators.

Components

The Delay Exit button has an **On Trigger**, which performs these tasks, plus two tally **Procedures** to update display tallies.

Button On Trigger

- Check the value of a variable (if = 1, then continue). This prevents exiting delay again.
- Set toggle state on Post Swap button (see Post Swap function above).
- Update Post Swap variable (optional) – this may be referenced by other triggers.
- Store current Monitor and Headphones routes to variables.
- Route Post Delay Crosspoint output to Monitor and Headphones.
- Call procedure to update *Mosaic* monitor hotkey lamps (see Chapter 19 for more info).
- Flash the Post Monitor button lamp.
- Set timer based on current delay time. When that time expires:
    - Call a procedure a to update lamps and text.
    - Update variables to track status.
    - Send Delay Off command to Talk Delay.

Tally Procedures

- Turn on appropriate lamps in the studio.
- Write text to screen.
- Can also be used to display status lamps elsewhere in the facility.
- Send assembler command to active fast lamp flash (to be replaced by Fast Flash command).

Further Information

- If Variable – Page 127
- Call Procedure – Page 53
- Set Toggle State – Page 74
- Device Store/Recall – Page 124
- Route – Page 59
- Lamp Flash – Page 57
- If Timer – Page 48
- Talk Delay Off – Page 72

Script Examples

```
trigger ae1 surf1 chan14 bus41 on                                              ~ Mosaic Narr Softkey Button 10

~=====================================^=====================================^=================================~
~ PROCEDURE: Delay Timer Exit v2.0 (SharcAttack)                               REVISED: 27th April 2005
~ DATA/ID: AE1 Port1 (StA) Narrow Softkey Big Button 10
~ DESCRIPTION: Change monitoring to Post Delay. Set timer to switch off delay when delay time expires.
~ NOTES/DEPENDENCIES: Variable status determines whether we are in delay or not.
~ DEBUG STATUS:
~=====================================^================~====================^=================================~

     if vDelayStAOn = 1                                ~ we are on delay at the moment
      call DelayTally_StA_Exit                         ~ update lamps and text
      cmd set toggle = 2 ( ae1 d[Ctrl - Surf1 GPI In] bus87 )                  ~ reset toggle to on state
      cmd set vDelayStAPostSwap = 1                    ~ set post swap mode
      cmd store ae1 d[Port1 Monitor In] route to vMonStAMonAmp                  ~ store Monitor Amp source device
      cmd store ae1 d[Port1 Headphones In] route to vMonStAOprHP                ~ store Opr Headphones source device
      cmd route ae1 s[Port1 Delay CP1 Out] to d[Port1 Monitor In]               ~ route post delay to Monitor Amp
      cmd route ae1 s[Port1 Delay CP1 Out] to d[Port1 Headphones In]            ~ route post delay to Opr Headphones
      call MosaicMonitor_StA_Monitor                   ~ update monitor hotkey lamps
      call MosaicMonitor_StA_Headphones                ~ update monitor hotkey lamps
      cmd ae1 d[Ctrl - Surf1 GPI Out] bus87 flag flash continuous              ~ flash lamp FUNC3

      if ae1 timer 111 talk d[Port1 Delay CP1 In] then
       call DelayTally_StA_Off                         ~ update lamps and text
       cmd set vDelayStAOn = 0                         ~ update variable
       cmd set vDelayStASafe = 0                       ~ update variable
       cmd ae1 d[Port1 Delay CP1 In] talk off          ~ force off to clear delay (PGM)
      endif
     endif



procedure DelayTally_StA_Exit

~=====================================^=====================================^=================================~
~ PROCEDURE: Delay Text Tally                                                  REVISED: 21st February 2005
~ DATA/ID: Studio A (StA) - Delay Exit
~ DESCRIPTION: Updates delay status on Mosaic surfaces.
~ NOTES/DEPENDENCIES:
~ DEBUG STATUS:
~=====================================^================~====================^=================================~

     cmd ae1 surf1 chan13 bus40 off                    ~ turn off "Delay On" lamp in this studio
     cmd ae1 asm  "02 04 B1 17 29 05"                  ~ fast flash "Delay Off" lamp in this studio
     cmd ae1 surf1 chan13 bus43 off                    ~ turn off "Dump" lamp in this studio

     cmd ae1 device2C line135 text pos49 "     Delay EXIT  "



procedure DelayTally_StA_Off

~=====================================^=====================================^=================================~
~ PROCEDURE: Delay Text Tally                                                  REVISED: 21st February 2005
~ DATA/ID: Studio A (StA) - Delay Off
~ DESCRIPTION: Updates delay status on Mosaic surfaces.
~ NOTES/DEPENDENCIES:
~ DEBUG STATUS:
~=====================================^================~====================^=================================~

     cmd ae1 surf1 chan13 bus41 on                    ~ turn on "Delay Off" lamp in this studio

     cmd ae1 device2C line135 text pos97 "     Delay OFF  "
```

Delay Exit – Method B (Hard Cut)

Functionality Description

This function turns off Talk Delay. When the operator wishes to exit delay, they will first press the "Post Mon" button to hear the delay empty. At the appropriate point in time, they press the "Delay Off". When the operator presses "Delay Off" button, the delay is instantly killed and they are live. At this time, the lamps and text are also updated.

This method provides a fairly simple control method for operators.

Components

The Delay Exit button has an **On Trigger**, which performs these tasks, plus a **Procedure** to update display tallies.

Button On Trigger

- Check the value of a variable (if = 1, then continue). This prevents exiting delay again.
- Call a procedure a to update lamps and text.
- Update variables to track status.
- Send Delay Off command to Talk Delay.

Tally Procedures

- Turns on appropriate lamps in the studio.
- Writes text to screen.
- Can also be used to display status lamps elsewhere in the facility.

Further Information

- If Variable – Page 127
- Call Procedure – Page 53
- Talk Delay Off – Page 72

Script Examples

```
trigger ae1 surf1 chan14 bus41 on                                          ~ Mosaic Narr Softkey Button 10

~=====================================^=====================================^=================================~
~ PROCEDURE: Delay Hard Exit v2.0 (SharcAttack)                             REVISED: 27th April 2005
~ DATA/ID: AE1 Port1 (StA) Narrow Softkey Big Button 10
~ DESCRIPTION: Switch off delay immediately and update tallies.
~ NOTES/DEPENDENCIES: Variable status determines whether we are in delay or not.
~ DEBUG STATUS:
~=====================================^================~====================^=================================~

     if vDelayStAOn = 1                                  ~ we are on delay at the moment
      call DelayTally_StA_Off                            ~ update lamps and text
      cmd set vDelayStAOn = 0                            ~ update variable
      cmd set vDelayStASafe = 0                          ~ update variable
      cmd ae1 d[Port1 Delay CP1 In] talk off             ~ force off to clear delay (PGM)
     endif



procedure DelayTally_StA_Off

~=====================================^=====================================^=================================~
~ PROCEDURE: Delay Text Tally                                               REVISED: 21st February 2005
~ DATA/ID: Studio A (StA) - Delay Off
~ DESCRIPTION: Updates delay status on Mosaic surfaces.
~ NOTES/DEPENDENCIES:
~ DEBUG STATUS:
~=====================================^================~====================^=================================~

     cmd ae1 surf1 chan13 bus41 on                       ~ turn on "Delay Off" lamp in this studio

     cmd ae1 device2C line135 text pos97 "     Delay OFF  "
```

Delay Exit – Method C (Next Event)

Functionality Description

This function turns off Talk Delay. When the operator presses "Delay Off", monitoring is changed to Post Delay and tallies are updated to show "Exit Ready". This arms all of the fader ON buttons, and the next ON button that is pressed will activate the cut to live program. This method is well suited to stations that will fire an event (eg CD track or hard-disk replay) from a GPI activated source. It provides a very tight and definite cut-over point, whilst still being easy to use.

Components

The Delay Exit button has an **On Trigger**, which performs tally & status updates, and a **Procedure** to exit delay. In addition, a **Trigger** is required for every fader BUS0 ON event on that surface. These will check the status of the delay variable, and execute the exit procedure if required.

Button On Trigger

- Check the value of a variable (if = 1, then continue). This prevents exiting delay again.
- Update delay status variable (2 = delay exit ready) – this "arms" the fader exit routine.
- Set toggle state on Post Swap button (see Post Swap function above).
- Update Post Swap variable (optional) – this may be referenced by other triggers.
- Store current Monitor and Headphones routes to variables.
- Route Post Delay Crosspoint output to Monitor and Headphones.
- Call procedure to update Mosaic monitor hotkey lamps (see Chapter 19 for more info).
- Flash the Post Monitor button lamp.

Exit/Tally Procedure

- Sends off command to delay.
- Turns on appropriate lamps in the studio.
- Writes text to screen.

Fader On Triggers (one per fader on surface)

- Turns off CUE & TB on that channel (unrelated optional extra).
- If delay status = 2 then call procedure to exit.

Further Information

- If Variable – Page 127
- Call Procedure – Page 53
- Set Toggle State – Page 74
- Set Variable – Page 124
- Device Store/Recall – Page 124
- Lamp Flash – Page 57
- Talk Delay Off – Page 72
- Bus On/Off – Page 55

Script Examples

```
trigger ae1 surf1 chan14 bus40 on                                          ~ Mosaic Narr Softkey, Button 10

~=====================================^=====================================^==================================~
~ PROCEDURE: Delay Exit Next Event v2.0 (SharcAttack)                      REVISED: 27th April 2005
~ DATA/ID: AE1 Port1 (StA) Narrow Softkey BIG10
~ DESCRIPTION: If on, enter exit mode (flash lamp). Next event triggers Delay Off GPI.
~ NOTES/DEPENDENCIES: Variable status determines whether we are in delay or not.
~ DEBUG STATUS:
~=====================================^================~====================^==================================~

     if vDelayStAOn = 1                                ~ we are on delay at the moment
      cmd set vDelStAOn = 2                            ~ update delay status to exit ready (2)
      call DelayTally_StA_Exit                         ~ update lamps and text
      cmd set toggle = 2 ( ae1 d[Ctrl - Surf1 GPI In] bus87 )          ~ reset toggle to on state
      cmd set vDelayStAPostSwap = 1                    ~ set post swap mode
      cmd store ae1 d[Port1 Monitor In] route to vMonStAMonAmp         ~ store Monitor Amp source device
      cmd store ae1 d[Port1 Headphones In] route to vMonStAOprHP       ~ store Opr Headphones source device
      cmd route ae1 s[Port1 Delay CP1 Out] to d[Port1 Monitor In]      ~ route post delay to Monitor Amp
      cmd route ae1 s[Port1 Delay CP1 Out] to d[Port1 Headphones In]   ~ route post delay to Opr Headphones
      call MosaicMonitor_StA_Monitor                   ~ update monitor hotkey lamps
      call MosaicMonitor_StA_Headphones                ~ update monitor hotkey lamps
      cmd ae1 d[Ctrl - Surf1 GPI Out] bus87 flag flash continuous      ~ flash lamp FUNC3
     endif



procedure DelOffDelay_StA

~=====================================^=====================================^==================================~
~ PROCEDURE: Send Delay Off Commands & Text Tally                          REVISED: 27th April 2005
~ DATA/ID: StA - Delay Off
~ DESCRIPTION: Sends off commands to delay. Updates delay status on Mosaic surfaces
~ NOTES/DEPENDENCIES:
~ DEBUG STATUS:
~=====================================^================~====================^==================================~

     cmd set vDelayStAOn = 0                       ~ update variable
     cmd set vDelayStASafe = 0                     ~ update variable
     cmd ae1 d[Port1 Delay CP1 In] talk off        ~ force off to clear delay (PGM)
     cmd ae1 surf1 chan13 bus41 on                 ~ turn on "Delay Off" lamp in this studio
     cmd ae1 device2C line135 text pos97 "     Delay OFF  "



trigger ae1 surf1 chan01 bus00 on

~=====================================^=====================================^==================================~
~ PROCEDURE: Kill Cue & TB / Delay Exit on next                            REVISED: 27th April 2005
~ DATA/ID: AE1 Port1 (StA) Fader 1
~ DESCRIPTION: Turns off Cue bus (BUS02) and TB bus (BUS14) on fader when fader goes on (BUS00)
~              Also checks for Delay "Exit Ready" status, and runs delay off procedure if necessary.
~ NOTES/DEPENDENCIES:
~ DEBUG STATUS:
~=====================================^================~====================^==================================~

     cmd ae1 surf1 chan01 bus02 off
     cmd ae1 surf1 chan01 bus14 off

     if vDelSt1OnDelay = 2
      call DelOffDelay_StA
     endif

~ Repeat this fader trigger for every fader on surface!!
```

Delay Exit – Method D (Ramp Down)

Functionality Description

This function turns off Talk Delay. When the operator presses "Delay Exit", the delay will begin to ramp down. This may take a number of minutes. The delay time will eventually reach 0.0, but the display will stay on. Monitoring will remain on pre-delay during this time.

This method provides simple operation, however does not provide a quick exit. It would usually be offered in conjunction with another form of quick exit (Methods A, B, or C).

Components

The Delay Exit button has an **On Trigger**, which performs these tasks, plus a **Procedure** to update display tallies.

Button On Trigger

- Check the value of a variable (if = 1, then continue). This prevents exiting delay again.
- Call a procedure a to update lamps and text.
- Update variables to track status.
- Send Delay Exit command to Talk Delay.

Tally Procedures

- Turns on appropriate lamps in the studio.
- Writes text to screen.
- Can also be used to display status lamps elsewhere in the facility.

Further Information

- If Variable – Page 127
- Call Procedure – Page 53
- Talk Delay Off – Page 72

Script Examples

```
trigger ae1 surf1 chan14 bus41 on                                              ~ Mosaic Narr Softkey Button 10

~======================================^=======================================^=================================~
~ PROCEDURE: Delay Ramp Down v2.0 (SharcAttack)                                 REVISED: 27th April 2005
~ DATA/ID: AE1 Port1 (StA) Narrow Softkey Big Button 10
~ DESCRIPTION: Switches delay into ramp down and update tallies.
~ NOTES/DEPENDENCIES: Variable status determines whether we are in delay or not.
~ DEBUG STATUS:
~======================================^================~====================^=================================~

     if vDelayStAOn = 1                                 ~ we are on delay at the moment
      call DelayTally_StA_Off                           ~ update lamps and text
      cmd set vDelayStAOn = 2                           ~ update variable
      cmd set vDelayStASafe = 0                         ~ update variable
      cmd ae1 d[Port1 Delay CP1 In] talk stop           ~ force ramp down to exit delay (PGM)
     endif



procedure DelayTally_StA_Off

~======================================^=======================================^=================================~
~ PROCEDURE: Delay Text Tally                                                   REVISED: 21st February 2005
~ DATA/ID: Studio A (StA) - Delay Off
~ DESCRIPTION: Updates delay status on Mosaic surfaces.
~ NOTES/DEPENDENCIES:
~ DEBUG STATUS:
~======================================^================~====================^=================================~

     cmd ae1 surf1 chan13 bus41 on                      ~ turn on "Delay Off" lamp in this studio

     cmd ae1 device2C line135 text pos97 "     Delay EXIT "
```

21 Intercom Examples

You can use your **Logitek** system to create sophisticated Intercoms between studios. Setting up an Intercom is as simple as routing the appropriate source (e.g. a microphone) to the **Talkback Rtn** input of the appropriate surface. However, with a little more programming you can create a more powerful version that your operators will love to use.

Logitek Intercom with calling station display

Functionality Description

This Intercom system uses the internal **Logitek Fiber Network** and **Triggers** to create a slick user interface. It is designed to make use of fibre-linked **Audio Engines**, with all possible Intercom mics already shared across all **Engines**. If not, you need to set this up in *AEConfig* first.

It can also interface with external units, although for best results each external station should provide a GPI closure to indicate each destination it calls. If this is not possible, you can use the **External Cue** input on the surface – however, without a GPI interface external stations will not be able to activate Monitor or Headphones dimming upon talkback being activated.

When an Intercom destination is in use, a **Procedure** is called to light up the appropriate buttons for that station at all other stations. This provides the operators with an indication that station is in use. A lockout feature then prevents other stations from breaking in when one station is receiving talkback. At the called station, the calling station will flash to indicate who is speaking. The lamp will continue flashing for a few seconds longer, in case the operator was not watching the console at the time they were called. This also provides a good visual cue of which button to press to respond.

To build this Intercom system, first allocate appropriate buttons for each studio and the stations it can call. It is important to allocate these first.

Components

The Intercom system makes use of a number of **Trigger** and **Procedure** components. **Procedures** are used to cut down on code repetition, and centralise updates.

In each studio, an ON and OFF **Trigger** set is required for each button that calls another location. This **Trigger** will call upon the required procedures to turn ON or OFF the destination lamps, and assist with routing.

You will notice that two timer numbers are referenced in each **Trigger** set. These timers allow the flashing source lamp to stay on for a number of seconds after a station called.

The logic is also structured so that a button release can only turn off an Intercom route when that particular station set it in the first place. This prevents another station accidentally killing a conversation that is already in place.

Button On Trigger (press)

- Check lockout variable to ensure it is equal to zero (destination not currently in use).
- Cancel the timer that the destination Off Trigger sets (restart when the button is released).
- Cancel the timer that the destination sets for this source (prevents the lamp being turned off when this studio replies).
- Set the Off Trigger Active (ensures the off only happens after the on successfully executes).
- Route the source Mic to the Talkback Return at the destination.
- Call Procedure to turn Talk Return audio on.
- Call Procedure to turn on all lamps for that destination so all surfaces see it is in use.
- Set the calling station lamp to flash at destination.

Button Off Trigger (release)

- Call Procedure to turn Talk Return audio off.
- Call Procedure to turn off all lamps for that destination so all surfaces see it is free.
- Set appropriate timer with 4 seconds delay.
- This timer will turn off the flashing source lamp when the time expires.
- Disable this Off Trigger, so it won't run until the next On Trigger enables it.

Lamps On Procedures (one per destination station)

- Turn ON that destination lamp at each station (except itself, where there is no button).
- For consistency and code copying, we have kept the self-to-self line, but commented it out.

Lamps Off Procedures (one per destination station)

- Turn OFF that destination lamp at each station (except itself, where there is no button).
- For consistency and code copying, we have kept the source-to-source line, but commented it out

Receive On Procedure (one per destination station)

- Turn ON the Talkback Return bus at the destination (bus numbers may vary on older DSP).
- Set the lockout variable to 1.

Receive Off Procedure (one per destination station)

- Turn OFF the Talkback Return bus at the destination (bus numbers may vary on older DSP).
- Route silence source to Talkback Return bus, just to be sure no audio continues (optional).
- Restore the lockout variable to 0.

Options

These scripts can be customized to suit your requirements. For example, **Talkback Rtn** can be routed to Cue Speaker, Headphones (including selectable mute when Mics on), Monitor speakers, and Guest/Studio send. These functions are set by various bus switches on the monitoring inputs, and are usually pre-set in your *AEConfig* DSP tables.

If you wish to send talkback to an external location, you can omit the **Talkback Rtn** bus on/off lines in the procedure. If you wish to receive talkback from an external location, write a **Trigger** that is activated by a GPI ON and OFF instead of a button (it will have no lamp tallies obviously).

Timers

Each Intercom source/destination set needs a unique timer number allocated to it. It is important these timers are not shared, or unpredictable results may occur with source lamp flashing.

To allocate the timer numbers, we suggest you make a matrix of all studios/stations down the page, with 12 columns for timer numbers across the page. 12 is the normal number to use, as most **Logitek** button panels have groups of 12 buttons. It doesn't matter if you don't use all the buttons.

Dest / *Source*	**Studio A**	**Studio B**	**Etc**	**Etc**	**Etc**	**Etc**						
Studio A	1	2	3	4	5	6	7	8	9	10	11	12
Studio B	13	14	15	16	17	18	19	20	21	22	23	24
Etc	25	26	27	28	29	30	31	32	33	34	35	36

When allocating the timers, first start at the source and locate the destination you are calling. E.g., for Studio A calling Studio B, timer 2 will be used to set the source lamp flash.

Then find the reverse direction, in this case timer 13. This timer must be cancelled in the On Trigger, otherwise if Studio A promptly replies to Studio B, the lamp would be turned off by the previously set timer from Studio B.

This may sound a little cryptic. If so, try some Intercoms without the second timer being cancelled. You will notice if you have a two-way conversation with a particular station, you will often end up with the lamp going off unexpectedly on the button you are pressing. This timer logic prevents that situation.

Further Information

- If Variable – Page 127
- Cancel Timer – Page 75
- Set Trigger Active/Notactive – Page 74
- Route – Page 59
- Call Procedure – Page 53
- Lamp Flash – Page 57
- If Timer – Page 48
- Bus On/Off – Page 55
- Set Variable – Page 124

Script Examples

```
trigger ae1 d[Ctrl - Surf1 GPI In] bus88 on

~=======================================^=======================================^==================================~
~ PROCEDURE: Intercom On                                                         REVISED: 17th February 2005
~ DATA/ID: AE1 Port1 (StA) Wide Softkey TALK1
~ DESCRIPTION: Routes intercom audio, turns return bus on, lights tally lamps.
~ NOTES/DEPENDENCIES: Off trigger is set active only if this trigger runs (to ensure an OFF can't happen without an ON)
~ DEBUG STATUS:
~=======================================^================~====================^==================================~

     if vIcomInUse_Ed1 = 0
      cmd cancel timer1                                 ~ cancel timer to allow restart in off trigger
      cmd cancel timer13                                ~ cancel timer from dest to prevent source lamp cancel
      cmd active trigger ae1 d[Ctrl - Surf1 GPI In] bus88 off                   ~ set off trigger active
      cmd ae2 route s[StA Mic1] to d[Port1 TalkbackRtn In]                      ~ route audio to talk rtn input
      call IcomRxOn_Ed1                                                         ~ talk rtn audio on
      call IcomDestLampsOn_Ed1                                                  ~ tally destination lamps
      cmd ae2 surf1 bridge lamp01 flash continuous                              ~ flash source lamp
     endif

trigger ae1 d[Ctrl - Surf1 GPI In] bus88 off

~=======================================^=======================================^==================================~
~ PROCEDURE: Intercom Off                                                        REVISED: 17th February 2005
~ DATA/ID: AE1 Port1 (StA) Wide Softkey TALK1
~ DESCRIPTION: Turns return bus off, turns off tally lamps.
~ NOTES/DEPENDENCIES: This trigger is set active only if ON trigger runs (to ensure an OFF can't happen without an ON)
~ DEBUG STATUS:
~=======================================^================~====================^==================================~

     call IcomRxOff_Ed1                                 ~ talk rtn audio off
     call IcomDestLampsOff_Ed1                          ~ tally off destination lamps
     if timer1 wait4
      cmd ae2 surf1 bridge lamp01 off                   ~ turn off source lamp after delay
     endif
     cmd notactive trigger ae1 d[Ctrl - Surf1 GPI In] bus88 off                 ~ set off trigger not active again



procedure IcomDestLampsOn_StA

~=======================================^=======================================^==================================~
~ PROCEDURE: Intercom Destination Lamps on/off                                   REVISED: 17th February 2005
~ DATA/ID: ABC Renmark Studio A (StA)
~ DESCRIPTION: Turns intercom destination lamps on or off at all stations.
~ NOTES/DEPENDENCIES: Commented lines not used, but included for possible future use.
~ DEBUG STATUS:
~=======================================^================~====================^==================================~

~    cmd ae1 d[Ctrl - Surf1 GPI Out] bus   on            ~ Dest Lamp ON at stn - Studio A (Not used)
     cmd ae2 surf1 bridge lamp01 on                      ~ Dest Lamp ON at stn - Edit 1 (Bridge Lamp 1)
     cmd ae2 surf2 bridge lamp01 on                      ~ Dest Lamp ON at stn - Edit 2 (Bridge Lamp 1)
     cmd ae2 d[ER Intercom Send] bus31 on                ~ Dest Lamp ON at stn - Equip Room (COM-12 Lamp 1)


procedure IcomDestLampsOff_StA

~=======================================^=======================================^==================================~
~ PROCEDURE: Intercom Destination Lamps on/off                                   REVISED: 17th February 2005
~ DATA/ID: ABC Renmark Studio A (StA)
~ DESCRIPTION: Turns intercom destination lamps on or off at all stations.
~ NOTES/DEPENDENCIES: Commented lines not used, but included for possible future use.
~ DEBUG STATUS:
~=======================================^================~====================^==================================~

~    cmd ae1 d[Ctrl - Surf1 GPI Out] bus   off           ~ Dest Lamp OFF at stn - Studio A (Not used)
     cmd ae2 surf1 bridge lamp01 off                     ~ Dest Lamp OFF at stn - Edit 1 (Softkey Lamp 1)
     cmd ae2 surf2 bridge lamp01 off                     ~ Dest Lamp OFF at stn - Edit 2 (Bridge Lamp 1)
     cmd ae2 d[ER Intercom Send] bus31 off               ~ Dest Lamp OFF at stn - Equip Room (COM-12 Lamp 1)
```

```
procedure IcomRxOn_StA

~=====================================^=====================================^=================================~
~ PROCEDURE: Intercom Receive Bus on/off                                     REVISED: 17th February 2005
~ DATA/ID: ABC Renmark Studio A (StA)
~ DESCRIPTION: Turns intercom receive bus on at station (either TalkRtn1 bus or crosspoint level)
~ NOTES/DEPENDENCIES:
~ DEBUG STATUS:
~=====================================^================~====================^=================================~

      cmd ae1 d[Port1 Monitor In] bus03 on               ~ monitor dim and TB receive enable ON
      cmd set vIcomInUse_StA = 1                        ~ lockout status ON


procedure IcomRxOff_StA

~=====================================^=====================================^=================================~
~ PROCEDURE: Intercom Receive Bus on/off                                     REVISED: 17th February 2005
~ DATA/ID: ABC Renmark Studio A (StA)
~ DESCRIPTION: Turns intercom receive bus off at station (either TalkRtn1 bus or crosspoint level)
~ NOTES/DEPENDENCIES:
~ DEBUG STATUS:
~=====================================^================~====================^=================================~

      cmd ae1 d[Port1 Monitor In] bus03 off              ~ monitor dim and TB receive enable OFF
      cmd ae1 route s[AE1 Silence] to d[Port1 TalkbackRtn In]  ~ route silence to be sure no audio continues
      cmd set vIcomInUse_StA = 0                        ~ lockout status OFF
```

22 On-air Switching Examples

The **Logitek** system can be used to build flexible on-air delegation switchers. These can be customized to suit station requirements and provide for almost any switching scenario.

As the design of a delegation switcher can be complex in many cases, we suggest engaging **Logitek Electronic Systems** or your local reseller to consult on the design and programming.

Examples of different on-air switchers from real-life customer sites are provided below. The script examples for these are not included, due to the size and number of **Triggers** required. Please contact **Logitek Electronic Systems** or your reseller for more information on these.

Assignment Mixer

The **Assignment Mixer** allows studios to mix into the final output. A studio may assign itself to a particular on-air path, with multiple studios assigned at the same time if desired. This facilitates seamless studio changeovers and allows news booths to directly assign to-air if required.

An available **Mixer Surface** port is required on an **Audio Engine**. This is a virtual surface, so no physical console is required.

The size of this virtual surface depends on the assignment mixer size. One fader input is required per studio (additional studio Mix Minus or Network Cleanfeeds will also require a fader each). One Mix Bus (e.g. PGM, AUX1-8) is required per destination path. If using AUX4-8 as destination paths, the virtual surface must be on Port 1. In larger facilities, we suggest using a dedicated MCR Audio Engine for the **Assignment Mixer**. In some cases, custom **DSP** tables can be provided to re-allocate other **Audio Engine** resources to gain additional destination paths.

If using *SharcAttack* **DSP** cards, each input to the delegation mixer can have EQ and dynamics settings applied, providing basic protection limiting and compression prior to link paths.

The **DSP** table for this **Audio Engine** Port will be customized to set fader inputs to unity gain by default, with the relevant BUS settings. The mixer is then controlled by **Triggers**.

Panel Example

Below is an example of a 12 button panel. This could be a *Numix* **Bridge**, *Remora* **Bridge** (vertical), or a *Mosaic* MLX-42/MLX-NSOFT modules mounted in a compatible **Wide Meter Bridge** assembly.

Network 1	Network 2	Network 3	Network 4	Network 5	Network 6	Network 7	Network 8	Local		ENABLE	OnAir Master

Functionality

- Press the Enable button to turn the panel lock off. It will flash.
- Use the On-air Master to turn your studio on or off air in on action.
- When on-air, press any of the paths to add your studio into that mix. Press again to remove.
- When off-air, press any of the paths to pre-select (flashing). Press again to remove.
- Routes can also be tallied to text screens.

Requirements

- Virtual surface port available on **Audio Engine**.
- 1 Fader per input to **Assignment Mixer**.
- 1 Mix Bus per output to destination paths.
- 1 button per path in each studio, plus a master on/off button.

Network Source Switcher – Direct Access Version

The **Network Source Switcher** provides simple cut-based switching between sources.

On a 24-button panel, this switcher works well for up to 5 outputs with 3-5 sources each. One button is required per source/destination combination (e.g. Studio A Post Delay to transmitter 1). If more source/destinations combinations are required, the XY version of this switcher can be used (see following section).

This panel allows an operator to preselect a number of sources, which will indicate with a flashing lamp in all studios. The "TAKE" button will execute all these pre-selected routes in one action.

The **Network Source Switcher** provides easy to use operation, with visibility of the current routes on the button panel in each studio. A "TAKE" operation is a cut from one source to another – there is no capacity to mix two sources simultaneously.

One **DSP Crosspoint** or **Output Route** is required per output path. Any number of inputs can be routed, limited only by button capacity on the panel. **DSP Crosspoints** are recommended if gain control or mono mixing is required. If using **DSP** Crosspoints on a *SharcAttack* **DSP** card, processing and input meters are also available.

Commonly, the **DSP** Crosspoints could be allocated to a Port 3 virtual surface. The stereo Crosspoints on Port 3 will allow a mono mix from a stereo source if required. As Port 3 is often unused, the outputs can be duplicated on two **Audio Engines** for redundancy. This also removes the need for a separate **Audio Engine** just for MCR switching and is therefore quite suitable for smaller facilities.

Panel Example

Below is an example of a 24 button panel. This could be a *Numix* NW24 wedge, *Remora* BTN24 module, or two *Mosaic* MLX-42/MLX-NSOFT modules mounted in a compatible **Wide Meter Bridge** assembly.

For each destination, a number of sources are available. These lights will be shown in each studio, providing an overview of the delegation at all times. Pre-selects (flashing) also show in each studio.

TX - A			TX - B				EXCH / OB RETURN				
Studio A Post Delay	Prod Bth Post Delay	Network	Studio A Post Delay	Prod Bth Post Delay	Network		Studio A Post Delay	Studio A Mix Minus	Prod Bth Mix Minus	TX A	TAKE
Studio A Post Delay	Studio A Mix Minus	Prod Bth Post Delay	Prod Bth Mix Minus	NewsBth Mix Minus	TX A						
ISDN											

Studio A Post Delay is selected to TX-A, TX-B, EXCH & ISDN. TX-A is pending to ISDN.

Functionality

- Press the desired source-to-destination button.
- Lit destinations cannot be de-selected.
- Unlit destinations can be pre-selected (flashing) if allowed.
- Once pre-selected to destination, you have a lock on that pre-select. The button will flash.
- Press TAKE to accept, or press the flashing pre-select again to cancel.
- Routes can also be tallied to text screens.

Requirements

- 1 Output or DSP Crosspoint & Output per path (DSP provides gain control and mono mix).
- If using DSP Crosspoints a virtual Router surface is required on an available port.
- 1 Button panel per studio.
- 1 Button on panel per source/destination combo.

Network Source Switcher – XY Version

The **Network Source Switcher – XY Version** provides the same functionality as the **Direct Access Version**, but allows for more source/destination combinations.

The disadvantage is that current routes are less visible on the panel itself and surface or virtual text screens must be used to tally routing information.

Like the **Direct Access NSS**, the **XY version** uses **DSP Crosspoints** on a virtual surface or **Output Routes**.

Panel Example

Below is an example of a 24 button panel. This could be a *Numix* NW24 wedge, *Remora* BTN24 module, or two *Mosaic* MLX-42 modules mounted in a compatible **Wide Meter Bridge** assembly.

The top row of 12 buttons is for sources and the TAKE button.
The bottom row of 12 buttons is for destinations, creating up to an 11x12 router.

Studio A Post Delay	Studio A Mix Minus	Studio B Post Delay	Studio B Mix Minus	Edit Bth Program	Edit Bth Mix Minus	Network	Network Standby	ISDN	TX A	Silence	TAKE
TX A	TX B	POTS	ISDN	EXCH	SPARE						

In this example, Studio A Post Delay, TX A, TX B and TAKE buttons are lit. ISDN path is flashing.

Functionality

- Press source button (top row). If allowed in this studio, it will display the current destination routes on the bottom row.
- Lit destinations cannot be de-selected.
- Unlit destinations can be pre-selected (flashing) if allowed.
- Once pre-selected to a destination, you have a lock on that pre-select. The button will flash.
- Press TAKE to accept, or press the flashing pre-select again to cancel.
- Press the source to clear all.
- To view, you can press the source you are interested and see which destinations it feeds.
- Routes can also be tallied to text screens.

Requirements

- 1 Output or DSP Crosspoint & Output per path (DSP provides gain control and mono mix).
- If using DSP Crosspoints a virtual Router surface is required on an available port.
- 1 Button panel per studio.
- 1 Button on panel per source, plus 1 "TAKE" button (12 suggested, using top row of BTN24).
- 1 Button on panel per destination (12 suggested, using bottom row of BTN24).

23 Device Control Examples

The **Logitek** system can interface to a variety of external devices using serial or IP based control. This chapter shows examples of commonly used device control.

BetaBrite Signs

BetaBrite manufacture scrolling LED signs that can be used to display messages from *JetStream Server*.
For more information on BetaBrite signs, see the company's website at http://www.betabrite.com.

For this example, you will require:

- A BetaBrite message board
- *Supervisor*
- An unused RS-232 port on your *Supervisor* PC

When you setup the sign, program a static message into the display using either the remote control or the supplied PC software. It's best to not overwhelm the operators with text and flashy displays so it will be more obvious to the operator when an alarm condition exists.

We suggest setting one message with the time of day only, or a single period mark set to "Hold". This will give you a static display that lets you know the sign is turned on. Once that is stored in the BetaBrite, connect it to the available RS-232 port on your *Supervisor* computer.

> ⚠ Current BetaBrite signs that are commonly available use USB instead of serial. BetaBrite signs using the older serial protocol are still available as the "BetaBrite Classic" although they are difficult to find.

Configuring Supervisor

To configure *Supervisor* to support the external device:

1. Click on the ***Com Port Control*** tab
2. Look in the ***Auxiliary Ports*** box. In the "Lowest Aux Ports" box, put the port number the sign is connected to. In the "Number of Aux Ports" box, put 1.
3. In the grid inside the ***Auxiliary Ports*** box you will now see the new Aux port and it will be highlighted in blue. Right click on this line. You will now be able to set the port's parameters. You must set the port as follows to communicate with the BetaBrite:

Baud Rate	9600
Data Bits	7
Stop Bits	2
Parity	Even

4. Click ***Accept*** once you have set the parameters and then check the ***Aux Ports Active*** box.
5. Close and re-open *Supervisor* to open the required Com Ports.

➥ *For more information on Aux Ports see the Supervisor User's Manual.*

Functionality Description

In this example, we will use the ***Priority Message*** command in the BetaBrite to display alert messages to operators. When the BetaBrite receives a ***Priority Message***, it interrupts whatever the sign was displaying and switches to the text sent with the priority command, until it receives a command to stop and return to the previous message routine.

Components

In this example, we have used an **Audio Engine** GPI as the ON & OFF **Triggers** to display the message on the BetaBrite sign.

To activate the message, we need to send a serial string to the Aux port. Please note the required space between the <ESC> and the lowercase letter that follows it.

In this example, we will send "On Air" to com 2:

GPI On Trigger

- Send text “On Air” to BetaBrite.

GPI Off Trigger

- Send clear command to BetaBrite (if you do not issue this command, the alarm display will never turn off!).

Further Information

- Text to Com Port – Page 70

If you would like further assistance with BetaBrite signs, please visit the Logitek Tech Forum at http://www.logitekaudio.com. Further examples are available here, and you can communicate with other Logitek users who have implemented BetaBrites.

Some additional BetaBrite protocol information follows.

Font Size & Effects

You can adjust the mode (effect) and character set (font size) by adjusting two characters in the string.

In the command below, the “b” character makes the message hold on screen. The “2” preceding the text sets the font size. These characters can be substituted to provide other effects.

```
cmd port2 "<nul><nul><nul><nul><nul><soh>Z00<stx>A0<esc> b<sub>9On Air<eot>"
```

Name	Description	Command
Rotate	Message travels right to left	a
Hold	Message remains stationary	b
Flash	Message remains stationary and flashes	c
Roll Up	Previous Message is pushed up by a new message	e
Roll Down	Previous Message is pushed down by a new message	f
Roll Left	Previous Message is pushed left by a new message	g
Roll Right	Previous Message is pushed right by a new message	h
Wipe Up	New message wipes over old from bottom to top	i
Wipe Down	New message wipes over old from top to bottom	j
Wipe Left	New message wipes over old from left to right	k
Wipe Right	New message wipes over old from right to left	l
Automode	Sign picks modes at random	o

Name	Command
Five Slim	1
Five Stroke	2
Seven Slim	3
Seven Stroke	4
Seven Slim Fancy	5
Seven Stroke Fancy	6
Seven Shadow	7
Wide Stroke Seven Fancy	8
Wide Stroke Seven	9
Seven Shadow Fancy	:
Five Wide	;
Seven Wide	<
Seven Fancy Wide	=
Wide Stroke Five	>

➥ *Contact BetaBrite for further information on the sign protocol.*

Script Examples

```
trigger ae1 device01 bus01 on                                                    ~ AE1 GPI Input #1

~=======================================^=======================================^=================================~
~ PROCEDURE: BetaBrite Message                                                   REVISED: 14th June 2005
~ DATA/ID: AE1 GPI 1 ON
~ DESCRIPTION: Sends / clears message text to Beta Brite sign on COM2
~ NOTES/DEPENDENCIES: COM port must be active in Supervisor as AUX port
~ DEBUG STATUS:
~=======================================^================~==================^=================================~

     cmd port2 "<nul><nul><nul><nul><nul><soh>Z00<stx>A0<esc> b<sub>9On Air<eot>"



trigger ae1 device01 bus01 off                                                   ~ AE1 GPI Input #1

~=======================================^=======================================^=================================~
~ PROCEDURE: BetaBrite Message                                                   REVISED: 14th June 2005
~ DATA/ID: AE1 GPI 1 OFF
~ DESCRIPTION: Sends / clears message text to Beta Brite sign on COM2
~ NOTES/DEPENDENCIES: COM port must be active in Supervisor as AUX port
~ DEBUG STATUS:
~=======================================^================~==================^=================================~

     cmd port2 "<nul><nul><nul><nul><nul><soh>Z00<stx>A0<eot>"
```

24 Guest Panel Examples

This chapter includes some examples of using **Trigger** functions with **Logitek's** guest headphone panels. These **Triggers** allow the guest to control their mic on/off/mute functions, receive tally information, and control the guest panel timer display. These examples are for a *GST-22* panel, however also apply to a *GST-20*.

➥ *See the Logitek Utility Panel User's Manual for how to set up guest panels.*

Mic On/Off

Functionality Description

Allows the guest to turn their microphone on and off from the guest panel. The microphone input can be on any fader of a surface – the command is sent to the first matching instance of the mic input on any **Surface** of the specified **Audio Engine**.

If you do not wish the guest to have mic on/off control, either omit these **Triggers**, or disable them in *CommandBuilder*. You could also customize the **Triggers** using **Variables** and **If Statements** to only permit the functionality during certain scenes.

Components

This function has an ON (button pressed) **Trigger** for the mic ON and OFF buttons.

On Trigger – Mic ON button

- Turn on microphone.
- Turn GST-22 ON lamp on.
- Turn GST-22 OFF lamp off.
- Turn GST-22 Relay Tally on.
- Turn GST-22 timer on.
- Run GST-22 timer.
- Reset GST-22 timer to zero.

On Trigger – Mic OFF button

- Turn off microphone.
- Turn GST-22 ON lamp off.
- Turn GST-22 OFF lamp on.
- Turn GST-22 Relay Tally off.
- Reset GST-22 timer to zero.
- Stop GST-22 timer.

Off Triggers

Off Triggers for these buttons are not required (an **Off Trigger** occurs when a button is released). The "stubs" for these can be included if desired, and are shown at the bottom of the examples.

Further Information

- Bus On/Off – Page 55

Script Examples

```
trigger ae1 d[St731 Guest 1 H/P] bus31 on

~=====================================^=====================================^==================================~
~ PROCEDURE: Guest Mic ON/OFF                                               REVISED: 12th July 2005
~ DATA/ID: AE1 Port5 (731) GST-22 #1 Mic ON
~ DESCRIPTION: Activate Mic if On button pressed. Cancel Mic if Off button pressed.
~ NOTES/DEPENDENCIES: Activates/Deactivates Main bus for Mic input.
~ DEBUG STATUS:
~=====================================^================~====================^==================================~

     cmd ae1 s[St731 Mic 2] bus0 on                      ~ Mic on
     cmd ae1 d[St731 Guest 1 H/P] bus11 on               ~ On lamp on
     cmd ae1 d[St731 Guest 1 H/P] bus12 off              ~ Off lamp off
     cmd ae1 d[St731 Guest 1 H/P] bus20 on               ~ Tally Output on

     cmd ae1 d[St731 Guest 1 H/P] bus21 on               ~ timer on
     cmd ae1 d[St731 Guest 1 H/P] bus22 on               ~ timer run
     cmd ae1 d[ST731 Guest 1 H/P] bus23 on               ~ timer reset


trigger ae1 d[St731 Guest 1 H/P] bus32 on

~=====================================^=====================================^==================================~
~ PROCEDURE: Guest Mic ON/OFF                                               REVISED: 12th July 2005
~ DATA/ID: AE1 Port5 (731) GST-22 #1 Mic OFF
~ DESCRIPTION: Activate Mic if On button pressed. Cancel Mic if Off button pressed.
~ NOTES/DEPENDENCIES: Activates/Deactivates Main bus for Mic input.
~ DEBUG STATUS:
~=====================================^================~====================^==================================~

     cmd ae1 s[St731 Mic 2] bus0 off                     ~ Mic off
     cmd ae1 d[St731 Guest 1 H/P] bus11 off              ~ On lamp off
     cmd ae1 d[St731 Guest 1 H/P] bus12 on               ~ Off lamp on
     cmd ae1 d[St731 Guest 1 H/P] bus20 off              ~ Tally Output off

     cmd ae1 d[St731 Guest 1 H/P] bus23 on               ~ timer reset
     cmd ae1 d[St731 Guest 1 H/P] bus22 off              ~ timer stop


trigger ae1 d[St731 Guest 1 H/P] bus31 off
     ~ not required


trigger ae1 d[St731 Guest 1 H/P] bus32 off
     ~ not required
```

Mic Mute

Functionality Description

Allows the guest to mute their microphone from the guest panel. The microphone input can be on any fader of a surface – the command is sent to the first matching instance of the mic input on any **Surface** of the specified **Audio Engine**. The mic input should be on ONE fader only.

If you do not wish the guest to have mic mute control, either omit these **Triggers**, or disable them in *CommandBuilder*. You could also customize the **Triggers** using **Variables** and **If Statements** to only permit the functionality during certain scenes.

Components

This function has an ON (button pressed) and an OFF (button released) **Trigger** for the mic mute.

Button On Trigger (pressed)

- Scan surface to see if microphone channel is on.
- Turn on Cough Mute function (BUS8) on microphone channel.
- Tally text "MUTE" to microphone channel on surface (so operator sees mute is active).
- Turn GST-22 MUTE lamp on.

Button Off Trigger (released)

- Turn off Cough Mute function (BUS8) on microphone channel.
- Clear tally text to microphone channel on surface (so operator sees mute is not active).
- Turn GST-22 MUTE lamp off.

Further Information

- If State Scan – Page 130
- Bus On/Off – Page 55
- Text Label – Page **Error! Bookmark not defined.**

Script Examples

```
trigger ae1 d[St731 Guest 1 H/P] bus33 on

~=======================================^=======================================^=================================~
~ PROCEDURE: Cough Mute                                                          REVISED: 12th July 2005
~ DATA/ID: AE1 Port5 (731) GST-22 #1 Mic Mute ON
~ DESCRIPTION: Activate Mic Mute if Mute button pressed. Cancel Mic Mute if Mute button released.
~ NOTES/DEPENDENCIES: Activates/Deactivates Mute bus for Mic input.
~ DEBUG STATUS:
~=======================================^================~====================^=================================~

     if state scan ( ae1 surf1 s[St731 Mic 2] bus0 on ) then
      cmd ae1 s[St731 Mic 2] bus8 on                      ~ cough mute on
      cmd ae1 s[St731 Mic 2] text label "  MUTE  "        ~ tally label to fader
      cmd ae1 d[St731 Guest 1 H/P] bus13 on               ~ guest mute lamp on
     endif



trigger ae1 d[St731 Guest 1 H/P] bus33 off

~=======================================^=======================================^=================================~
~ PROCEDURE: Cough Mute                                                          REVISED: 12th July 2005
~ DATA/ID: AE1 Port5 (731) GST-22 #1 Mic Mute OFF
~ DESCRIPTION: Activate Mic Mute if Mute button pressed. Cancel Mic Mute if Mute button released.
~ NOTES/DEPENDENCIES: Activates/Deactivates Mute bus for Mic input.
~ DEBUG STATUS:
~=======================================^================~====================^=================================~

      cmd ae1 s[St731 Mic 2] bus8 off                     ~ cough mute off
      cmd ae1 s[St731 Mic 2] text label "        "        ~ tally label to fader
      cmd ae1 d[St731 Guest 1 H/P] bus13 off              ~ guest mute lamp off
```

Lamp Tallies

Functionality Description

Tallies the microphone on/off lamps on the *GST-22* panel when the console operator turns the mic on or off. The microphone input can be on any fader of a surface – the command is registered whenever a matching instance of the mic input is changed on any **Surface** of the specified **Audio Engine**.

Even if you do not give the guest control of their mic on/off, you should include these **Triggers** to provide the guest with tally status when their mic is turned on or off by the console operator.

Components

This function has an ON (mic turned on) and an OFF (mic turned off) **Trigger** for the mic channel.

On Trigger

- Scan surface to ensure microphone channel is ON at this surface.
- Turn GST-22 ON lamp on.
- Turn GST-22 OFF lamp off.
- Turn GST-22 Relay Tally on.
- Turn GST-22 timer on.
- Run GST-22 timer.
- Reset GST-22 timer to zero.

Off Trigger

- Scan surface to ensure microphone channel is OFF at this surface.
- Turn GST-22 ON lamp off.
- Turn GST-22 OFF lamp on.
- Turn GST-22 Relay Tally off.
- Reset GST-22 timer to zero.
- Stop GST-22 timer.

Further Information

- If State Scan – Page 130
- Bus On/Off – Page 55

Script Examples

```
trigger ae1 s[St731 Mic 2] bus0 on

~=======================================^=======================================^=================================~
~ PROCEDURE: Mic ON/OFF                                                          REVISED: 12th July 2005
~ DATA/ID: AE1 Port5 (731) GST-22 #1 Mic Mute OFF
~ DESCRIPTION: Activate Mic Mute if Mute button pressed. Cancel Mic Mute if Mute button released.
~ NOTES/DEPENDENCIES: Activates/Deactivates Mute bus for Mic input.
~ DEBUG STATUS:
~=======================================^================~====================^=================================~

     if state scan ( ae1 surf1 s[St731 Mic 2] bus0 on ) then
      cmd ae1 d[St731 Guest 1 H/P] bus11 on               ~ On lamp on
      cmd ae1 d[St731 Guest 1 H/P] bus12 off              ~ Off lamp off
      cmd ae1 d[St731 Guest 1 H/P] bus20 on               ~ Tally Output on

      cmd ae1 d[St731 Guest 1 H/P] bus21 on               ~ timer on
      cmd ae1 d[St731 Guest 1 H/P] bus22 on               ~ timer run
      cmd ae1 d[St731 Guest 1 H/P] bus23 on               ~ timer reset
     endif



trigger ae1 s[St731 Mic 2] bus0 off

~=======================================^=======================================^=================================~
~ PROCEDURE: Mic ON/OFF                                                          REVISED: 12th July 2005
~ DATA/ID: AE1 Port5 (731) GST-22 #1 Mic Mute OFF
~ DESCRIPTION: Activate Mic Mute if Mute button pressed. Cancel Mic Mute if Mute button released.
~ NOTES/DEPENDENCIES: Activates/Deactivates Mute bus for Mic input.
~ DEBUG STATUS:
~=======================================^================~====================^=================================~

     if state scan ( ae1 surf1 s[St731 Mic 2] bus0 off ) then
      cmd ae1 d[St731 Guest 1 H/P] bus11 off              ~ On lamp off
      cmd ae1 d[St731 Guest 1 H/P] bus12 on               ~ Off lamp on
      cmd ae1 d[St731 Guest 1 H/P] bus20 off              ~ Tally Output off

      cmd ae1 d[St731 Guest 1 H/P] bus23 on               ~ timer reset
      cmd ae1 d[St731 Guest 1 H/P] bus22 off              ~ timer stop
     endif
```

25 vSnapshot Examples

This chapter includes some examples of using **Trigger** functions with **Logitek's** *vSnapshot* application. These **Triggers** allow the user to perform Capture, Recall and Edit functions from the Mosaic / Artisan surface. These examples are for a *MLX-WSOFT* panel, but also apply to any panel with user programmable buttons.

➥ *See the Logitek vTools User's Manual for how to set up vSnapshot.*

Capture On

Functionality Description

Launches *vSnapshot* Capture Mode and captures settings in the console's RAM. The command is sent to the specified **Audio Engine**.

Components

This function has an ON (button pressed) **Trigger** for the specified button.

On Trigger – Capture ON button

- Turn on reserved bus to launch *vSnapshot* Capture Mode.
- Turn button's lamp on.

Off Triggers

Off Triggers for these buttons are activated when the Done button is clicked on the *vSnapshot* Capture Page.

Further Information

- Bus On/Off – Page 55

Script Examples

```
trigger ae1 d[Ctrl - Surf1 GPI In] bus42 on

~=======================================^=======================================^=================================~
~ PROCEDURE: vSnapshot Capture ON/OFF                                    REVISED: 1st February 2007
~ DATA/ID: AE1 Port1 (StA) Wide Softkey SCENE 11
~ DESCRIPTION: Activate vSnapshot Capture Mode if On button pressed.
~ NOTES/DEPENDENCIES: Tally Lamp is turned off when Done is clicked in vSnapshot.
~ DEBUG STATUS:
~=======================================^================~==================^=================================~

     cmd ae1 device02 bus201 on                          ~ turn on vSnapshot Capture Mode
     cmd ae1 d[Ctrl - Surf1 GPI Out] bus42 on            ~ turn lamp on



trigger ae1 device02 bus201 off

~=======================================^=======================================^=================================~
~ PROCEDURE: vSnapshot Capture ON/OFF                                    REVISED: 1st February 2007
~ DATA/ID: AE1 Port1
~ DESCRIPTION: Activate vSnapshot Capture Mode if On button pressed.
~ NOTES/DEPENDENCIES: Tally Lamp is turned off when Done is clicked in vSnapshot.
~ DEBUG STATUS:
~=======================================^================~==================^=================================~

     cmd ae1 d[Ctrl - Surf1 GPI Out] bus42 off           ~ turn lamp off
```

Recall On

Functionality Description

Launches *vSnapshot* Recall Mode and recalls settings to the console's RAM. The command is sent to the specified **Audio Engine**.

Components

This function has an ON (button pressed) **Trigger** for the specified button.

On Trigger – Recall ON button

- Turn on reserved bus to launch *vSnapshot* Recall Mode.
- Turn button's lamp on.

Off Triggers

Off Triggers for these buttons are activated when the Done button is clicked on the *vSnapshot* Recall Page.

Further Information

- Bus On/Off – Page 55

Script Examples

```
trigger ae1 d[Ctrl - Surf1 GPI In] bus43 on

~=====================================^=====================================^==================================~
~ PROCEDURE: vSnapshot Recall ON/OFF                                        REVISED: 1st February 2007
~ DATA/ID: AE1 Port1 (StA) Wide Softkey SCENE 12
~ DESCRIPTION: Activate vSnapshot Recall Mode if On button pressed.
~ NOTES/DEPENDENCIES: Tally Lamp is turned off when Done is clicked in vSnapshot.
~ DEBUG STATUS:
~=====================================^================~====================^==================================~

     cmd ae1 device02 bus202 on                          ~ turn on vSnapshot Recall Mode
     cmd ae1 d[Ctrl - Surf1 GPI Out] bus43 on            ~ turn lamp on


trigger ae1 device02 bus202 off

~=====================================^=====================================^==================================~
~ PROCEDURE: vSnapshot Recall ON/OFF                                        REVISED: 1st February 2007
~ DATA/ID: AE1 Port1
~ DESCRIPTION: Activate vSnapshot Recall Mode if On button pressed.
~ NOTES/DEPENDENCIES: Tally Lamp is turned off when Done is clicked in vSnapshot.
~ DEBUG STATUS:
~=====================================^================~====================^==================================~

     cmd ae1 d[Ctrl - Surf1 GPI Out] bus43 off           ~ turn lamp off
```

Edit On

Functionality Description

Launches *vSnapshot* Edit Mode. The command is sent to the specified **Audio Engine**.

Components

This function has an ON (button pressed) **Trigger** for the specified button.

On Trigger – Edit ON button

- Turn on reserved bus to launch *vSnapshot* Edit Mode.
- Turn button's lamp on.

Off Triggers

Off Triggers for these buttons are activated when the Done button is clicked on the *vSnapshot* Edit Page.

Further Information

- Bus On/Off – Page 55

Script Examples

```
trigger ae1 d[Ctrl - Surf1 GPI In] bus40 on

~=======================================^=======================================^=================================~
~ PROCEDURE: vSnapshot Edit ON/OFF                                    REVISED: 1st February 2007
~ DATA/ID: AE1 Port1 (StA) Wide Softkey SCENE 9
~ DESCRIPTION: Activate vSnapshot Edit Mode if On button pressed.
~ NOTES/DEPENDENCIES: Tally Lamp is turned off when Done is clicked in vSnapshot.
~ DEBUG STATUS:
~=======================================^================~====================^=================================~

     cmd ae1 device02 bus203 on                          ~ turn on vSnapshot Recall Mode
     cmd ae1 d[Ctrl - Surf1 GPI Out] bus40 on            ~ turn lamp on


trigger ae1 device02 bus203 off

~=======================================^=======================================^=================================~
~ PROCEDURE: vSnapshot Edit ON/OFF                                    REVISED: 1st February 2007
~ DATA/ID: AE1 Port1
~ DESCRIPTION: Activate vSnapshot Edit Mode if On button pressed.
~ NOTES/DEPENDENCIES: Tally Lamp is turned off when Done is clicked in vSnapshot.
~ DEBUG STATUS:
~=======================================^================~====================^=================================~

     cmd ae1 d[Ctrl - Surf1 GPI Out] bus40 off           ~ turn lamp off
```

What's New in CommandBuilder

- Designed for **Trigger Table** compatibility with *JetStream Server 5*.

Release Notes

The following notes details known issues, version history and current releases of *CommandBuilder*.

Known Issues

The following issues have been reported and confirmed by **Logitek**. As these issues do not affect many users, they have not been assigned development priority. If any of these issues affect your installation, please contact **Logitek Electronic Systems** for possible workarounds or further information.

April 2015

- First release of Command Builder 5
- A drop down list of known IP addresses of the JetStreams derived from the JetFiles is scheduled to be created on the Upload page. Until this is created, users must manually type in IP Addresses of each JetStream Server.

Version History

November 2013

- Last scheduled release of CommandBuilder 3.6. All development moved to the v5 platform.

Dec 2004

- Service Release

Jan 2005

- Corrected issues with **IF VARIABLE** tests using device numbers and variable comparisons.

April 2005

- Problems with **COMPRESSOR GAIN** commands corrected.

May 2005

- 16 character version `CommandBuilder16.exe` release.
- Service release of `CommandBuilder2002.exe` (compiled in Delphi 7).
- Prevents **VARIABLE SELECT** without corresponding **IF CANCEL** (which can freeze *Supervisor*).

June 2005

The following updates apply only to CommandBuilder2002.exe:

- New flash rates now supported. Keyword `ONCE` for flash is no longer supported.
- Fixes incorrect error message on incomplete **SET SELECTION MODE**.
- Fixes **TEXT TO COM PORT** problem with <1C> hex byte being sent.

For 16 character systems, the June 2005 fixes have been incorporated into *CommandBuilder 3*.

August 2005

CommandBuilder3 merges recent fixes and support for 8 & 16-character Audio Engines in the one executable:

- **TIMERS** were incorrectly translating to support tenths of seconds prior to *Supervisor* support being added. *CB 3* & *Supervisor 3* will ultimately support tenths of seconds. Some beta versions of *CB 3* may contain incorrect translations and should not be used. The release version does not have this issue.
- **IF STATE** and **IF STATE SCAN** test statements were not translating correctly if the device name contained "CHAN". This has been resolved.
- **IF STATE SCAN** test statement now validates the device name, and will warn of an invalid device name.
- **ROUTE SELECT** command now validates the device name, and will warn of an invalid device name.
- **MESSAGE FLAG** command now validates the device name, and will warn of an invalid device name.
- **ROUTE3 TEXT** commands now fully support the use of d[Device] notation.
- **VARIABLES** now allow the text "IF" to appear in the variable name. Previously this would attempt to translate as an IF test statement.

September 2005

The following updates apply only to CommandBuilder3.exe:

- *Mosaic* Set Color command can now be sent to a source device using the s[Device] notation.

November 2005

The following updates apply only to CommandBuilder3.exe:

- *Mosaic* Set Color command now supports the v2.x fader module format in *Mosaic*. This version, and future updates should not be used with *Mosaic* v1.x if using the Set Color command for fader On/Off buttons. The new format allows a flash rate to be specified (see *Mosaic* manual for more information).

January 2009

The following updates apply only to CommandBuilder3.5.exe:

- Fixes a bug with **TIMERS** contained within **IF VARIABLE** and **IF STATE** test statements. Previously, **TIMERS** contained within these statements would start regardless of whether the conditions being tested for were true or not and the ensuing commands would execute. This bug fix remedies the problem so that **TIMERS** contained within these test statements do not start unless the test conditions are true. This has been fixed in both the **Trigger** and **Procedure** sections of *CommandBuilder*.

August 2009

The following updates apply only to CommandBuilder3.6.exe:

- *A new **Remember Password** tickbox has been included as of CommandBuilder v3.6.2.0. This is ticked by default, however, if you are remotely managing multiple sites it is a good idea to untick the **Remember Password** box. This can greatly assist in preventing the accidental uploading of an incorrect **Audio Engine Config** to a site.*

Appendix A Keyword Summary

Introduction

This is the list of keywords which are recognized by *CommandBuilder* and which cause actions to occur or are a part of a command statement. Words not on this list can be added to improve the readability of the **Trigger** scripts but will be ignored. The parameters required by certain keywords should not be separated from them by other keywords. Usually a keyword and its parameter are separated only by spaces.

Keywords

d[ ]	Destination Device name
s[ ]	Source Device name
u[ ]	UDP address
ip[ ]	IP address
=	Equal sign – also used to connect a variable or keyword to a parameter value
<>	Not equal
<	Less than
>	Greater than
<=	Less than or equal to
>=	Greater than or equal to
-	Minus sign. Used to set fader trim or pan
+	Plus sign. Same as using no sign
ACCEPT	The accept button on the console
ACTIVE	Sets a Trigger to active state
ADDMIC	Adds talkback mic to Mix Minus
AE	Audio Engine number
ALL	Refers to all lines in a text commands
ALWAYS	Used to set the Mix Minus Bus always on
ANY	Optional keyword for use with a Route Trigger
AQUA	A color used for text on *Numix II* color display
BIG	Sets user text string displayed on a console to large size
BRIDGE	Refers to the Bridge buttons on console
BUS	Bus number in engine
BUTTON	Refers to Softkey, Bridge, or other buttons on console

CANCEL Used to cancel a Timer or other command
CALL Calls a named Procedure set of commands
CENTER Used in pan setting commands
CHAN Shortened form of CHANNEL
CHANNEL Refers to a Fader on a console (same as FADER)
CHANGE Used in show selection commands
CLOCK Used to display a system time string on the console
CLEAR Used to clear displayed text from a console
CMD Indicates the beginning of a command line
CONNECT Makes a serial port connection

DATE Used to set the date of a Schedule Event Trigger
DAY Used in the repeat interval of a Schedule Event Trigger
DEFINE Used to define a group of commands
DELAY Sets the number of seconds before which a timer will fire
DESTINATION Sets the destination com port for outgoing serial data
DEVICE Audio Engine Device number
DISCONNECT Removes a serial port connection
DUMP Used to dump the time in a Talk Delay

ENDDEF Defines the end of a define section
ENDIF Defines the end of an IF section of commands
ENDWHEN Defines the end of a WHEN section of commands
EXIT Stops processing of any further commands in the current trigger (same as QUIT)

FADER Refers to a console Fader (same as CHANNEL)
FLAG Sets large arrow above fader with optional text message
FLASH Same as blink
FRIDAY Day of the week used in Schedule Event trigger
FUNCTION Refers to the function screen on the console

GREEN A color used for text on *Numix II* color display

HOUR Used in the repeat interval of a Schedule Event Trigger

IF Begins a Conditional Trigger or Test Statement
INIT A Trigger that is executed whenever *Supervisor* is started
INSERT Sets the insert mode for user text strings – the line is not erased before the text is displayed

LABEL Text to the screen above a Fader on a console
LAMP Refers to the Lamp in a Button (same as LIGHT)
LEFT Used in setting Fader pan
LEVEL Used to set console Fader level
LINE Line number for user text strings displayed on a console
LIST A mode of the selection screen (alternate to the message mode)
LIGHT Refers to the Lamp in a Button (same as LAMP)

MAGENTA	A color used for text on *Numix II* console
MAX	Used to set the maximum delay in a Talk Delay function
MESSAGE	A mode of the selection screen (alternate to the list mode)
MINUTES	Used in the repeat interval of a Schedule Event Trigger
MIX MINUS	Sets the Mix Minus in effects commands
MIX-	Same as Mix Minus
MONDAY	Day of the week used in a Schedule Event Trigger
NONE	Equivalent to a 0 setting for Fader Trim
NORMAL	User text string that is not highlighted
NOTACTIVE	Used to set a trigger to the inactive mode
NOT	Negates
NUMIX1	Specifies *Numix* version 1 control surface
NUMIX2	Specifies *Numix* version 2 control surface
OFF	Not on, usually a button or talk delay
ON	Not off, usually a button
ONEWAY	Com port communicates from source to destination only
PAN	Sets a Fader pan value
PORT	Reference to a Com Port (with a number to identify)
POSITION	Position in a line of text
PROCEDURE	Used in naming a procedure or subroutine
PULSE	Sends a pulse command to a relay in the Audio Engine or a Surface
QUESTION	Sends text to the Question display area on a console
QUIT	Stops processing of any further commands in the current Trigger (same as EXIT)
RADIO	Sets the Function Button on the console to radio push button mode
RECALL	Retrieve a value that has been stored in a User Variable
RED	A color used for text on *Numix II* console
RELAY	Refers to a relay in Audio Engine or Surface (with number to identify)
RELEASE	Used to set release time with Set Fader Limiter and Set Fader Compressor commands
REMORA	Specifies *Remora* control surface
REPEAT	Used to make a Schedule Event Trigger repeat on a given interval
RIGHT	Used in setting Fader pan
ROC	Refers to the *ROC 5* or *ROC 10* consoles
ROUTE	Assigns a Source Device number to a Destination Fader or Channel

SATURDAY Day of the week used in a Schedule Event Trigger
SCAN Checking for state or Fader input
SCHEDULE Used to denote Schedule Event Triggers
SELECT Used in selection routines
SET Used to set User Variables and Trigger States
SOFTKEY Refers to the Softkey Buttons on *Numix* console
SOURCE Com port that receives incoming data stream
START Used to begin the Talk Delay function
STATE Used in "if state of engine equals" tests
STEREO Sets Mix Minus bus to stereo
STOP Used to remove a clock text display from a console or stop the Talk Delay function
STORE Saves a value to a User Variable
SUNDAY Day of the week used in a Schedule Event Trigger
SURFACE Port number of console
SWITCH Same as BUS or RELAY

TALK Used to set a Talk Delay function
TEMPERATURE Sets the temperature sensor parameters
TEXT Defines a line of text for console
THURSDAY Day of the week used in Schedule Event Trigger
TIME Used to set the time of a Schedule Event Trigger
TIMEOUT Sets the number of seconds before which a timer will fire (same as DELAY or WAIT)
TIMER Numbered timer for system use
TITLE Sets the title in show selection routines
TOGGLE Defines an on Trigger as the Toggle Type
TRIGGER Define trigger event line
TRIM Sets the Fader input trim
TUESDAY Day of the week used in Schedule Event Trigger
UDP Specifies UDP socket address for text data destination
UNROUTE Refers to a route that is replaced by a new route

v Lower case as a prefix designates a User Variable name

WAIT Sets the number of seconds before which a timer will fire (same as DELAY or TIMEOUT)
WEDNESDAY Day of the week used in Schedule Event Trigger
WHITE A color used for text on *Numix II* color display
WHEN Specifies command to execute when channel goes off

YELLOW A color used for text on *Numix II* color display

z Lower case as a prefix designates a System Variable name