Viz Automation and Integration User Guide

Version 1.0

Disclaimer

Vizrt provides this publication “as is” without warranty of any kind, either expressed or implied. This publication may contain technical inaccuracies or typographical errors. While every precaution has been taken in the preparation of this document to ensure that it contains accurate and up-to-date information, the publisher and author assume no responsibility for errors or omissions. Nor is any liability assumed for damages resulting from the use of the information contained in this document. Vizrt’s policy is one of continual development, so the

content of this document is periodically subject to be modified without notice. These changes will be incorporated in new editions of the publication. Vizrt may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time.

Vizrt may have patents or pending patent applications covering subject matters in this document. The furnishing of this document does not give you any license to these patents.

Antivirus

Vizrt does not recommend or test antivirus systems in combination with Vizrt products, as the use of such systems can potentially lead to performance losses. The decision for the use of antivirus software and thus the risk of impairments of the system is solely at the customer's own risk.

There are general best-practice solutions, these include setting the antivirus software to not scan the systems during operating hours and that the Vizrt components, as well as drives on which clips and data are stored, are excluded from their scans (as previously stated, these measures cannot be guaranteed).

Technical Support

For technical support and the latest news of upgrades, documentation, and related products, visit the Vizrt web site at www.vizrt.com.

Created on

2/14/2024

CONTENTS

Chapter 1 ABOUT THIS GUIDE

Vizrt live production systems deliver impressive production power 'right out of the box'. Their ability to simplify and automate custom operations and workflows, and to leverage the features and content of other platforms in the ecosystem, is the icing on the cake. This guide introduces all of these capabilities.

Even if you are the hands-on, never-ask-directions type, please peruse this page. If any questions arise later, you may find the information here allows you to jump directly to the details you need with a minimum of reading.

For the purpose of clarity within this document the term "Vizrt live production system" refers to any of our systems, including not only TriCaster but also many other products.

- *PART I – OVERVIEW:* The introduction to the Vizrt ecosystem also explains the organization of this guide.

- *PART II – AUTOMATION:* Introduction to the Vizrt product macro system, including the Vizrt live video mixer and Viz 3Play macro implementations, all about macro editing and management, and discussion of the numerous ways you can trigger macros.

- *PART III – INTEGRATION:* This section covers cross-product and cross-platform integration and communication.

- *PART IV – APPENDICES:* Third Party solutions, a master Macro Command list, and a time-saving comprehensive keyword index.

PART I (OVERVIEW)

This section provides a high level overview of the different components of the Vizrt live production ecosystem, and serves a guide to your locating andutilizing those tools that will help you accomplish your production needs and creative goals.

Chapter 2 THE ECOSYSTEM

Vizrt live video production systems are everywhere. 'If only they could talk, the stories they would tell'; but wait, they do communicate! And not only with their siblings. Increasingly, third-party developers have prepared software and systems that can also talk to them, with manifold benefits.

Section 2.1 NO MAN IS AN ISLAND

Appealing as insularity may seem at times, John Donne had it right – "No man is an island". We are inextricably connected. This is truer now than ever. In the twenty-first century, technology multiplies, extends, and amplifies our connections enormously.

Perhaps that's why, four centuries later, we have the temerity to extend Donne's aphorism as follows: "No *system* is an island". To elaborate just a bit, the more efficient, flexible, and deep the connections between systems are, the greater the rewards in creativity and productivity.

Fluid data transfer between systems is obviously fundamental to modern video production and broadcast. Beyond that, systems with deeply integrated and open communication and control capabilities offer collaborative, efficient workflows and outcomes simply impossible by other means.

2.1.1 FAMILY AND FRIENDS

At Vizrt, we deeply respect these principles. Our systems are engineered with extensive, open and innovative integration in mind. They 'talk' to each other, to make your work more efficient *and* rewarding. In particular, the TriCaster and Viz 3Play product families, along with supporting Vizrt software and hardware products, enjoy a high-level integration.

We also appreciate the importance of 'family friends.' Our customers have diverse needs, and equally unique pipelines. Simple connectivity is important, but much larger workflow benefits can accrue from more advanced interaction. The better we 'play well with others', the more we all gain.

No doubt our obsession with integration accounts for the rapidly expanding collection of collaborative systems and software we refer to as 'the Vizrt ecosystem'.

Section 2.2 SYMBIOSIS AND COMMUNICATION

Communication in one or another form lies at the core of symbiotic relationships. Many Vizrt systems can 'listen' for input from external hardware control devices and systems in several forms. You'll find these discussed in Section 5.1.

Vizrt live production systems can also exchange a/v streams and metadata, system status details (including 'tally', audio VU levels, etc.), and control instructions with suitably prepared external systems and software.

For example, audio, video, media files, and system control commands can easily be transmitted bi-directionally between systems across a shared network. Third-party solutions can even 'cross-pollinate' with Vizrt systems, endowing the latter with dedicated custom macro commands specially designed to work with their product.

Chapter 3 AUTOMATION AND INTEGRATION

This chapter briefly explains various aspects of automation and integration as an aid to understanding the terms and technology discussed in this guide, and provides an overview of how the different elements in these areas work together to offer flexible solutions to satisfy your needs.

Let's briefly consider distinctions between automation and integration as the terms are used in this guide. This is a bit trickier than it might seem, but we want to make the effort as it will enable you to quickly locate the type of information you want without too much tedious searching.

Section 3.1 INTRODUCTION TO AUTOMATION

To begin, let's pay automation its due by spending a few moments on its virtually endless benefits. Even in its simplest forms, automation can render repetitive operations effortless and error-free at the same time.

The principle *native* engine of automation for Vizrt live production systems is the Macro system. The very same commands exposed for your convenience in this system account for virtually every operation executed by the system.

And, of course, macros can be combined endlessly, with full control over timing. Without much effort at all, you will be able to customize your Vizrt system to streamline your workflow and accommodate your personal preferences.

Hint: Chapter 4 provides a thorough introduction to the Macro system.

Without belaboring the point, obviously automation can be extremely simple, or more complex. For example, a simple macro might select a transition and perform an *Auto* to display a specific video source. For convenience, you might assign this macro to a keyboard shortcut.

A slightly more complex macro might load a designated *M/E preset*, select it on the main Switcher's *Preview* row, load a custom transition, perform an *Auto*, and reconfigure the *Audio Mixer* to match the changes. Again, a single keystroke, click, or button press can trigger all of this.

Hint: Vizrt live production products support numerous and diverse input methods for triggering macros – see Chapter 5 for details.

Let's raise the automation bar even more. Rather than relying on manual input to trigger a sequence of automated tasks, the entire process can be driven by external systems.

For example, an external software application serving as a master control process can execute all manner of complex production operations according to predefined scheduling or other stimuli.

A typical example of this type of automation would include newsroom MOS protocol implementations, which are used to create, edit, manage, schedule and execute virtually every audio, video and graphic element of on air segments in many news centers worldwide.

By mentioning this level of automation, we have wandered into the fuzzy boundary between 'automation' and 'integration'. Before moving on to discuss the latter, let's touch briefly on one special automation task.

In high end, mission critical production settings, failsafe systems are de rigueur. One of the more important redundant systems in such settings is the primary video mixer. Vizrt live video mixers have unique features that permit them to serve in these environments.

Section 3.2 Introduction to Integration

In this guide, we use the term "integration" when referring to broader topics, including cross-platform communication, data transfer and also multi-system control. This may all sound daunting, but some wonderful things are actually readily accessible thanks to 'smart' connectivity inherent in Vizrt live production systems.

Let's consider an example. You may already know that most Vizrt video systems provide native support for network sources. A/V signals supplied across standard TCP/IP network connections can be ingested live, just like any other source.

When one Vizrt system is linked to another, both are 'aware' of the live connection, and the two systems are able to 'converse' without any further configuration. This allows simple signal traffic, such as tally notification, to pass between systems automatically. More than this, however, the native *Macro* systems of both units allow operators to send instructions from one system to another.

Thus, a Vizrt live video mixer operator could easily create a macro that would i) jump the live video stream from a Viz 3Play networked source back 5 seconds, ii) select a custom "Instant Replay!" transition, iii) commence slow motion playback of the Viz 3Play recording, iv) *Auto* the network source onto *Program* output, v) select a "Back to Live!" transition, and vi) *Auto* back to the original source 15 seconds later – then execute the whole thing with a single click at any time.

A new and growing resource is NDI® which (among other things) allows you to access video streams among NDI enabled devices. Imagine your production switcher, capture system, media server—any NDI-enabled device on the network—can now see and access content from all other devices, allowing more sources than ever before to be used for live production.

Of course, integration with kindred Vizrt systems is just the beginning. The chapters in the Integration section of this guide (Part III) also discuss integration with third party systems and products too. This includes, as well, coverage of related topics such as file import and export, and drive formats.

Note: For your free NDI Tools bundle, visit https://ndi.video/tools/.

PART II (AUTOMATION)

Full details of the macro system native to the Vizrt live production family, along with an explanation of redundant control over Vizrt live video mixer systems.

Chapter 4 THE MACRO SYSTEM

Macros can smooth your workflow, reducing complex operations to a single button press, and making it easier to produce sophisticated programs. Macros can also eliminate embarrassing operator errors. Too, the fact that there are nearly endless ways to trigger macros provides many opportunities for both workflow streamlining and creative applications.

Keeping up with the action is one of the hardest things about live production. Fingers can only move so fast, and it can be hard to recall and perform important sequential steps without any slipups.

Macros are the answer to that dilemma. Record a sequence of events and play it back with one click. Or trigger it with a keystroke, control surface, MIDI panel or sequencer, your smart phone, automatically on a *HotSpot* action, on achieving a designated audio threshold, or many other alternatives.

Macros can do almost anything. Preload and play content, modify audio settings, automate multi-step sequences or perform synchronous operations. The amazing versatility of macros more than justifies the prominence the *Macros* button gets in the *Dashboard* area of most Vizrt live video mixer and Viz 3Play models.

Section 4.1 MACRO CONFIGURATION

Click *Macros* to show a menu which lists a *Configure Macros* item at the bottom. Select this menu item to open the *Macro Configuration panel,* which in turn enables you to create, edit, and manage your macros.

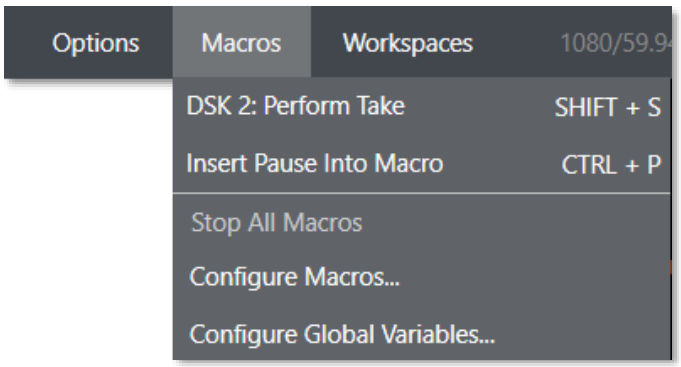


FIGURE 1

Note: You may notice some differences in layout of this panel from one product and model to another, but basic functionality is generally as described in this guide.

4.1.1 SYSTEM MACROS

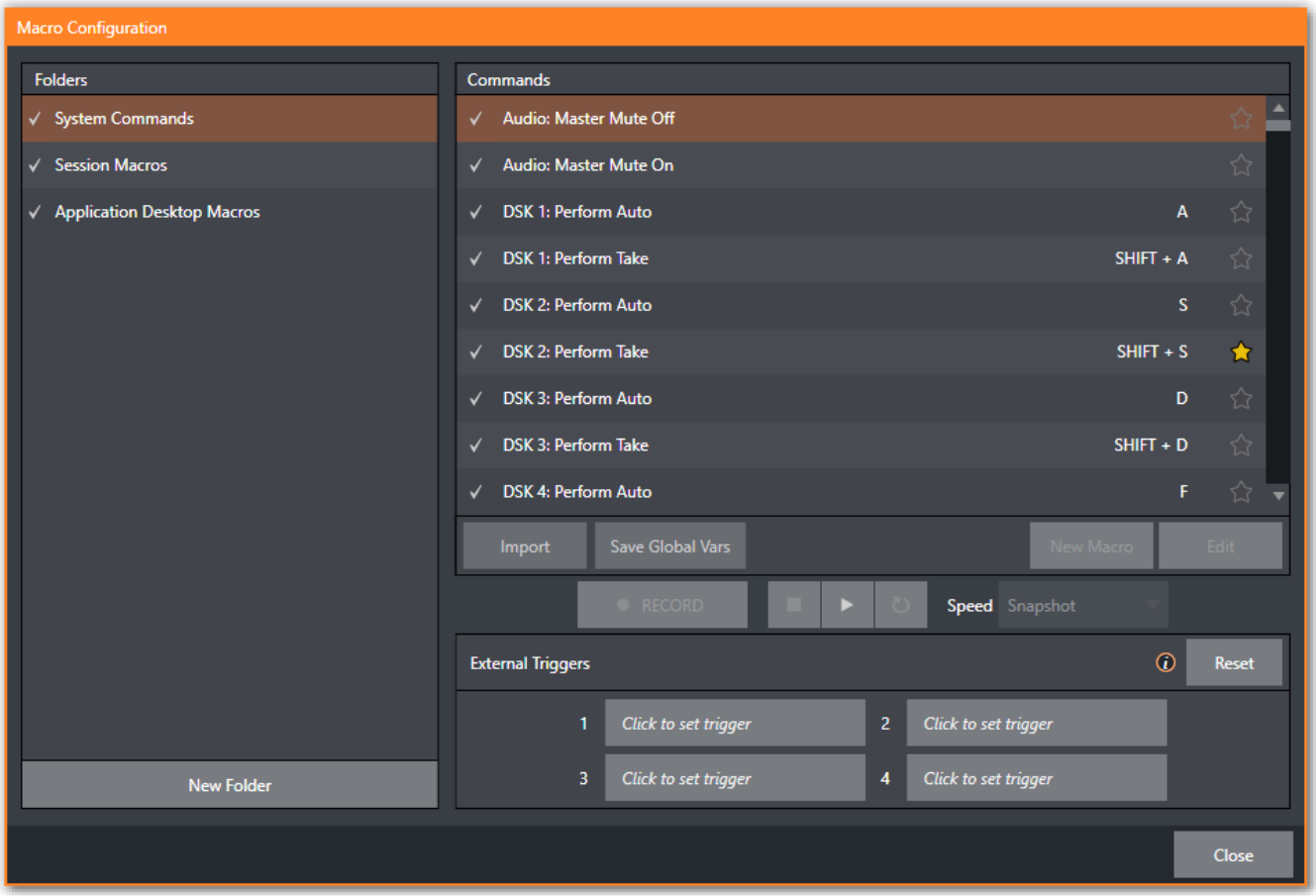


FIGURE 2

The largest part of the (resizable) *Macro Configuration* panel consists of the *Macro List*. By default, for any product, this list includes an uppermost entry labeled *System Commands*. Use the scroll bar at the far-right to see a lengthy list of these important macros. The macro entries in this group actually invoke the same shortcuts called by the user interface and *Control Surface* to operate your system.

Hint: Notice that keystroke shortcuts assigned to macro entries are visible at right.

It's worth noting a few unique aspects of *System Commands*. First, *System Commands* are specially safeguarded within the system. *Rename* and *Delete*, functions normally available from a right-click folder or entry context menu, are disabled.

Hint: If you copy of a System Macro outside that group, the copy becomes editable.

Individual entries in the list can be disabled by un-checking the switch at left; not surprisingly, removing the checkmark beside the *System Macros* folder itself will result in the failure of all 'system default' keystroke shortcuts. By design, this does not affect *Control Surface* operations, however.

4.1.2 SESSION MACROS

Session Macros is another macro folder that always appears in the list. Macros you create in or move into this special folder are available in the current session (only). This collection gives you a place to collect custom macros that are designed for use within a specific production without cluttering up the list.

Note: The Session Macros group itself cannot be deleted or renamed.

One advantage of the *Session Macros* implementation is that it lets you invoke session specific variants of a macro using the same keystroke shortcut (or MIDI surface button, etc.) without conflicts. For example, you might set up macros that behave similarly in every session, but which point to different content.

Hint: One effortless way to copy content from one Session Macros folder to a different session is to Clone the folder and rename it. Then launch the target session, and move the macros you want to transfer from the renamed clone into the current Session Folder.

4.1.3 APPLICATION DESKTOP MACROS

Application Desktop Macros have been specifically crafted to seamlessly integrate with the Live Call Connect Feature of TriCaster. This feature is supported in TriCaster 2 Elite, TriCaster Vectar, and TC1 Pro. These macros effortlessly incorporate your callers into your broadcast, supporting connectivity with various conferencing applications such as Zoom, Skype, Microsoft Teams, and more.

Section 4.2 RECORDING MACROS

Creating a new macro is simple. Selecting a folder in the list (other than the *System Macros* folder) enables the *New Macro* button. Click this button to add a new macro entry.

FIGURE 3

Continue to define the macro by clicking the *Record* button at the bottom of the panel, and perform the sequence of operations you wish to include in the macro.

You can use mouse, keyboard, and *Control Surface* operations when doing so. When finished, click the *Stop* button to complete recording.

Hint: Double-click directly on the name field for a folder or macro to edit it, or select Rename from the context menu.

Test the new macro by clicking the *Play* button (or by double-clicking the macro entry in the list). You'll notice that an animated bar in the background of the macro's entry in the list tracks playback progress. You can set macros to loop using the button at right, or modify the playback rate using the nearby menu.

Hint: You can record a macro that includes other macros. Depending on your order of operations, you may need to re-highlight the newly recorded macro in the list to show its Stop control (to end macro recording).

4.2.1 SNAPSHOT MODE

One option in the playback rate menu bears explanation: *Snapshot* is rather special. When you choose *Snapshot* as the macro's 'speed', you essentially tell it to jump to its end result. Any operation or delay that is irrelevant in achieving that end is simply omitted.

Snapshot mode is extremely useful for macros that configure Vizrt live video mixer to a particular state. For example, you might want to instantly reconfigure multiple *M/Es* with different angles of a single virtual set for an impending scene change; or perhaps you want to quickly disable *LiveMatte* for all *Media Players* at once. The possibilities are endless.

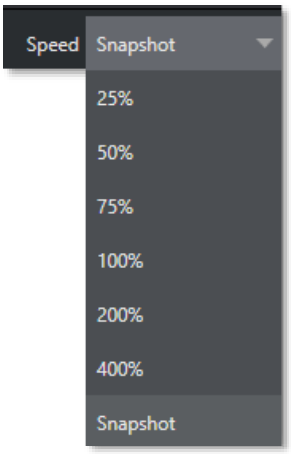


FIGURE 4

4.2.2 FAVORITES MENU

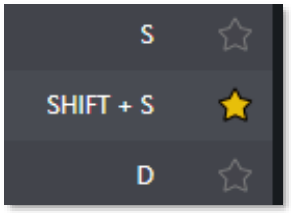


FIGURE 6

You'll see a 'star' gadget (Figure 6) at right for each macro entry in the *Macro Configuration* panel. Click the star to include the macro in the quick access *Favorites* list, shown in the Dashboard *Macro* menu (Figure 5).

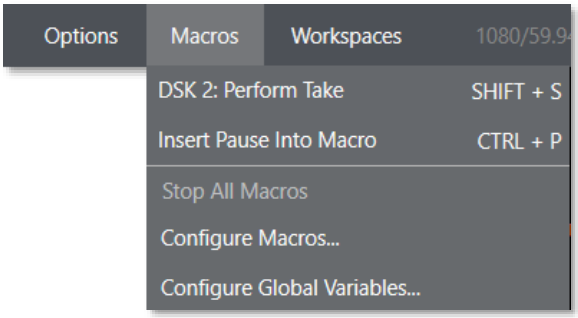


FIGURE 5

Section 4.3 Managing Macros

The *Macro Configuration* panel has numerous features to help you organize and manage your macros, including not only folders, but also rename, clone, copy and paste, and hotkey assignment, as well as *Import* and *Export*.

4.3.1 The Context Menu

Entries in the macro lister have a context menu, shown when you right-click an item. Menu items allow you to play or record a macro, delete or rename it. You can, of course, cut, copy, and paste macros, or clone them, combining the latter two operations in one step.

You can also export selections, including multi-selected macros, or even entire folders.

The corresponding import item is shown in the menu if you right-clicked either a folder or a blank area in the macro list pane. Import and export can be used to share macros with multiple users and systems, but provide another important service, too.

A good deal of time can be spent preparing complex macros designed to support your production. It would be a shame for these to be lost unintentionally through some mishap, as by some overly-tidy assistant deleting a folder on your day off (or by performing a *System Restore*). For this reason, we encourage you to use the *Export* feature to prepare a backup archive of your painstakingly designed macros.

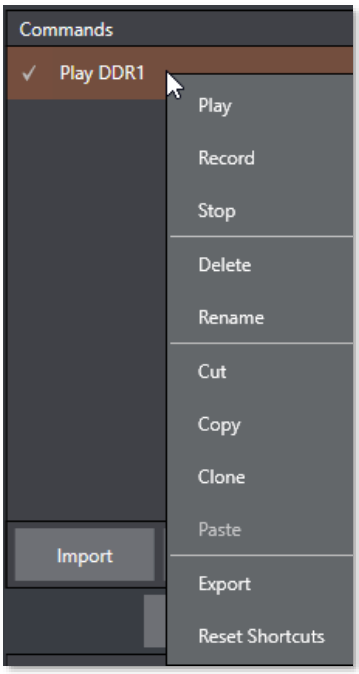


FIGURE 7

Section 4.4 Editing Macros

Often you will wish to modify values assigned to the various steps in an existing macro, rather than re-recording it; or you may want to experiment with other possibilities. Click the *Edit* button to open the *Macro Editor* for the currently selected macro.

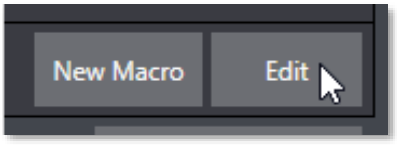


FIGURE 8

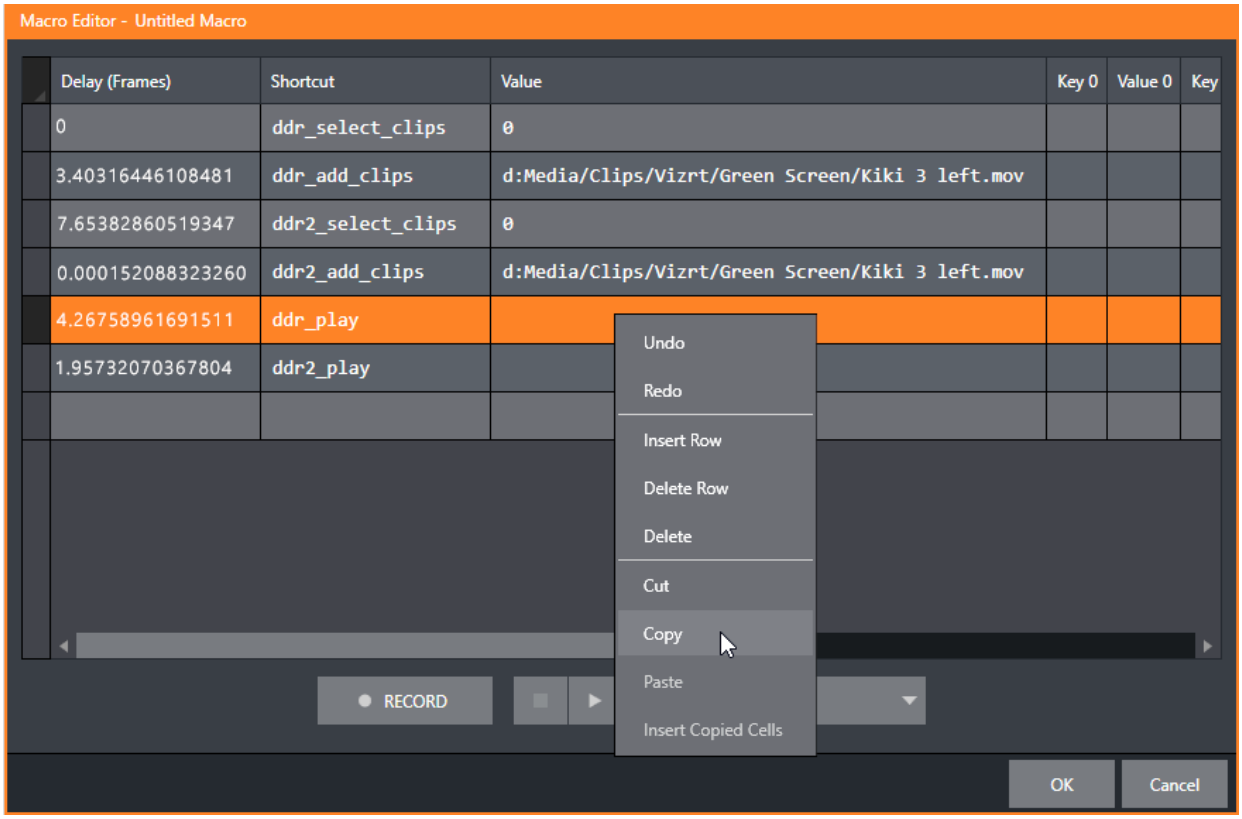


FIGURE 9

This deceptively simple editor presents the shortcut sequence your macro contains, along with all of its values, including timing information for each line, in a simple to comprehend 'spreadsheet-style' interface. Simply click a cell to edit the current entry, or use the arrow keys to navigate.

- Click any cell in the table to select it for editing or other operations. Or click the left-most cell to select a row. Select multiple rows using Shift or Ctrl modifier keys in the usual manner.
- Right-clicking opens the editor's context menu, which allows you to *Undo*, *Redo*, *Insert* a row (the keyboard shortcut Ctrl + i also inserts a row), *Delete*, or *Cut*, *Copy* and *Paste* selections.
- Standard copy and paste keyboard shortcuts are supported as well. When done editing a macro, click *OK* (or *Cancel*, to close the editor without saving your changes).

Hint: Use the Record button in the footer of the Editor to directly record new entries to be inserted into the current macro at the selected line.

Section 4.5 More About Macros

Macros consist of a *shortcut*, or command, and (at times) one or more *values*. Let's consider an example. The shortcut *grab_still* is easy to use. You can simply enter it as shown in and run the macro.

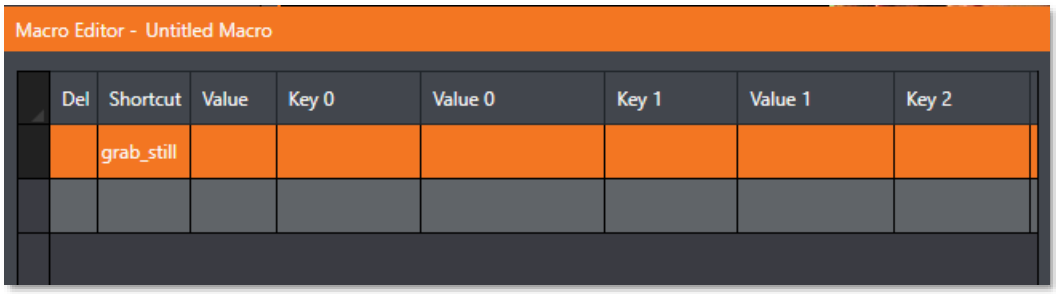


FIGURE 10

In this case, the result will be identical to what would happen if you clicked the GRAB button in the interface of a TriCaster. The grab operation will use the current settings from the *Grab Still Configuration* dialog. Macros can do much more, however. Let's see what the *grab_still* shortcut is really capable of.

4.5.1 UNDERSTANDING SHORTCUTS

TriCaster and Viz 3Play deliver an integrated LivePanel webpage that includes a handy Shortcut Commands reference page.

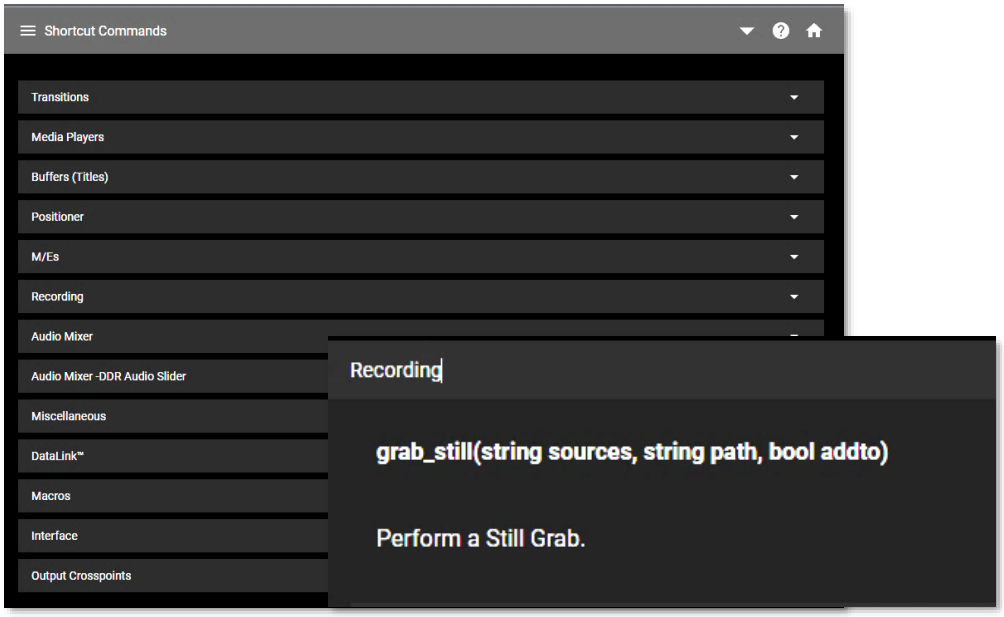


FIGURE 11

Shortcut commands are listed under expandable group headings, with supported arguments (keys and datatypes) shown. As the webpage is served by the system, this list is always up to date.

Those who want *more* detail about a shortcut can exit to Windows®, and locate the file named shortcuts_cfg.xml. Depending on your product, you will be able to find this file in the (hidden) directory C:\ProgramData\Vizrt*product-name (e.g., TriCaster)* \Configuration\. You can view the content of this file in most web browsers, or you might prefer to view it in a text editor.

Hint: You may wish to save this file to another system for reference purposes, but note that the content of the file changes from time to time as new features are added.

Either way, if you search the file for "grab", you will be able to locate the lines below, which include a *dictionary* that defines the properties of the *grab_still* shortcut.

```
<shortcut name="grab_still">
  <shortcut_dictionary>
    <entry key="sources" type="string" />
    <!-- ',' separated sources to capture, the pattern is [output/input]+[index], like
     "output1,input1", if this is not set selected sources in the combobox will be used-->
    <entry key="path" type="string" />
    <!-- the root path to store the files, if this is not set then the default session's still path would
     be used-->
    <entry key="addto" type="bool" />
    <!--whether or not add images to the selected destinations-->
  </shortcut_dictionary>
```

Hint: See also Exporting Macros in Section 7.2.1.

KEYS AND VALUES

Notice that the *dictionary* (in our example above) defines three *entry keys*, and explains their use:

- A key labeled *sources* allows us to specify which sources to grab. If we supply values for the *sources* key, we can tell *grab_still* which specific sources to grab from.
- Another key, labeled *path*, allows us to tell the system where to store the grabbed files.
- Finally, the key *addto* tells the system whether or not to automatically add the grabbed files to the current playlist of one or more *Media Player* modules.

As you can see, values can take different forms. The sources and path keys take text values (*string*), while the dictionary tells us that *addto* takes a *boolean* (0 or 1, true or false) value. Let's try these out.

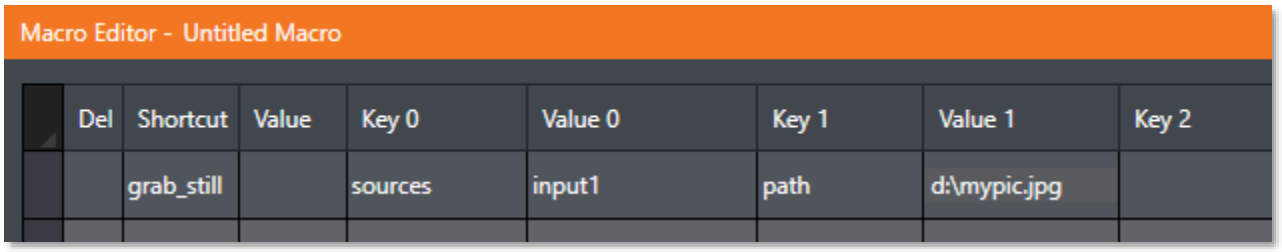


FIGURE 12

In Figure 12 we added *sources* and *path* keys to our example, and gave them values. Executing our macro grabs an image from the source of switcher Input 1, storing it on the D:\ drive with the base name "mypic".

Hint: Note that the sources key supports values that are not available in the Grab Configuration panel. It is not unusual for a shortcut to provide extended capabilities in this manner, which is another advantage of macros.

PREFIXES, SUFFIXES, AND TARGETS

Grab_still is quite a simple *shortcut*, really just a single command. Many other *shortcuts* consist of two parts, a *prefix*, which is prepended to the shortcut string to identify its target, and the *suffix*, (typically the *shortcut* or command itself). Syntax for a slightly more complex shortcut is thus as follows: Prefix_shortcut (value).

For example, the "_auto" shortcut requires a *prefix*, but does not require a *value*. A prefix tells the shortcut what the target of the instruction is. Recording a macro will often capture the prefix you need, but in any case you can search for "prefix" in shortcuts_cfg.xml to find a lengthy list of applicable prefixes.

This list includes the examples below (among many others):

- "main"
- "main_background"
- "main_dsk1"
- "v1_dsk1"

If we add the prefix *main* to the shortcut *_auto*, we get *main_auto*, which will perform an auto (transition) on Vizrt video mixer's main *Switcher*. In this case, the auto will be applied to all delegated video layers, just as if you clicked the *Auto* button below the *T-Bar* in the interface. However, other prefixes let you apply the shortcut to individual layers.

For example, a macro containing the compound shortcut *main_background_auto* exclusively applies the auto to the Program/Preview layers, ignoring other delegated video layers. Similarly, *main_dsk1_auto* affects only the first DSK video layer, while *v1_dsk1_auto* is limited to the first key layer for M/E 1 (a.k.a., v1).

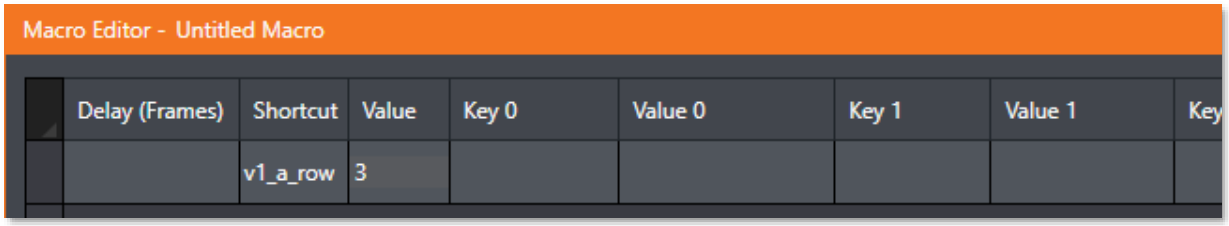


FIGURE 13

In some cases, a *value* further refines a shortcut. For example, "v1" is the prefix that targets M/E 1. The index value "0" identifies the first video input. Thus, supplying the shortcut *v1_a_row* with a value of *3* selects Input 4 on the A row of M/E 1.

As we have seen in this section, it is simple matter to record a macro (4.1.3), and tweak it using the Macro Editor (Section 4.4). This is usually the quickest approach to finding a macro shortcut (command), any prefixes it requires, and the sort of values that it requires. The shortcuts_cfg.xml file provides deeper detail that can be helpful at times.

4.5.2 MULTI-STEP MACROS

The *Macro Editor* permits you to create and execute multi-step macros. Adding the line "#waitforcontinue" (or simply, "#pause") to a macro using the *Macro Editor* causes the macro to wait for user input at that step in its execution.

The *Continue Paused Macro* shortcut, assigned by default to the *backtick* key (`) serves to resume playback. This feature can be used in endless ways, for example to allow a user to step dynamically through a series of animated CG overlays on demand.

4.5.3 USING VARIABLES

At times it can be extremely powerful to be able to employ variables in macros. For example, consider a complex macro which defines the target for a shortcut with a variable. This would allow a large number of instructions in the macro to be retargeted by simply revising that one variable's value.

Variables are evaluated to determine their current value each time they are invoked. Thus, even if you update the value for a variable during the execution of a macro, the new value is used each time the variable is encountered as the macro proceeds.

GLOBAL AND LOCAL VARIABLES

There are two similar but distinct types of variables:

- **Global variables** can be invoked in any macro you execute.
  - They can be created and their values can be set or modified in two ways:
    - You might use the convenient Configure Global Variables dialog (which you can open from the Macros menu in the Dashboard of supporting live production systems).
    - Alternatively, create them using the shortcut "set_global_var" with a value like "myvar=4" (both without quotation marks)
  - Global variable names and values persist within a session - even after you shut down the system.
  - A "Save Global Variables to Macro" option appears in the context menu of the Macro Configuration window's right-hand pane. Better yet, the "OK" button will save the contents while closing the window.

    Either method copies the current global variables into a macro for export, editing or later recall.

- **Local variables** are 'transient', only existing during the execution of a macro. Hence they are exclusively created and populated using the shortcut command "set_local_var" (without quotation marks) within a macro.
  - NOTE: Although we recommend unique names, if a local variable and a global variable have the same name, the local one is used.
  - While you might think a local variable is only useful in the macro it is defined in, there is an important exception: Local variables extend to macros run by the same parent macro.
    - This lets you write 'generic' macros with targets or values to be provided later. For example, a macro might include a shortcut like ddr{DdrNum}_play, with no value supplied DdrNum. However, if you run this macro from one which does provide the value, the child macro derives the necessary value for DdrNum from the parent.
  - Outside the defining macro and those it spawns, though, local macros 'have no life.'
  - You could even use a single local variable name in simultaneous macros with different values.

Hint: At times it can be useful to convert a global variable to a local one. For example, you might want to increment a value as a macro runs without affecting the global variable's value. You can do this by copying the global value to a local variable, and referring to the local variable on subsequent lines. (Conversely, a similar approach would allow you modify a global value once a macro begins, without affecting the local variable.)

Common Features and Use

- Variable names may not start with a number, contain spaces, or non-alphanumeric characters (with the exception of the underscore character, _.
- Variables values can be strings or integers (similar to DataLink).
    - Note that string or math operations will fail when applied to values having the wrong data type.
- Variables within curly brackets are evaluated and replaced by their current value when parsed in a macro.
- Variables can be used anywhere in a macro – even in the middle of a word or shortcut.
    - For example: if the variable DdrNumber holds a suitable value, ddr{DdrNumber}_play will cause the DDR identified to begin playback.
- The value of a variable is newly evaluated by the macro processor each time it is accessed
    - For example: If the current value of variable named myvar is 5, when the macro processor encounters {myvar} during execution, it treats it as a 5.

DataLink and Variables

The shortcut set_global_var can also set Datalink variables.

Hint: The % symbols are required to identify DataLink variables.

Examples:

- %AwayTeamScore% = {away_score}
- %MyDlText% = {somevar} {somevalue} {thirdthing}

You can also copy values from Datalink keys to variables.

- For example:
    - {%Time%} expands to the current time value.
    - myvar = {%Time%} assigns the current time to myvar
    - Since variables are re-evaluated whenever they are used, assigning {%Time%} as the value of a variable 'locks' it to the time you set the value. It will not change thereafter without deliberate manipulation.
        - By contrast, entering {%Time%} directly at different points in a macro (rather than using a variable) produces a different value each time it is processed.

Hint: Variables enclosed in 'curly brackets', such as {myvar}, are evaluated whenever the shortcut executes. This means that even something like {%my_datalink_key%} will work, allowing you to use a DataLink value in places where the code normally would not have access to it.

Simple string manipulations are supported. For example, If the variable myvar contains the string "Bob", then myvar = {myvar} Smith assigns the value Bob Smith to myvar (characters outside the curly brackets are treated as literals).

To perform more advanced string or math functions, you can use EXP( ) to indicate you're employing an expression. For example:

- %HomeTeamScore% = EXP({%HomeTeamScore%} + 1)

> **Hint: You can (and often must) force values to be treated as strings by wrapping them in apostrophes**. (This can also be used to pad entries with spaces, which might otherwise be automatically stripped.)
>
> For example, if %HomeTeamScore% holds the value 1, the entry %HomeTeamScore% = exp({%HomeTeamScore%} + 10) would assign the numerical value 11 to HomeTeamScore. By contrast, %HomeTeamScore% = EXP(%HomeTeamScore% + ' 10') would give it a string value as follows: 1 10.

A number of familiar math and string operators are supported in expressions. These include:

- + (addition)
  - For example, if myvar holds the value 2, invoking set_global_var with the value %dlvar%=EXP({myvar} + 2) assigns the value 4 to %dlvar%.
- - (subtraction)
- * (multiplication)
- / (division)
- % (modulus)
- TRIM: Used to remove all leading or trailing white space characters or escape sequences such as \r, \n, \t,
- SUBSTRING: Yields a string of the length specified, starting from the point specified in the string.
  - For example, EXP(SUBSTRING('{myvar}',2,3)) yields a value comprised of three characters starting from the second position in the string contained by myvar (using apostrophes to ensure a string value).
- LEN: Provides the number of characters comprising the string supplied in the form LEN(string)
  - For example, if the value of myvar is abc, EXP(LEN('{myvar}')) yields 3, the number of characters in myvar's value.

Comparing Values

IIF is a powerful function that allows you to determine whether another expression is true or false, and return different values for each case. For example, EXPR(IIF ('{myvar}'='Bob Wins', 'Bob is Happy!', 'Bob is Sad!')) returns Bob is Happy if myvar contains Bob Wins.

IIF supports the following comparison operators:

- < (less than)
    - For example: EXP(IIF({myvar} < 3, 'True', 'False')) will supply "True" when the variable myvar holds the value 2 (etc.)
- > (greater than)
- <= (less than or equal to)
- >= (greater than or equal to)
- <> (does not equal)
- = (equal to)
- LIKE (compare strings)
    - Note that LIKE can use the asterisk (*) as a wildcard when it leads or trails the exemplar string. For example:
        - EXP(IIF('{myvar}' LIKE 'DD*', 'This is a DDR', 'This is not a DDR')) will supply "This is a DDR" when the variable myvar holds any value that begins with "DD".

Hint: While EXP(IIF('{myvar}' < 3, 'True', 'False')) Math operators may fail or provide unexpected results when applied to string values, even when you wrap the value in apostrophes.

Chapter 5 TRIGGERING MACROS

Macros can reduce or eliminate embarrassing errors. Too, the fact that there are nearly endless ways to trigger macros provides many opportunities for workflow streamlining and creative applications.

As discussed in the previous chapter, one way to execute a macro is directly from the *Macro Configuration* panel, by double-clicking a macro entry, or by clicking the *Play* button.

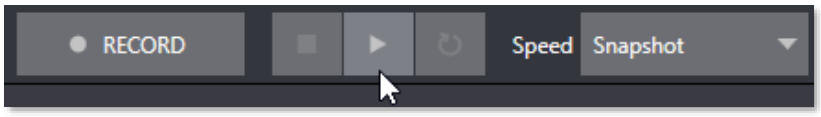


FIGURE 14

This is just the beginning, however. Macros can be triggered in so many ways that we've devoted a whole chapter to the topic.

For example, macros can be triggered by the following means:

- Keystroke shortcuts
- Control surface buttons
- MIDI pads and sequencers
- GPI signals
- Software events such as:
    - A Vizrt live video mixer *HotSpot* 'hit'
    - An audio event
    - Or input state change.
- Third-party software applications communicating with your Vizrt live production system over a network
- HTTP commands sent by a webpage designed for the purpose

Section 5.1 HARDWARE

As mentioned earlier, macros can be triggered by any of a wide array of supported external hardware devices. Obviously, this includes your keyboard; and the majority of Vizrt control surfaces compatible with your live production system have a *Macro* button for this purpose.

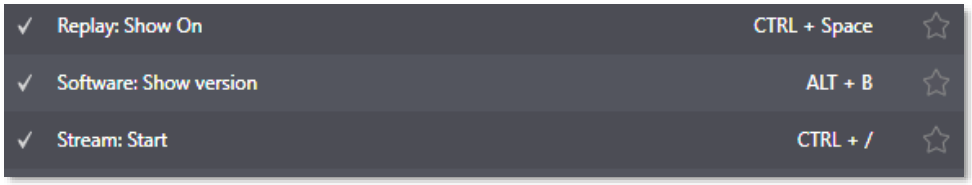


FIGURE 15

As the simplest example, let's briefly consider keyboard shortcuts before looking into some of the other options.

Of course, many of the *System Commands* entries have default keystroke shortcuts pre-assigned. The first shortcut assigned to a macro (some systems support multiple shortcuts) is displayed at right on the row, near the *Favorites* star mentioned earlier.

5.1.1 KEYBOARD SHORTCUTS

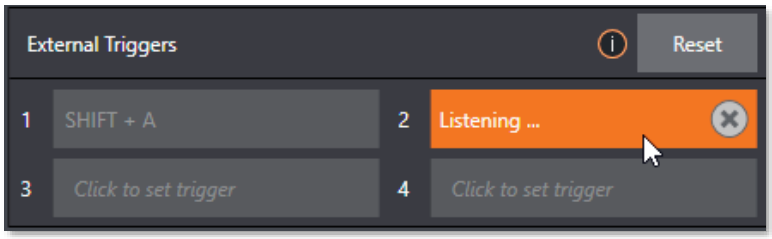


FIGURE 16

To set a new shortcut or replace an existing one, click a 'gesture field' in the *Shortcuts* group at the bottom of the *Macro Configuration* panel. It will display a "Listening …" tag. Then press the desired keystroke.

Hint: For clarity, lower-case characters are uniformly shown as upper-case. True upper-case letters are displayed in the form [Shift + (character)].

FEELING CONFLICTED?

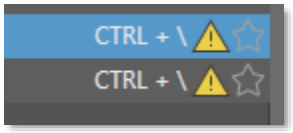


FIGURE 17

By the way, assigning identical shortcut combinations to multiple macros *is* supported, and deliberately so. Still, as you may wish to avoid conflicts, a yellow triangular gadget referred to as a 'bang' (or, if you are a 'foodie', a 'nacho') is shown in this case.

Bangs appear at right for all macro entries in the *Macro Configuration* panel lister with shortcut conflicts (Figure 17).

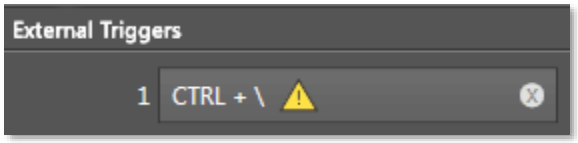


FIGURE 18

Of course, when multiple shortcuts are assigned, the first shortcut for a macro – i.e., the one displayed at right in the *Macro Configuration* panel lister – may not actually be the one that is conflicted; or there can be several conflicts for a single macro.

In such a case, select the macro in the list to show the corresponding *Shortcuts group* entries at the bottom of the panel (Figure 18). Conflicted 'Listen' controls will *all* show bangs. Clicking a bang automatically jumps to the next conflicted entry, so you can advance quickly through the list resolving conflicts as you go.

Obviously, you can resolve a conflict by assigning different keystrokes to conflicted macros. Or you can disable conflicted macros if you prefer, using the checkmark switch.

Hint: Folder level checkmark switches offer a method for managing 'deliberate' shortcut conflicts. For example, the shortcuts assigned to entire folders of macros designed for various purposes can conflict with shortcuts in another folder, but keystrokes for any inactive folders will be ignored.

DELIBERATE 'CONFLICTS'

On the other hand, your 'conflicts' may be deliberate; running multiple macros with just one button press or gesture may be just what you had in mind. Pressing the conflicted shortcut key will perform *all* macros sharing that keystroke assignment.

5.1.2 VIZRT CONTROL SURFACES

FIGURE 19

Several Vizrt manufactured control surfaces feature a dedicated *Macro* button. When this is true, a macro can be assigned to buttons on the control surface in much the same manner as it would be assigned to a keyboard button. You would simply do as follows:

1. On Vizrt live video mixer's Live Desktop, open the *Macro Configuration* panel.
2. In the macro list, select the macro you wish to assign to a button.
3. Click the mouse in a *Listen* control at the bottom of the panel.
4. Hold down the MACRO button (on the control surface) and press the button you want to assign the macro to.

That's it – close the *Macro Configuration* panel, and test the result.

Hint: When you press the MACRO button, all buttons that currently have assignments light up. This makes it easy to see which buttons are available for your use.

5.1.3 MIDI CONTROLLERS

The MIDI (Musical Instrument Digital Interface) protocol and devices and systems supporting it offers another extremely useful (and often very affordable) macro trigger option. MIDI devices are often used in the audio and events industries, but can be found in other realms as well. Thousands of devices and systems of this sort are available.

The *Macro Configuration* panel system can 'listen' for button presses from most MIDI devices, just as it recognizes input from the keyboard or native control surface.

Note: Many MIDI devices provide 'plug-and-play' convenience. Some, though, require non-standard device drivers. Generally, adding device drivers to Vizrt products is discouraged, since these may not have been prepared with the rigorous demands of live production in mind.

If you install a driver and encounter unintended consequences, you can resolve the problem by restoring to factory defaults and, if necessary, updating to the current software version appropriate for your Vizrt system.

Too, a wide variety of MIDI software and extensions are available for various platforms, including mobile devices such as tablets and smart phones. These can be used to create unique custom Vizrt live video mixer control alternatives. See the control surface documentation and addenda for your live production system for more on this topic.

5.1.4 GPI CONTROLLERS

GPI, or General Purpose Interface, is a long-serving analog control signal system based on simple contact closure. GPI inputs and outputs are quite common on professional production equipment. The macro system in Vizrt live production devices can take advantage of intermediary devices, such as the eBOX™ network/GPI hardware interface from JLCooper Electronics, to support both GPI signal input and output.

CONFIGURATION

For an external GPI device to communicate with a Vizrt live production system, it must be manually defined by text entries in the file named gpi_setup.xml. This file can be located in the directory shown below as appropriate for your platform:

- C:\ProgramData\Vizrt\ProductName(e.g., TriCaster)\Configuration\

The entry for a given GPI control device must contain an IP address and port, password, and custom name, entered as follows:

< *device* name="*name* " ip="*###.###.###.###*" port="*##*" password=" "/>

At the time of writing, the xml 'element name' signified above by the placeholder *device* should be "jlcooper", without the quotation marks. The value for the "name" attribute that follows is a custom name of your choosing.

Hint: Normally, connected GPI devices are identified by unique names in this file; otherwise (if GPI devices share a single name) GPI commands are issued to them simultaneously.

The remaining configuration attributes ("ip", "port" and "password") are set at the external hardware device (refer to the vendor's documentation for details); the corresponding values need only be transferred into the XML configuration file. A typical entry might look like this:

<jlcooper name="JLCooper1" ip="192.168.128.102" port="23" password=""/>

Listening for GPI Triggers

Just like keyboard shortcuts, control surface and MIDI button operations described earlier, properly configured and connected GPI devices can trigger macros. To assign a GPI trigger to a macro, simply click a 'gesture field' in the *Shortcuts* group at the bottom of the *Macro Configuration* panel (Figure 16); then send the desired external GPI trigger to the system. The 'listening' control will record the GPI signal, and a suitable shortcut entry will be displayed.

Sending GPI Commands

A special macro command allows you to send GPI signals to external devices and systems via network-connected GPI interface devices (such as the eBOX from JLCooper Electronics). GPI macro entries are formatted as shown below:

Delay (ms)	Shortcut	Value	Key 1	Value 1	etc.
####	gpi	*name*	*GPI_pin#*	*boolean*	

- **Delay** – the interval, in milliseconds, between the time when the command on the prior line (if any) was issued to the system, and execution of this line.
- **Shortcut** – Use the entry "gpi" in this field to send a GPI signal.
- **Value** – The shortcut value is the name of the GPI device (defined earlier in gpi_setup.xml) that you want the signal defined on this line to address.
- **Key # (0 – *n*)** – The value you enter in this field identifies a target pin on the external DVI device to receive a signal defined in the following field.
  The entry should be formatted as "pin#" (e.g., "pin12", without quotation marks).
- **Value # (0 – *n*)** – This value controls the contact closure state (on or off) for the GPI device pin identified by the preceding key. The value can be entered variously as "1" or "0" **,** "on" or "off", "true" or "false" (without quotations).

A typical entry might look like the following:

Delay (ms)	Shortcut	Value	Key 1	Value 1	etc.
500	gpi	jlcooper	pin12	1	

Hint: Multiple GPI pins can be targeted simultaneously by key/value pairs entered on a single line.

Alternatively, some GPI devices require a GPI 'pulse' of a specified duration. In such a case, you might send an "on" command on one line, followed – after a suitable delay – by an "off" command sent to the same pin.oftware

As mentioned earlier, macros can also be triggered by software events of various types, including internal or interactive events such as a Vizrt live video mixer HotSpot 'hit', audio event or input state change, or externally, in response to commands from software applications or even a webpage designed for the purpose.

5.1.5 SWITCHER STATE

*T*he *State Change* controls located in the *Automation* tab of Vizrt live video mixer's *Input Configuration* panel allow you to flexibly trigger macros based on the utilization of video sources used in your production.

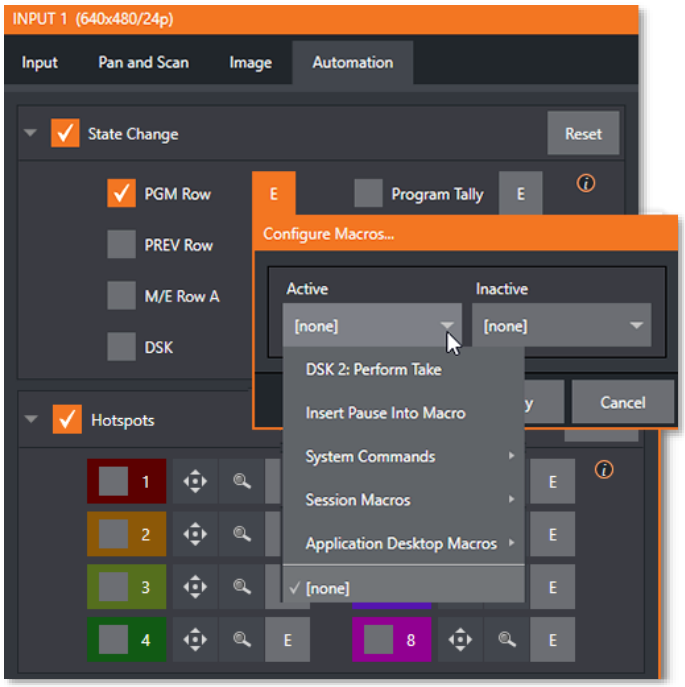


FIGURE 20

Macros can now be assigned to run on specific *Switcher* operations, such as:

- *Program* or *Preview* row selection
- Displaying/ hiding the source in a *DSK* or *KEY* channel
- Selecting/de-selecting it on an M/E's A row
- Or any M/E row, or …

- Showing or hiding a source on the *Program* or *Preview* output.

This feature is immensely powerful, and lends itself to all manner of applications, such as the following, to name just a few:

- Automatically fly in a title whenever you switch to a specified remote source.
- Then remove it again automatically after it is displayed for a specified time.
- Or automatically select a different *Audio Mixer* preset when you switch from viewing a source in the B monitor of a virtual set on *Program* to displaying it full-screen.
- And then change back to the original audio setup when you switch back to the anchor desk.

The possibilities are truly endless.

Simply click the E (Event) button next to a 'state' option for the input and select macros that will run when the source assumes or exits the specified state.

5.1.6 HOTSPOTS

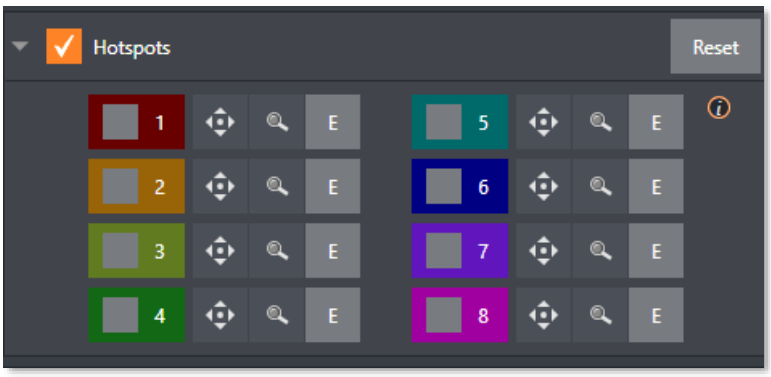


FIGURE 21

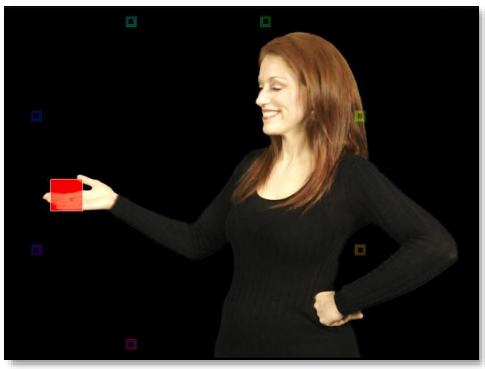


FIGURE 22

On-screen *HotSpots,* also configured in *Automation* tabs, provide yet another way to trigger macros – based this time on activity detected in specially defined regions of the video frame.

Hotspots can serve many purposes. For example, onscreen talent can trigger one macro by moving their hand (for example) into a *Hotspot*, another by moving it out.

- Use live action to play sounds, make *Overlays* and *DSKs* appear auto-magically, or switch the video in a virtual monitor by tapping it with a fingertip.
- Switch from a seated desk shot to a standup virtual set simply set by walking into it; then auto-switch to the next shot when you walk back out of the frame.
- Load up a different *DDR MEM slot*, audio configuration and camera assignments when talent moves from the desk shot to standup in a virtual set.

Hotspots are configured in the configuration panels for individual inputs. Double-click the viewport for a camera input to open this dialog, and click the *LiveMatte* tab. The lower portion of this tab contains the *Hotspots* control group.

Note: All Hotspots for an individual source can be enabled or disabled using the switch in the group header, or globally for all sources in a given session using the Options menu.

Hotspots are color-coded, and their respective colors are used to draw the *Hotspot* overlay boxes on your viewports when the *Hotspot Markers* overlay is enabled for a corresponding monitor viewport.

Scale and *Position* buttons allow you to re-size and place the 'trigger zone' for each *Hotspot* accurately.

The *Event* button for each *Hotspot*, marked by a capital "E", opens the *Event Triggers* dialog. This is where you will assign macros that are triggered when something moves into (On Screen) or out of (Off Screen) the otherwise transparent area defined by that *Hotspot*.

FIGURE 23

Hint: Use the Overlay option Flip View Horizontal to let talent see exactly where their marks are on a Multiview screen.

5.1.7 MEDIA PLAYER MACROS

Naturally, *Media Players* get *Macros* support like other *Switcher* inputs, as described above. We didn't stop there, though. *Every* item in a playlist – each clip, still image, audio file or title page – has its very own *Macros* features.

- Any macro you can record or create can be executed automatically on either playback or end of play for any and every individual playlist item.
- Improved multi-selection support in the playlist makes it a breeze to assign macros to multiple items.
- Automatically show titles for certain types of clips and not others.
  - Give them different title page types
  - Use macros to selectively adjust *Proc Amps* on a per-clip basis.
  - Or enable *LiveMatte* keying automatically when needed for certain items.

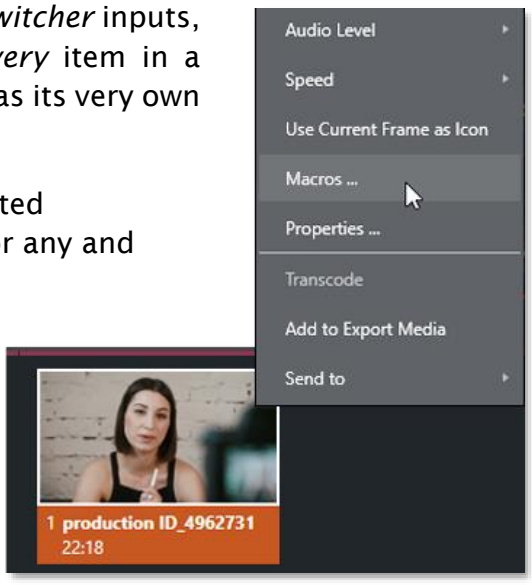


FIGURE 24

5.1.8 AUDIO AUTOMATION

Vizrt live video mixer's Audio Mixer provides similar automation functionality. The related controls are located in the *Triggers* group in the *Input Settings* tab of the advanced *Audio Configuration* panel.

Hint: To open this panel, roll the mouse over the control group for any input in Vizrt live video mixer's Audio Mixer, and click the gear gadget that appears at the top in the label for that input.

The *Triggers* control group contains *Follow Program Video* (also known as 'AFV', for Audio Follows Video) and *Macro* options.

FOLLOW PROGRAM VIDEO

Enabling *Follow Program Video* options for an audio source directs the Vizrt live video mixer to track switcher operations affecting the related video source.

Audio for sources with *Follow Program Video* enabled in the *Audio Configuration* panel is automatically removed from mixed outputs until one or more specified video sources are actually displayed on *Program* output.

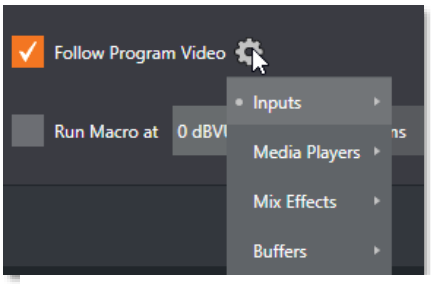


FIGURE 25

RUN MACRO

Audio threshold triggers allows you to specify a value in decibels to serve as a macro trigger. Whenever the sound level on that input rises above the threshold (or falls below it), designated macros will run. In this manner you could, for example, automatically perform a 'hands-free' camera switch to show someone who begins speaking, and then switch back again when he stops.

Hint: Transient sounds such as a brief cough are automatically filtered out.

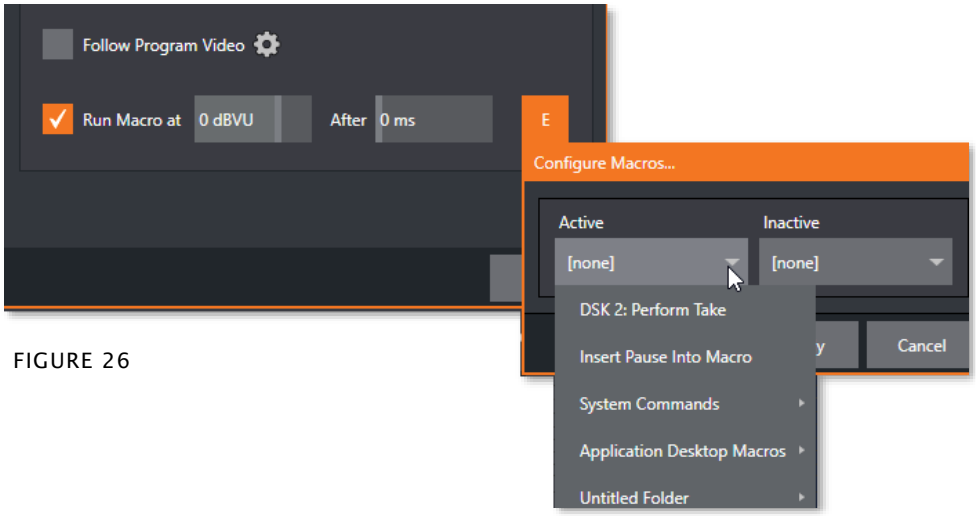
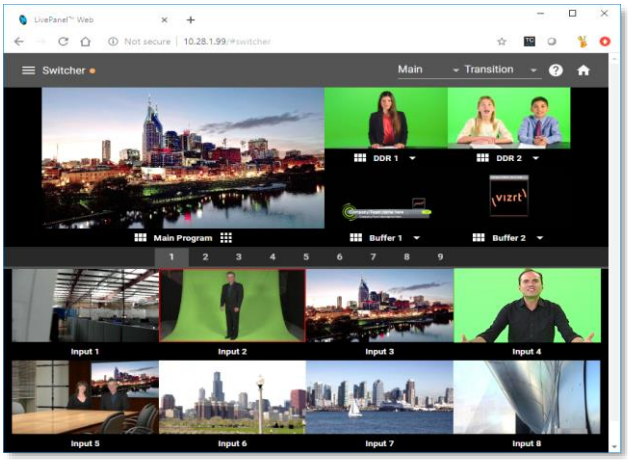


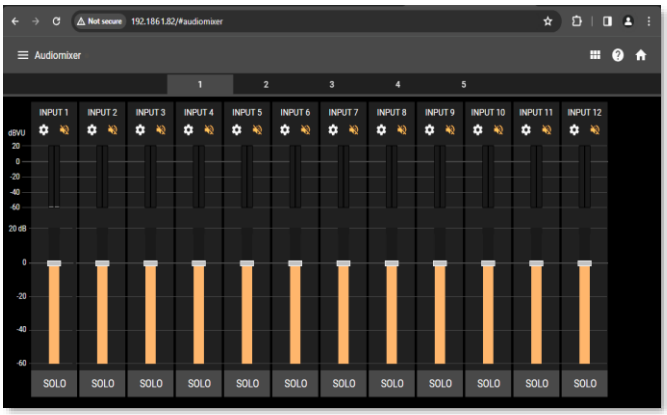
FIGURE 26

Section 5.2 NETWORK

5.2.1 LIVEPANEL

Viz's LivePanel application (available for selected systems as a standalone purchase or included with Premium Access membership) provides a wide variety of prepared web applets.





You can use these to switch a show, adjust audio levels, update DataLink values in your title pages, trigger macros, and much more. Configuration is as simple as entering the appropriate URL into your web browser, and you can even create custom webpage applets of your own using the included LivePanel Builder.

5.2.2 NDI CONTROL AND MORE

Additional means for communicating and controlling Vizrt live production systems over a network, including macros that use HTTP, NDI communication, and other methods relevant to developing custom applications that use TCP/IP or HTTP protocols, are discussed in Section 7.2 of PART III (Integration).

PART III (INTEGRATION)

in·te·gra·te [in-ti-greyt] – verb:

1. to bring together or incorporate (parts) into a whole.
2. to make up, combine, or complete to produce a whole or a larger unit.
3. to unite or combine.

Chapter 6 DATALINK

Section 6.1 INTRODUCTION

Vizrt live video mixer's integrated title page system provides many opportunities for internal automation and broader pipeline integration.

Its unique DataLink implementation combines with native as well as third-party automation systems to supply text and image updates for CG purposes from a wide array of live and prepared data sources.

Vizrt live video mixers include internal *Media Players* and native title systems. These are coupled to a world-class effects engine so you can display colorful and informative titles and graphics into your productions with ease and flair. And, as discussed in earlier chapters, the macro and automation features add to the value of this implementation:

- The display of title pages can be automatically controlled and timed based on diverse trigger events and Switcher states.
- Using the integrated title editor, text and image content of title pages can also be manually updated while live.
- Third-party software solutions can also control and update title page text and image content.

In addition, DataLink technology extends these capabilities, by providing realtime updates from a wealth of data sources.

6.1.1 LIVETEXT

Let's consider its best known prior role as background. The optional *standalone* version of Viz LiveText™, a titling and graphics software, has long benefitted from its native DataLink support.

Title pages prepared and displayed in LiveText can be updated in realtime based on data from a variety of different source types, including files in 'watch folders', data feeds from scoreboards, and more.

FIGURE 27

In turn, Viz LiveText (running on an outboard system) can transmit the updated title page display across the local network to a Vizrt live video mixer for use in live programming. (See your LiveText standalone documentation for more detail.)

Section 6.2 Video Mixers and DataLink

6.2.1 TITLE TEMPLATES

Vizrt live video mixers include a *native* DataLink implementation to dynamically update the values of *key name* entries in titles. When the page is displayed on output, information drawn from external data sources is substituted for the key name. (The external data is formatted with the attributes you assigned to the key name entry when creating the title page).

Hint: You can force a minimum number of whole digits before and places after the decimal as in the following example. If the current value for "some numeric key" is 12.23456789, we can force the title page to display four places before the decimal and just two after by using the following DataLink entry: %some numeric key[4.2]%

The displayed result will be "0012.23" (without quotation marks). Negative signs appear when called for, but the + sign can be optionally displayed as appropriate using something like %some numeric key[+4.2]%. Similarly, express a number as a percentage (of 1) using % numeric key[percent]%, or as an ordinal (1st, 2nd, etc.) using %some numeric key[ordinal]%.

You can enter DataLink keys as the source for text or image fields using the integrated version of Viz LiveText (provided with Vizrt video mixers for page authoring purposes), or later in Vizrt live video mixer's *live* text editor. Text fields on title pages can contain either literal text or a DataLink key. Let's spend on a few moments considering how you do the latter.

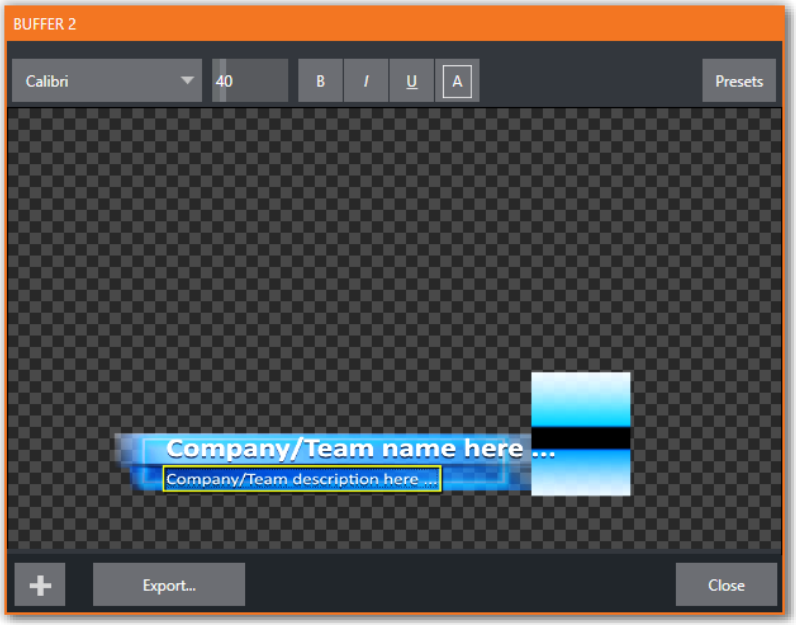


FIGURE 28

In Figure 28, note that the default entries for the name and description lines of the lower-third title page are bracketed by % (percentage) signs.

Hint: See the next section for a discussion of the special %Session xxxxx% keys.

Any text entry surrounded by % signs in this manner is automatically evaluated on display as a DataLink key, and the current value for that key is shown. You can manually enter keys by simply typing them into the field, but there's a method you might like better.

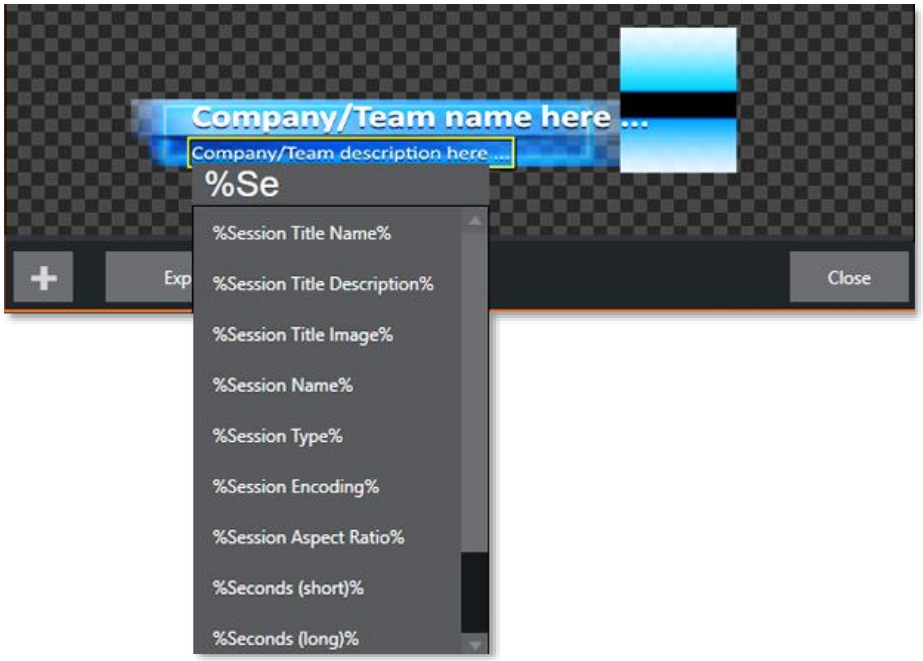


FIGURE 29

If you simply click a text field and type a % sign, DataLink keys are shown in a drop down menu. As you continue entering characters, the list updates to show relevant entries. You can use the arrow keys to highlight the key you want in the menu, and press Enter to select it. To enter a DataLink key for an image, right click the placeholder image in the live text editor, and select Properties.

Hint: Notice that many internal keys are provided, including keys based on time and day, Media Player metadata (such as %DDR1 Clip Alias% and %DDR1 Clip Comment%), and more. These allow you to easily, for example, use a single title page to automatically show the name and comment for the current DDR clip. If you then record a macro that displays that page in DSK1 (for example) briefly and then removes it, you can assign that macro to automatically display and hide the correct title for every clip you play from the DDR

FALLBACK KEYS

Datalink also incorporates support for fallback keys, specifically within a Buffer or Media Player. This functionality allows you to specify one or more 'fallback' keys, serving as default values if the primary key is not present.

For instance, consider a scenario where a title text line is set to display %mydate%:

- If the Datalink key mydate holds the value "January 1, 2024," the title will reflect that value.
- However, if the mydate key is absent, the title line will remain empty when displayed.

To address this, an alternative approach involves using the following syntax in the text input line:

%{mydate, Date}%.

In this case, the key Date, which always has a value, serves as a fallback. If mydate is not present, the current system date will be displayed instead.

Another option is to employ a literal string as a fallback key. Building on the previous example:

%{mydate, “this literal string”}%

- If the mydate key contains the value "January 1, 2024," the title will display that value.
- In the absence of the mydate key, the literal string 'this literal string' (without quotes) will be shown.

Multiple fallback keys can be added in a sequence, as demonstrated below:

%{first_key, fallback_key, another_fallback, "this literal string"}%

Or, referring to the earlier example:

%{mydate, Date, Year (full), “this literal string”}%

This flexibility allows you to set up a hierarchy of fallbacks, ensuring that the system seamlessly resorts to alternative values when the primary key is unavailable.

6.2.2 LIVEGRAPHICS

Viz LiveGraphics Creator for After Effects® (included with Viz Premium Access membership) makes creating animated titles and motion graphics for Vizrt live video production systems simple and fun.

FIGURE 30

You don't need to be an After Effects expert to create beautiful, multi-layered, editable LiveGraphic titles for your Vizrt live production systems make displaying and controlling these graphic pages a breeze.

Naturally, editable text can be modified in realtime using TriCaster's native Title Editor, which also lets you display or hide the individual graphic elements of your LiveGraphic title pages at will. In addition, the Title Editor lets you store different layouts as LiveGraphic presets. Recalling these presets triggers any number of dynamic layer animations to update the page.

DataLink keys can be inserted into LiveGraphics at the time of their creation, but of course the integrated Title Editor for Vizrt live production systems supporting LiveGraphics also lets you enter for keys for text lines and replaceable imagery later if you need to.

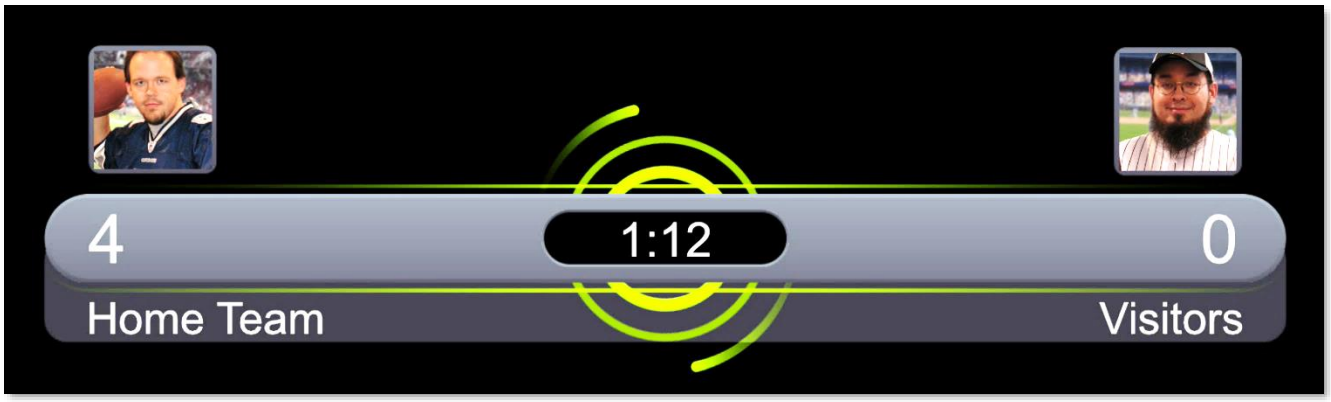


FIGURE 31

A line of text or image set to a *DataLink* key is automatically replaced by the value currently assigned to that key when the LiveGraphic layer containing it is displayed.

Section 6.3 DataLink Sources

Vizrt live video mixers include *internal* DataLink sources to complement the integrated title and CG toolset, extending the original data sources available in several ways. In some cases, support for a given source type has been enhanced; for example, the former ASCII text file support now includes XML and CSV file support. Beyond this, a number of important internal keys and external sources have been added.

Here's a list of some important data sources:

- **LivePanel**
  - Scoreboard applets
  - DataLink applet
- **File Watcher**
  - ASCII text files
  - XML files
  - CSV (Comma Separated Value) files
- **Database**
  - MySQL database queries
- **RSS** (Really Simple Syndication) feeds
- **External hardware controllers**
  - *Daktronics™
    - Allsport
      - Baseball
      - Basketball
      - Football
      - Hockey
      - Soccer
      - Volleyball
    - Allsport CG
      - Baseball
      - Basketball
      - Football
      - Hockey
      - Soccer
      - Volleyball
  - DSI (Basketball)
  - OES
    - Basketball
    - Hockey
  - Translux Fairplay
    - Basketball
    - Football

    - WhiteWay (Basketball)
    - Whiteway Rainbow (Basketball)
- **Internal**
    - Time
        - Time
        - Time (filename)
        - Hours (short)
        - Hours (long)
        - Hours (short, 24h)
        - Hours (long, 24h)
        - Minutes (short)
        - Minutes (long)
        - Seconds (short)
        - Seconds (long)
        - AM/PM (short)
        - AM/PM (long)
        - Next Event
        - Time Until Next Event (or end)
    - Date
        - Date (MM/DD/YY)
        - Date (DD/MM/YY)
        - Date (DD/MM/YYYY)
        - Date (filename)
        - SystemDate
        - Day (short, numeric)
        - Day (long, numeric)
        - Day (short)
        - Day (long)
        - Month (short, numeric)
        - Month (long, numeric)
        - Month (short)
        - Month (long)
        - Year (short)
        - Year (full)
    - PGM Source Name
    - PGM Source Comment
    - Session
        - Title Name
        - Title Description
        - Title Image
        - Session Name
        - Session Type
        - Session Encoding

        - Session Aspect Ratio
    - Media Player
        - DDR1 Clip Alias
        - DDR1 Clip Comment
        - Etc.
- **Web Browser**
    - DataLink Web plugin
        - text, including paragraphs
        - images (files or URLs)
- ** **Network**
    - TCP/IP
    - HTTP
    - NDI

* Certain Daktronics controllers (including Allsport 3000 and 5000 models) require an AllSport CG unit to convert the propriety Daktronics feed to serial data to DataLink. Please contact your Daktronics representative for more information.

** These methods are discussed in Chapter 7, Network A/V & Control.

6.3.1 DATALINK BROWSER EXTENSION AND MORE

As long as the above list might be, it is not complete. In addition, third-party applications can create DataLink keys and supply their values, and you can create and populate DataLink keys using the “datalink_set” shortcut.

One of the most interesting sources of *DataLink* keys is DataLink for the Vizrt live video mixer, Vizrt’s custom extension for the Chrome® web browser. Available without charge from the Chrome Web Store, DataLink Web allows you to easily populate both text and image DataLink keys from webpages.

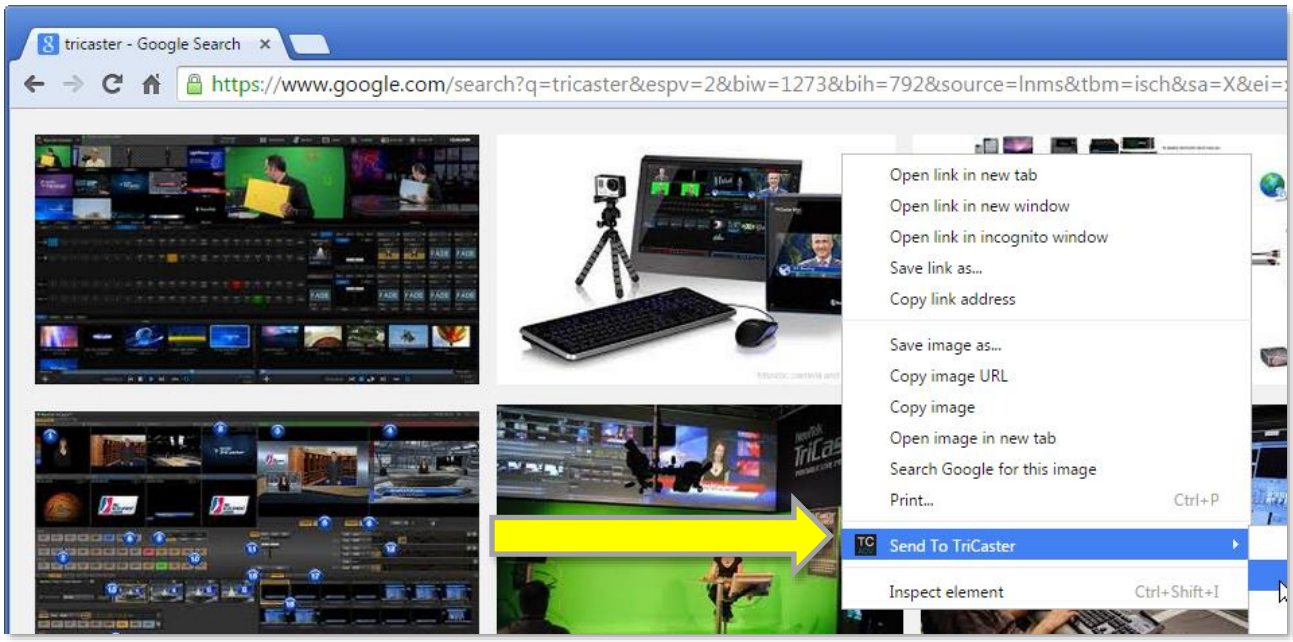


FIGURE 32

The *DataLink* keys and values are immediately available for use in Vizrt live video mixer title pages, and elsewhere in the live video mixer. Simply select some text, or an image, and use the right-click context menu (or a hotkey) to update a DataLink key you have defined. Any title page using that key will immediately update.

6.3.2 SESSION KEYS

Note that %Session Title Name% and %Session Title Description% are special DataLink keys. Along with %Session Title Image%, the values for these keys are defined in the Startup>Session screen.

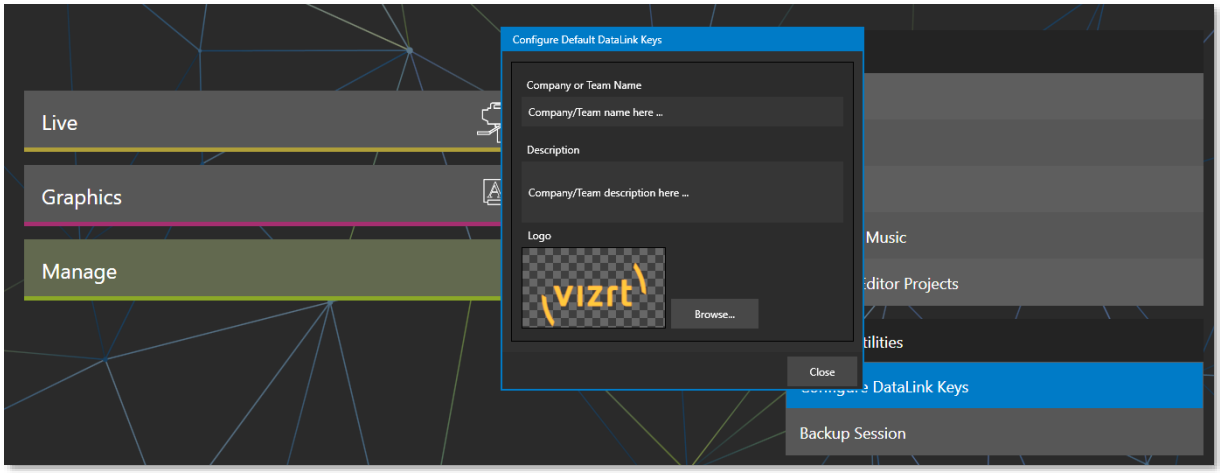


FIGURE 33

These special 'session keys' are pre-assigned by default to appropriate title entries in specific title pages supplied with your Vizrt live video mixer. When displayed live, these keys are replaced by the values you entered in the Startup screen. In certain cases, this means that simply taking a few moments to choose appropriate values for your company or client will automatically pre-populate stock titles in the session with those values. (Of course, you can modify the individual title page entries as you see fit, too).

6.3.3 TIME AND DATE

A diverse set of useful time and date keys are always available in the key insertion drop-down menu. These keys allow you to prepare clock and calendar objects that update in realtime on your title pages.

When these keys are displayed, the corresponding values are derived from the Vizrt live video mixer's production clock. This provides many useful and creative possibilities, including counting down to an upcoming events.

6.3.4 FILE WATCHER

Among other sources as described earlier, Vizrt live video mixer monitors files in designated *DataLink Watch* folders for changes to keys and their values.

The *DataLink Watch* folder system is implemented per-session, allowing you to automatically provide different video programs you produce with unique DataLink key setups. You will find the folder on your local Vizrt live video mixer host at (*Your session volume*):\Sessions*session-name*\DataLink Watch Folder.

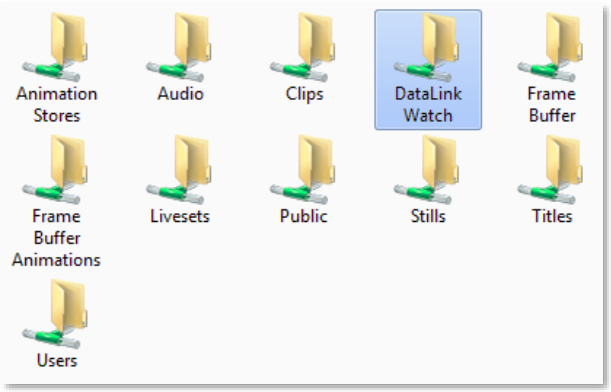


FIGURE 34

Hint: If you enable the Share Media Folders and Buffers option in Vizrt live video mixer's File menu (Live Desktop), this folder will be accessible to other systems on the network.

Note that it is best to write data into a different folder on the same volume (such as a sub-folder in the Watch folder), then move the file to the Watch folder. This reduces resource demands, and can prevent display glitches that may sometimes occur when modifying and saving files in the Watch folder itself.

ASCII TEXT

DataLink pulls data from ASCII text files (.txt) residing in the (constantly monitored) *DataLink Watch* folder. As this is the simplest source available to DataLink, let's use it to demonstrate a few basics before continuing.

1. Create a new text file in the folder (the filename doesn't matter), and open it in a text editor (Notepad will do).

To supply usable values for DataLink, the text files should contain only *key-value pairs*, arranged in the following format: [key] = [value]

Key names from the file(s) will be available as DataLink entries in your title pages. The value you enter beside the key name in the text file will be shown when the page is displayed on output.

Two Key-Value pairs entry examples are shown below:

city = San Antonio
temperature = 98˚

Note: Keys and values may contain punctuation and spaces.

XML

Similarly, XML (eXtensible Markup Language) files can supply DataLink keys and values. Consider the example xml file content provided below:

```
<Key>
  <word1>Hello</word1>
  <Child>
    <word2>World</word2>
  </Child>
</Key>
```

From the entries listed above, the DataLink drop-down will show the following keys with the values listed:

%Key word1% = Hello
%Key Child word2% = World

CSV FILES

Imagine using common spreadsheet functions to manage complex sport statistics, then pushing the results to a title page with a single keystroke. That's all possible, thanks to *DataLink's* CSV (Comma Separated Value) file support.

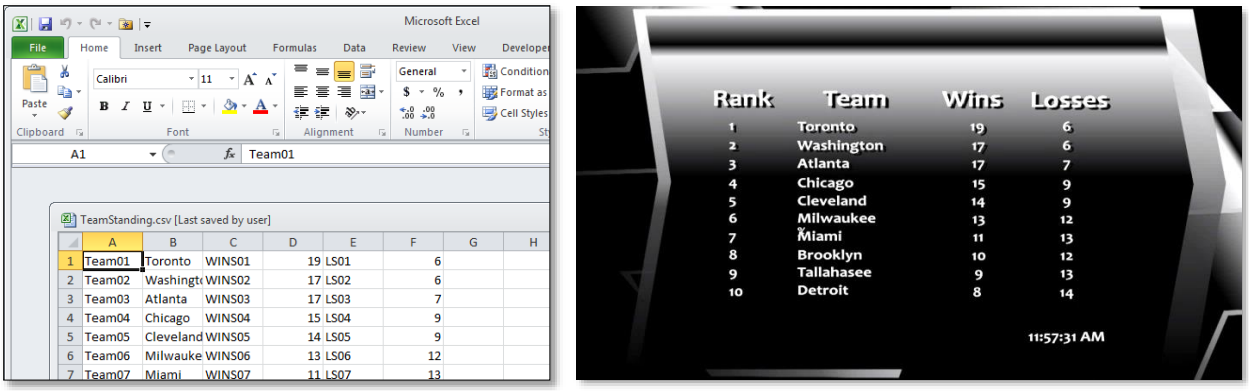


FIGURE 35

For example, simply save changes in the CSV file to Vizrt live video mixer's network-shared *DataLink Watch Folder*, and *DataLink* parses the keys and values it contains, then immediately updates the title page, even if it is on display at the moment.

DataLink parses key-value pairs from neighboring cells on each row as shown in the table below.

Team01	Germany	Team01Wins	6	Team02Losses	2
Team02	Belgium	Team02Wins	4	Team02Losses	1
etc.					

In this case, the key %Team01% would have the value "Germany"; %Team01Wins% would hold "6", etc.

Let's go on to consider the external hardware sources (such as scoreboards) supported by DataLink. Vizrt live video mixer depends on an external hardware connection to supply values for these keys. In the next section, we'll explain how to connect these external devices.

Section 6.4 Source Configuration

Internal DataLink sources, such as clip comments or time and date keys, do not require any configuration beyond populating them. Some other source types require a little more setup, however. Data from RSS feeds, database queries, and external hardware sources (such as scoreboards) fall into this category.

The necessary settings for these latter sources are conveniently located in the *DataLink Source Configuration* application, launched from the menu shown when you select *the Add-Ons* icon in the *Startup screen*. The *DataLink Source Configuration* panel has three tabs, *RSS*, *Database*, and *Scoreboard*. The purpose and contents of each is discussed next.

6.4.1 RSS

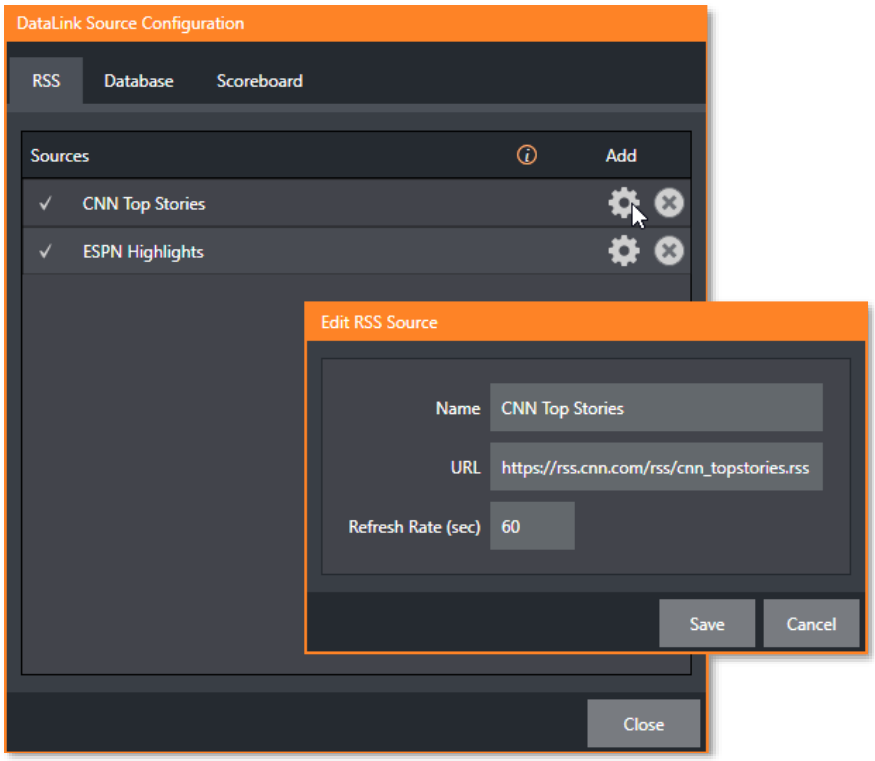


FIGURE 36

In the RSS tab, click the *Add* button at right to open a dialog that lets you define a new RSS source. Provide a *Name* to identify the new RSS source, and enter the URL to the feed below. The *Refresh Rate* entry below determines how often *DataLink* will poll the source for updates. Click *Save* to store the source (afterward, you can click the gear gadget that appears on rolling the mouse over the source entry to make changes, or the (x) to delete it).

Hint: Key names for RSS feed elements are automatically generated.

6.4.2 DATABASE

For database sources, *DataLink* monitors the value for keys you designate are produced by queries you define. The *Add a Database Key* dialog is shown when you click *Add* in the *Database* tab. Here you can enter a descriptive key *Name*, and the SQL query that will produce the desired value (or values).

Note: You (or someone helping you) will need a measure of familiarity with database addressing and queries.

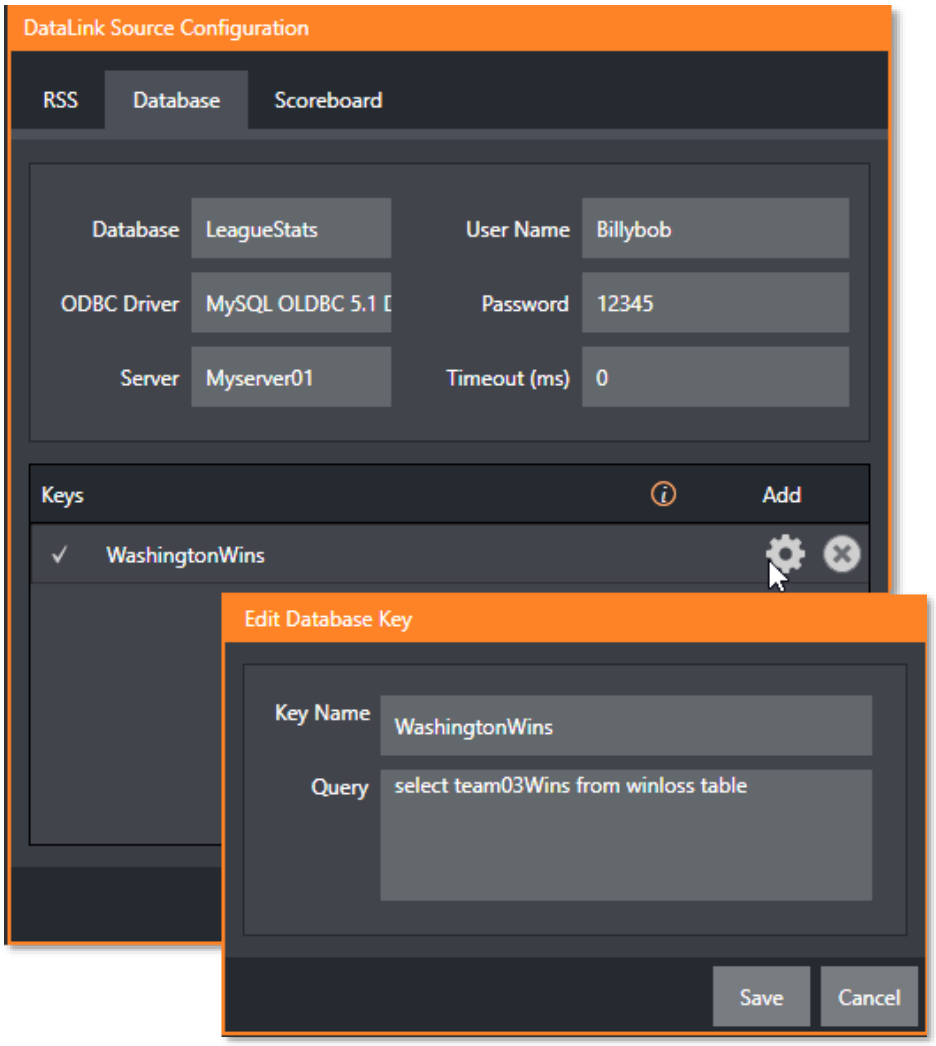


FIGURE 37

Enter a representative name in the *Database* box (this is simply to help you identify the data source; it need not be an actual file name). Then enter a *User Name* and *Password* for the database in the boxes provided, and specify the driver used for SQL queries in the *ODBC Driver* box. Finally, enter the *Server* name into the corresponding entry box.

Click *Add* to create a new DataLink key. Give it a suitable *Key Name* in the popup panel, and enter the query string that will produce the value(s) you wish to associate with this key into the large box below.

When the SQL query provides more than one match, DataLink creates a key/value pair for each qualified result.

> For example, a keyword “author” could produce an array of matches, which DataLink would arrange as follows:
>
> %author% -> "Voltaire"

%author.1% -> "James Joyce"

%author.2% -> "Herman Melville"

Click Save to finish the addition of the new key.

6.4.3 SERIAL (SCOREBOARD) SETUP

This DataLink component receives data from compatible external scoreboard hardware controllers. For information on connecting these devices to a Vizrt live video mixer, see Section 6.4.4.

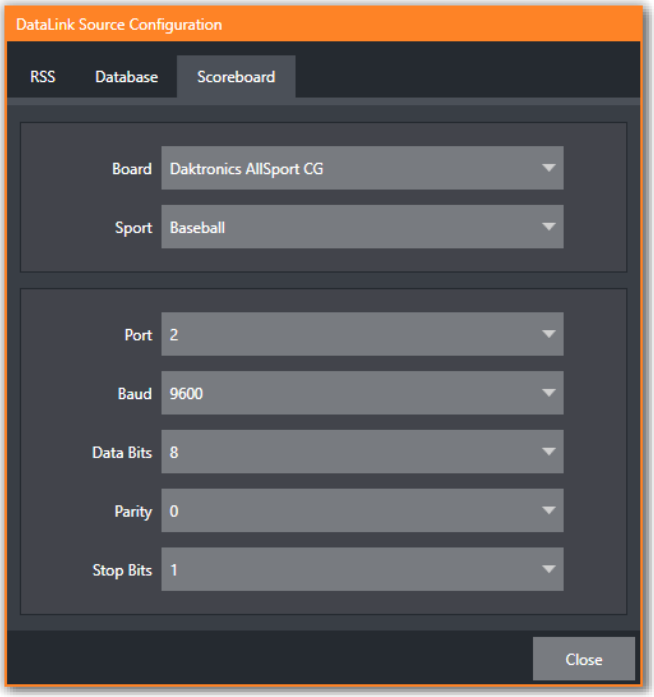


FIGURE 38

Once connected, use the *DataLink Source Configuration* utility to notify the Vizrt live video mixer that it is available as a source. Use the *Board* menu to choose the device brand/model you have connected from the list of supported devices. Choose a supported *Sport* in the same manner. The rest of the settings for serial devices auto-fill based on your Board and Sport selections, with one exception - select the Port using the information from the heading

Find the COM Port in Section 6.4.4.

Once you have a supported device successfully connected and configured, the drop-down key insertion menu in Viz LiveText's canvas will list valid key names for that device.

KEY NAME LIST

Appendix A, DataLink Hardware Keys lists the keys available for use with DataLink and the varied brands of external equipment it supports.

6.4.4 HARDWARE CONNECTIONS

Note: The steps in this section are mandatory if you require data from an external hardware scoreboard controller.

Naturally, for DataLink to communicate with an external data source, such as a scoreboard, that equipment must be connected to a Vizrt live video mixer and powered up. Also, DataLink must be configured to find and use the connection. We'll discuss how to make and configure connections under this heading.

USB-SERIAL ADAPTERS

The diversity of supported external systems, cable connectors, and available ports on the host system means this connection may require an adapter.

Newer external devices may use USB connections, but many use older RS-232 (25-pin) connectors (or occasionally, slightly more recent 9-pin) connectors.

Note: Unless the external system is supplied with a USB connection, a USB-Serial adapter is required to connect it to a Vizrt live video mixer.

To connect using a USB-Serial adapter, follow these steps:

- Connect the scoreboard controller's output cable connector to the USB-Serial adapter.
  - Connect the adapter to Vizrt live video mixer.
  - Install drivers for your USB-Serial adapter on the Vizrt live video mixer. Drivers are generally supplied on a Compact Disk (CD) packaged with the adapter by the manufacturer.

    Note that the Vizrt live video mixer may warn you about the dangers of foreign software if it does not recognize the driver for your adapter. (You may wish to ask Vizrt Customer Service about supported adapters or request that your favorite be qualified for exemption from these warnings.)

Note: Certain Daktronics controllers (including Allsport 3000 and 5000 models) require an AllSport CG unit to convert the propriety Daktronics feed to serial data for use in LiveText. Please contact your Daktronics representative for more information.

FIND THE COM PORT

The next step involves determining *which* COM port has been assigned to the new connection by the operating system. This information is required to configure DataLink.

- Right-click the *My Computer* icon on the *Windows Desktop*, and select *Manage* from the menu (to open the Computer Management panel).
- Open the *Device Manager*, and click the + sign next to *Ports (COM and LPT)* in the right-hand pane to disclose available communication ports.
- Locate the entry for your scoreboard controller – take note which COM port number is assigned to it (such as COM 1 or COM2).

Note: You should see your new connection listed. If it doesn't appear at first, try removing and re-inserting the USB cable connector – or you can use the "Scan for hardware changes" item in the Device Manager's Action menu. (If it appears, but shows a **!** icon next to its entry, this may indicate a problem with either the USB connection or your driver installation – try re-installing the driver, following the directions supplied with it.)

- Close the *Device Manager*.

Again, the port number you noted above is required to enable DataLink to recognize the external device.

Important Note: In some environments, Windows may arbitrarily reassign the external device to a different COM port following a reboot. If this happens, you could simply update the COM port entry in the affected configuration profile. However, you may prefer instead to lock the connected device to a specific COM port, using the Windows Device Manager.

To do this, please locate the current port entry for your scoreboard controller. Right-click the entry name, and select Properties in the drop-down menu. Next, click the Port Settings tab at the top of the Properties panel, and click the button labeled "Advanced." Use the Com Port Number drop-down menu to choose an unused port number, and click the OK button. OK the Properties panel too, then close the Device Manager. The Port Number you assigned should now be retained on subsequent reboots.

Chapter 7 NETWORK A/V & CONTROL

Most Vizrt live production systems support both ingest and output of a/v feeds over standard network infrastructure via NDI. This provides a plethora of valuable creative and efficient alternatives. In addition, many Vizrt systems can send or receive control instructions from networked devices and systems, offering many powerful possibilities.

Given Vizrt's preeminent position of innovation in IP solutions for video production, it will surprise no one that it is entirely possible to operate and/or control various Vizrt products from other Vizrt or third-party systems or software using network control methods. We will discuss various approaches available in this chapter.

Section 7.1 NDI

NDI (Network Device Interface) is an immensely popular standard for live production video over IP workflows, even when limited to Gigabit Ethernet networks. NDI allows systems and devices to identify and communicate with each other, to encode, transmit, and receive high quality, low latency, frame-accurate video and audio over IP in real time. NDI enabled-devices and software can enhance video production pipelines, making video input and output available anywhere your network runs.

Most Vizrt live video products and an immense number of third-party systems provide direct support for NDI, both for ingest and output.

7.1.1 CONTROL CONNECTIONS

NDI supports more than just a/v data transfer. As just one example, it makes tally (on-air) notification available to all connected NDI sources and systems. Even better, NDI provides the ability to transmit instructions between Vizrt live production systems and other connected devices.

For TriCaster, applications can be developed to send commands to systems connected by NDI. Here is a code example to perform a take:

```
NDIlib_metadata_frame_t meta_data;
meta_data.p_data = "<ntk_shortcut><shortcut name=\"main_dsk1_take\"
      value=\"\"/></ntk_shortcut>";
NDIlib_send_send_metadata(pNDI_send, &meta_data);
```

Hint: Multiple shortcuts can be sent this way.

In similar fashion. Datalink updates can be sent along NDI connections (requires V2 software). Here is example code to send a DataLink message:

```
NDIlib_metadata_frame_t meta_data;
    meta_data.p_data = "<ntk_shortcut><shortcut name=\"set_datalink\"
  datalink_key=\"second_line\" datalink_value=\"NewTek Inc\"/></ntk_shortcut>";
    NDIlib_send_send_metadata(pNDI_send, &meta_data);
```

Third-party developers can also implement custom commands to support their requirements. These can even be used in macros just like 'Vizrt native' commands. Documentation supplied with third-party products will provide information on custom commands that have been prepared for your use by their developers.

MACROS AND NDI

As well, Vizrt live production systems with macro support can use this mechanism to transmit and receive supported instructions across an NDI connection (see Chapter 4, The Macro System).

This makes it a trivial matter to send control messages between a TriCaster and Viz 3Play, or from one TriCaster to another, for example. Such a macro might perform synchronized operations on both systems and, conveniently, can be executed by a keystroke shortcut or other macro trigger (see Chapter 5, Triggering Macros).

When both parties to the network 'conversation' support NDI, there is no need for complicated configuration. The Vizrt live video system 'knows' which NDI source is connected to its network inputs, and automatically opens a bi-directional communication channel to its host. Let's consider an example scenario.

Example

You identify the upstream NDI source connected to a specific system's inputs as targets for instructions (rather than the local host) using a special "net*n*" shortcut entry in a macro.

Hint: Commands to both the remote target and local host can be entered on different lines in a single macro.

When examined in Vizrt live video system's Macro Editor, a macro entry of this type might look as follows:

Delay (ms)	Shortcut	Value	Key 1	Value 1
0	net1		shortcut_name	main_auto

In this case, the shortcut "net1" defines the Switcher input (i.e., input 1) whose NDI source system is the target for an instruction at right on the same row in the macro.

The “shortcut_name” entry in the Key 1 field tells the target system to expect a valid shortcut to be supplied in the Value 1 field that follows. Additional key/value entries can be supplied as arguments of the instruction performed when the macro is executed.

Hint: One way to learn what shortcuts and values are relevant is to record operation on the target system to a local macro, to see what was captured. Similarly, to debug a macro that sends shortcuts to an external NDI-connected device, record a macro on the target system during operation to see what is received.

Viz 3Play Channel Shortcuts

Viz 3Play also supports a small handful of unique ‘channel’ commands, listed below. Unlike standard shortcuts, these target the relevant module supplying either output A or B, whichever is connected to the Switcher input identified by the “net” shortcut. (The target for most other shortcuts is defined by individual prefixes, as discussed in Section 4.5.1.)

Note: The syntax for ‘channel’ commands differs slightly from standard usage detailed previously, as shown below.

CLIP_STORE

Delay (ms)	Shortcut	Value	Key 1	Value 1
(ms)	*net1*	clip_store	index	*ID*

This command stores a local reference ID for the current clip (the one visible on the network input). The value *ID* can be a string. ID is global and shared across your system (i.e., not per system output).

For example:

Delay (ms)	Shortcut	Value	Key 1	Value 1
.0001	net1	clip_store	index	AAA1

The entry above will ‘remember’ the current clip with the name “AAA1”. (The default for ID is an empty string, which is a valid storage target.)

CLIP_RESTORE

Delay (ms)	Shortcut	Value	Key 1	Value 1
(ms)	*net1 or net2*	clip_restore	index	*ID*

This command cues up content previously stored with a specified ID value on the upstream source channel assigned to the network input designated.

For example:

Delay (ms)	Shortcut	Value	Key 1	Value 1
.0001	net1	clip_restore	index	AAA1

The clip previously indexed as "AAA1" (using clip_store) is restored on the source system output channel connected to Net 1. The playhead is set to the beginning of the clip. (If the indexed clip is not located, nothing occurs.)

CLIP_ SELECT

Delay (ms)	Shortcut	Value	Key 1	Value 1
(ms)	*net1 or net2*	clip_select	index	#

Select a page (or clip) defined by the value assigned to index. This may be a number specifying a particular page or at times, another property.

For example, sending a "clip_select" command to Viz 3Play with a suitable numeric value assigned as the "index" key selects a specific Play List tab by index (assuming Play List mode is active. On the other hand, in Clip List mode, if the value for "index" was "0-023" the clip referred to would be selected.

For example:

Delay (ms)	Shortcut	Value	Key 1	Value 1
.0001	net1	clip_select	index	4

This would select the fourth Play List tab on the Viz 3Play output (A or B) connected to Net 1.

CLIP_ MOVE

Delay (ms)	Shortcut	Value	Key 1	Value 1
(ms)	*net1 or net2*	clip_ move	distance	#

Move the specified number of pages forwards or backwards from the current page.

For example:

Delay (ms)	Shortcut	Value	Key 1	Value 1
.0001	net1	clip_move	distance	-1

The entry above would select the previous clip on the source connected to Net 1.

CLIP_ PLAY

Delay (ms)	Shortcut	Value	Key 1	Value 1	Key 2	Value 2
(ms)	*net1 or net2*	clip_ play	speed	#	position	#

You can specify "speed," "position" or both keys (the order of keys is not important).

When "position" is not specified, play begins at the current frame. The value for position is specified in seconds, while speed is expressed as a playback rate value (1.0 = 100%).

For example:

Delay	Shortcut	Value	Key 1	Value 1	Key 2	Value 2
.0001	*net1*	clip_ play	speed	-.5	position	10

This entry would play a clip backwards at 50% speed from a position 10 seconds into the clip.

CLIP_ SCRUB

Delay (ms)	Shortcut	Value	Key 1	Value 1
(ms)	*net1 or net2*	clip_ scrub	distance	#

This command will move the playhead backward or forward by a distance of # seconds.

For example:

Delay (ms)	Shortcut	Value	Key 1	Value 1
.0001	net1	clip_scrub	distance	5

The entry above would advance the playhead five seconds further into a clip displayed on the source connected to Net 1.

NOTES: The software associates values with their key name, so keys can be entered in any order. The following are valid formats for numeric entries: "+0.1", "0.1", ".1", "-0.1".

Section 7.2 TCP/IP

It is also possible to communicate with many Vizrt live video systems over a standard network TCP/IP connection. Custom applications running on a networked host system can directly control the functionality of most Vizrt live production systems by this means.

Here is some sample code that will open and send a message over TCP:

```
int _tmain( int argc, _TCHAR* argv[] )
{     // Initialize Winsock (only needed once for the entire program)
      WSAData wsaData_;
      ::WSAStartup( MAKEWORD(2, 2), &wsaData_ );

      // The machine name we are going to connect too.
      const wchar_t* p_machine_name = (argc>1) ? argv[1] : L"127.0.0.1";

      // We create a socket to use
      SOCKET hSock = ::socket( AF_INET, SOCK_STREAM, 0 );
      if ( hSock == INVALID_SOCKET ) { /* Handle error */ return 1; }

      // We are now going to get the address information
      struct addrinfoW *p_result = NULL, hints = { 0 };
      hints.ai_family   = AF_INET;
      hints.ai_socktype = SOCK_STREAM;
      hints.ai_protocol = IPPROTO_TCP;

      // Resolve the server address and port
      if ( ::GetAddrInfoW( p_machine_name, /* Port No.*/ L"5951", &hints, &p_result ) != 0 )
      { /* Handle error */ return 1; }

      // We now try to connect to the actual destination
      if ( ::connect( hSock, p_result->ai_addr, p_result->ai_addrlen ) == SOCKET_ERROR ) {
      /* Handle error */ return 1; }

      // We are going to register to receive state information
      const char cmd_2[] = "<shortcut name=\"main_dsk1_take\" value=\"\"/>";
      const int cmd_2_sz = ::strlen( cmd_2 )+1;
      if ( ::send( hSock, cmd_2, cmd_2_sz, 0 ) != cmd_2_sz ) { /* Handle error */ return 1; }

      // Close the socket
      ::closesocket( hSock );

      // Finished !
      return 0;
}
```

7.2.1 FINDING COMMANDS

An easy way to find a command (shortcut) is to record a macro on the Vizrt system you are working with. Then you can use the integrated *Macro Editor* to view the recorded shortcuts, and see their arguments. For example, Figure 39 shows the result of recording a main switcher Background Auto. The "shortcut" column lists the command applied, and the subsequent columns list any values or additional data associated with its execution.

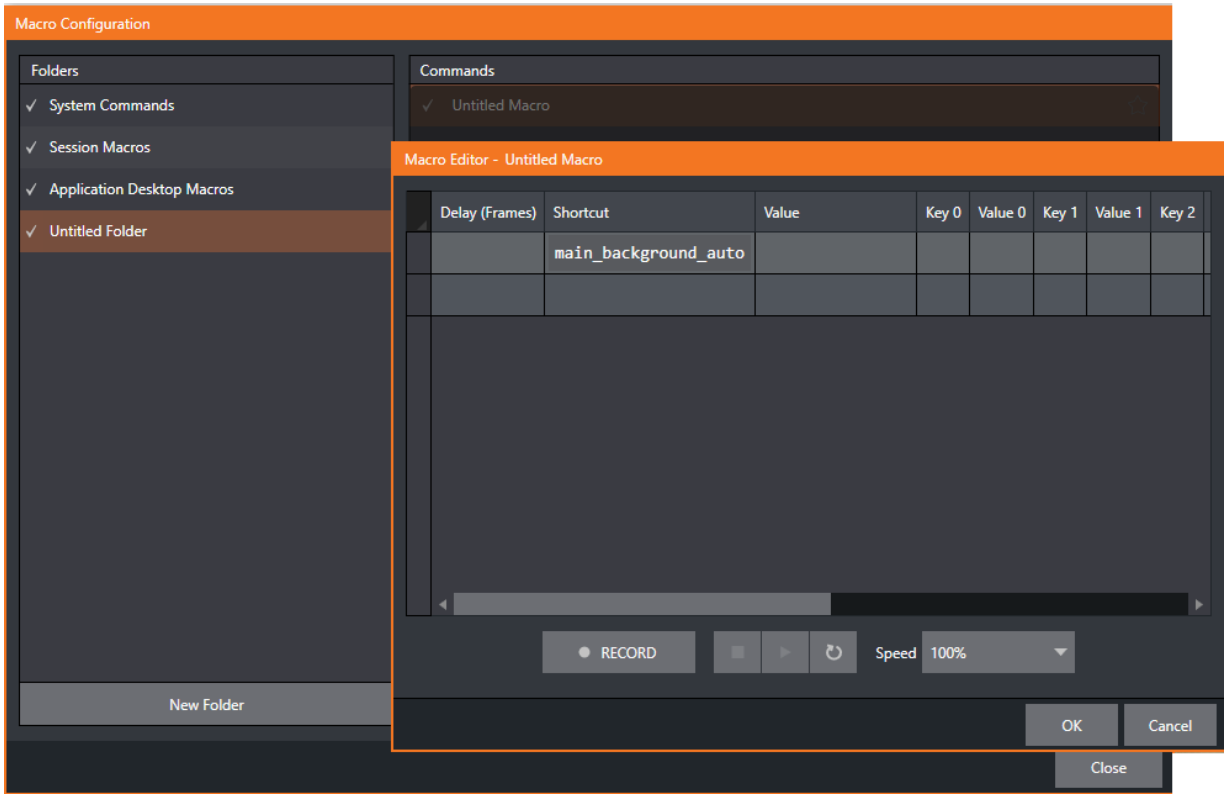


FIGURE 39

The next section explains how to use this data to produce the XML string needed to issue commands across the network, as illustrated in the code example above.

Hint: Section 4.5.1, Understanding Shortcuts, can help you to locate and use the shortcuts you need.

EXPORTING MACROS

Another way to obtain particularly useful information about macros and the shortcuts used in them is to use the Export feature provided in the Macro Editor provided on supporting Vizrt products.

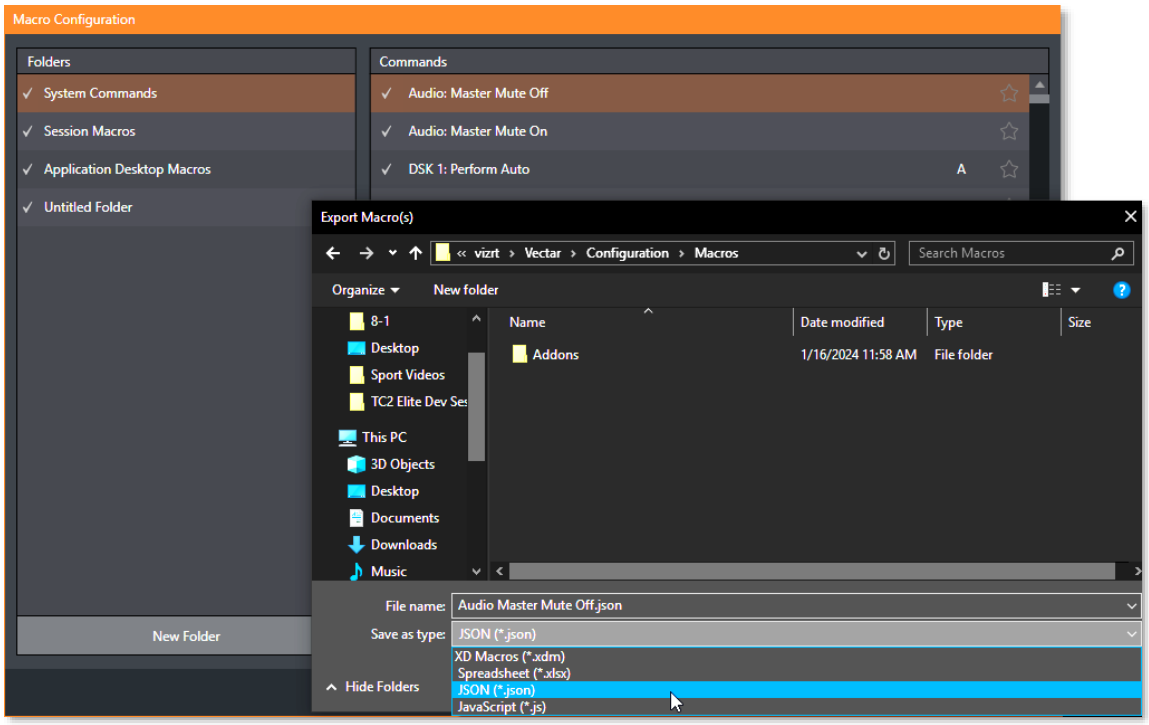


FIGURE 40

Right-click a macro (or even a folder full of macros) to show a menu including the Export item. Use the Save as Type menu at the bottom of the file-save window that opens to choose which format the export operation will produce from the following options:

- XD Macros – Vizrt's internal macro format
- Spreadsheet – Microsoft Excel's common XLSX format
- JSON – JavaScript Object Notation format, a popular data-interchange language supported by modern programming languages.
- Javascript – This is a particularly interesting and useful alternative, as the exported code includes a functional websocket implementation.

Javascript Export

A WebSocket connection can execute sequential commands faster a series of HHTP GET requests. The example code exported illustrates managing multiple commands, keys and values.

The Javascript code loops through consecutive shortcuts, including delays found in the original macro, and also demonstrates how to establish and maintain a WebSocket connection.

MACRO UTILITY SHORTCUTS

We'll highlight one particular group of shortcuts here. These commands operate on macros, and thus can serve to reduce or eliminate the need to send a batch of shortcuts sequentially; this is equally true whether you are using TCP/IP, HTTP or NDI for communication.

These are:

- play_macro_byname – the value supplies the name of a macro that will be run.
- play_macro_byid – internally, macros are identified by an ID string (see hint below). Launching a macro by its ID can be useful when it is possible that the operator may rename a macro.
- record_macro_byname
- stop_record_macro

Other macro-related shortcuts worth mentioning follow below:

- stop_macro_byname
- stop_macro_byid
- macro_is_enabled_byname
- macro_is_enabled_byid
- get_macroid_byname – prints the first matching macro ID to the Notifications panel.

Hint: macros are defined in two files. The first is named system_macros.xdm, and is located in the directory at C:\ProgramData\Vizrt\product-name\Configuration\Macros\. The file lists (write-portected) system macros as well as any global (non-'session') macros you create. The second is located in the session folder, typically D:\Sessions\session_name\ and is named session.xdm. This file holds macros affecting just that one session.

Alternatively, names and IDs for all macros can be read in a web browser by entering the URL below:

http://[tricaster-ip]/v1/dictionary?key=macros_list

7.2.2 COMMAND FORMAT

Commands (shortcuts) you will send across the network are formatted as XML strings as follows:

```
<shortcut name="" value=""/>
```

Commands with multiple parameters are formatted as follows:

```
<shortcut name="" value="" key1="value1"/>
```

For instance, the following table illustrates several commands in XML format:

Command formatted as XML	Description
<shortcut name="main_background_take"/>	Perform a Take
<shortcut name=" main_a_row_named_input" value=”ddr1” />	Select DDR 1 on the program row

All XML strings are sent in UTF8 encoding, and should be terminated either with a carriage return, line feed, or NULL terminator. Many commands can be sent back to back without requiring the socket to be closed.

To send multiple commands at once, you may use the following method:

<shortcuts><shortcut name="" value=””/></shortcuts>

7.2.3 TALLY EXAMPLE

Here is an example in which we connect to a Vizrt system via TCP and get tally (On Air) notifications.

The sample code supplied below will connect to a TriCaster TC1 (or any Vizrt switcher) and start listening for the current states of the system.

```
// Remote Control.cpp : Defines the entry point for the console application.
//

#include "stdafx.h"

int _tmain( int argc, _TCHAR* argv[] )
{    // Initialize Winsock (only needed once for the entire program)
     WSAData wsaData_;
     ::WSAStartup( MAKEWORD(2, 2), &wsaData_ );

     // The machine name we are going to connect too.
     const wchar_t* p_machine_name = (argc>1) ? argv[1] : L"127.0.0.1";

     // We create a socket to use
     SOCKET hSock = ::socket( AF_INET, SOCK_STREAM, 0 );
     if ( hSock == INVALID_SOCKET ) { /* Handle error */ return 1; }

     // We are now going to get the address information
     struct addrinfoW *p_result = NULL, hints = { 0 };
     hints.ai_family   = AF_INET;
     hints.ai_socktype = SOCK_STREAM;
     hints.ai_protocol = IPPROTO_TCP;
```

```
    // Resolve the server address and port
    if ( ::GetAddrInfoW( p_machine_name, /* Port No.*/ L"5951", &hints, &p_result ) != 0 ) {
/* Handle error */ return 1; }
    // We now try to connect to the actual destination
    if ( ::connect( hSock, p_result->ai_addr, p_result->ai_addrlen ) == SOCKET_ERROR ) { /*
Handle error */ return 1; }

    // We are going to register to receive state information
    const char cmd_2[] = "<register name=\"NTK_states\"/>";
    const int cmd_2_sz = ::strlen( cmd_2 )+1;
    if ( ::send( hSock, cmd_2, cmd_2_sz, 0 ) != cmd_2_sz ) { /* Handle error */ return 1; }

    // For the next minute or so, we display whatever is happening on in terms of states
    for( DWORD start_time = ::GetTickCount(); ::GetTickCount() - start_time < 60000; )
    {   // Lets receive some data. We want to do something better than this in production
        // code, but this is meant to keep things short.
        char some_data[ 4096 ];
        if ( ::recv( hSock, some_data, sizeof(some_data), 0 ) < 0 ) { /* Handle error */ return
1; }

        // Display the data
        ::puts( some_data );
    }

    // Lets stop receiving state changes from now
    const char cmd_3[] = "<unregister name=\"NTK_states\"/>";
    const int cmd_3_sz = ::strlen( cmd_3 )+1;
    if ( ::send( hSock, cmd_3, cmd_3_sz, 0 ) != cmd_3_sz ) { /* Handle error */ return 1; }

    // Close the socket
    ::closesocket( hSock );

    // Finished !
    return 0;
}
```

You will receive messages like this one when the tally changes :

```
<shortcut_states>
  <shortcut_state name="program_tally" value="INPUT1|BFR2|DDR3" type="" sender="" />
  <shortcut_state name="preview_tally" value="INPUT7" type="" sender="" />
</shortcut_states>
```

In this example, INPUT1, BFR1, DDR3 are identified as being on Program output, while INPUT7 is on Preview.

Section 7.3 HTTP

Most current Vizrt products also function as HTTP servers, and are able to receive either GET or POST messages. HTTP methods that are easily accessed by advanced end users.

Using a simple text editor, anyone conversant with HTML can create useful web browser 'applets' that can interoperate with Vizrt systems.

Add programming knowledge (say, javascript or Python for example) and an entire world of possibilities open up. The internal web server can be addressed as in this example:

http://*NewTek live video mixer SystemName*

Hint: In order to determine if the session is running, you can query "http:// NewTek live video mixer SystemName /v1/live". Alternatively, us the IP address for the system, for example "http:// 192.168.1.24/v1/live".

This will return a text string of TRUE when in a session, and FALSE when in the control panel.

7.3.1 PASSWORD PROTECTION

Password protection for web access conforms to standard practices for a web server, using a standard Apache server password file. Users or developers can use standard command line tools to configure the password as follows:

https://httpd.apache.org/docs/current/programs/htpasswd.html

The password file is typically located in the folder below:

%PROGRAMDATA%\NewTek*product_name*\Configuration\web_passwords

Deleting this file removes web password protection for the system. The TriCaster *Administrator panel* also provides a convenient "Change password" link. Upgrade installers may show option to "Disable web access password". By default this option is disabled (i.e., the password is enabled by default). The default username and passwords are both "admin."

The web server employs standard HTTP authentication methods. While RFC3986 is supported, we would recommend digest access authentication which follows RFC2069.

For more information, please refer to shttps://en.wikipedia.org/wiki/Basic_access_authentication and https://en.wikipedia.org/wiki/Digest_access_authentication.

7.3.2 GET COMMANDS

You can send shortcut commands and (macro) trigger messages by using http GET commands.

For example, to send a shortcut that executes an "Auto" (transition) on the main Switcher's background video layer, you would simply get the address below:

http://*Insert_IP_Address*/v1/shortcut?name=main_background_auto

Hint: The URL you wish to 'get' should be formatted as follows:

/v1/shortcut?name=NAME&value=VALUE&value1=ANOTHER_VALUE

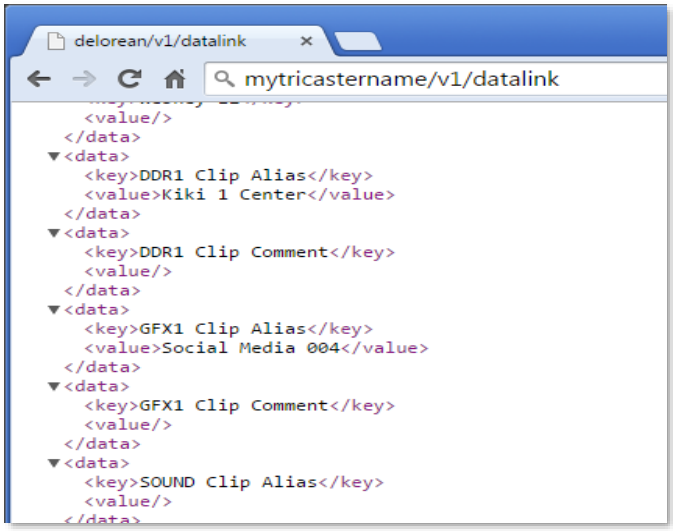
To issue a trigger you would GET the address "/v1/trigger?name=NAME"

To issue a DataLink update, you would GET the address "/v1/datalink?key=KEY&value=VALUE". You can easily obtain a full list of the current set of DataLink key/value pairs by querying the URL below:

http:// NewTek live video mixer SystemName /v1/datalink

This returns the current DataLink keys and values in the format listed next:

```
<datalink_values>
<data>
<key>Time</key>
<value>5:08:50 PM</value>
</data>
<data>
<key>Hours (short)</key>
<value>5</value>
</data>
<data>
<key>Hours (long)</key>
<value>05</value>
</data>
<data>
<key>Hours (short, 24hr)</key>
  (etc…)
```



Note: The name and value pairs must be correctly escaped.

7.3.3 POST COMMANDS

You can POST a <shortcut …> or <trigger …> message directly. You should send these to any URL that is in the path /v1/shortcut, /v1/trigger or /v1/datalink. Since these are XML messages, the content-type should be text/xml. For instance, if you wished to send a shortcut contained in the file "MyShortcut.xml," you would use the following CURL command line:

curl -X POST -d @MyShortcut.xml http:// NewTek live video mixer SystemName/v1/shortcut --header "Content-Type:text/xml"

If you are using PHP, it is common to wish to post commands in a form “shortcut=<shortcut name=value…”. This is automatically detected for “shortcut=”, “trigger=”, “datalink=” where the value is the regular XML form of the shortcut (the value is fully escaped).

7.3.4 MACROS AND HTTP

It’s also possible to use the native Macro engine on Vizrt systems to send (HTTP) GET commands to external Vizrt systems or other supporting devices (such as a PTZ camera). To do this, use the shortcut “http_request” in your macro.

Doing so sends the GET request to the target identified in the value string you supply. For example, a macro prepared as shown below will trigger an Auto (transition) on the main Switcher of a TriCaster with the IP address 192.168.1.120. (Any response the request generates is ignored.)

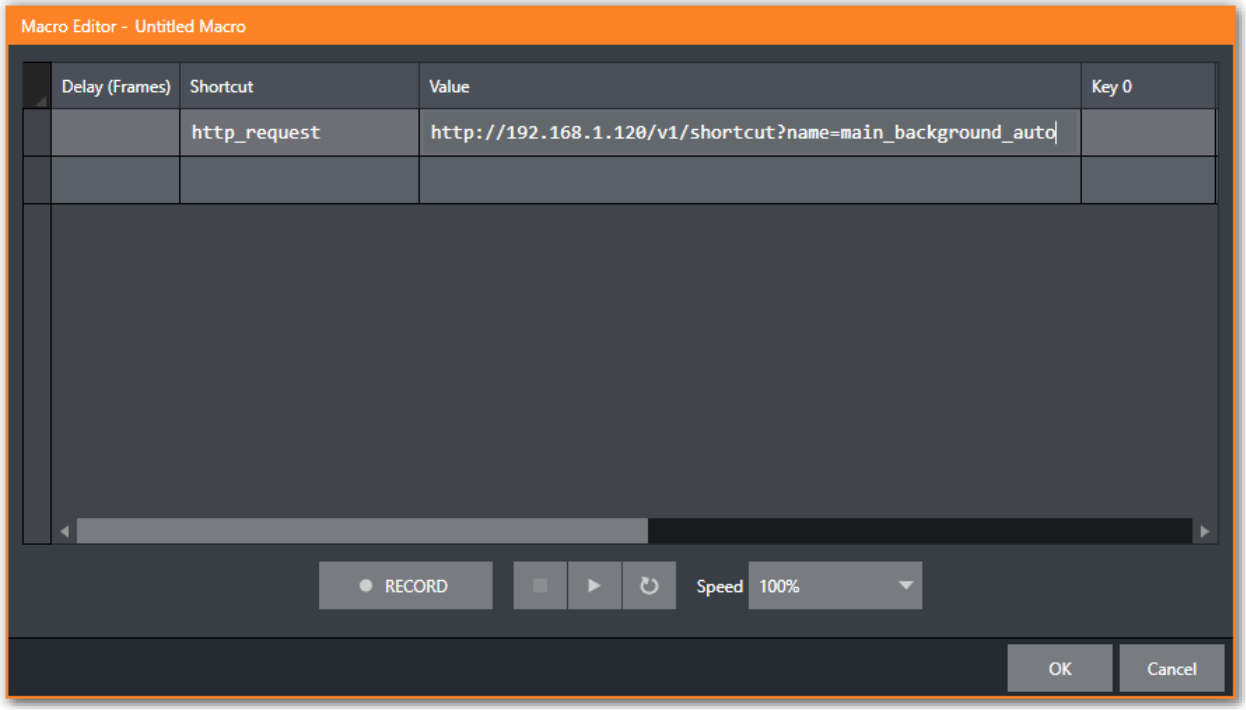


FIGURE 41

7.3.5 FILE TRANSFER

It is possible to send media files directly to a Vizrt live video mixer via HTTP post commands. Post commands are sent to http:// NewTek live video mixer SystemName/v1/file. As an example, the following curl command would post a PNG file to the Vizrt live video mixer system.

```
curl.exe -X POST –data-binary @YourCoolFile.png http:// NewTek live video mixer
    SystemName/v1/file --header "Content-Type: image/png " --header
    "Filename:HelloWorld.png" --header "Overwrite:true"
```

The header “Filename:somefilename” is optional, but allows you to give the Vizrt live video mixer a hint as to what file-name to choose for this file. If one is not specified then a filename will be chosen for you automatically.

Specify “Overwrite:true” if you wish the filename to be used no matter whether it exists or not (by default, overwrite is false).

The response from the file transfer will either be a 400 error, with a text description of the error, or a 200 success with the filename of the file returned; allowing you to then use this in titles, DDRs, etc.

Supported content types are:

Type	Description
audio/mpeg	MP3 Audio file
audio/basic	AU Audio file
audio/x-wav	WAV file
audio/x-aiff	AIFF file
image/gif	GIF image file
image/jpeg	JPEG image file
image/png	PNG image file
image/tiff	TIFF image file
image/bmp	BMP image file
video/x-msvideo	AVI file
video/quicktime	QuickTime file (MOV)
video/mp4	MPEG-4 wrapper (H.264 normally)
video/mpeg	MPEG-2 video file.

7.3.6 VIDEO PREVIEWS

It is possible to receive JPG images back for any source within the Vizrt live video mixer or Viz 3Play system. These may be queried as described next.

Possible names of sources for use with this method are listed in the table below:

Source Name	Description	Products
output1	The primary video output	All Vizrt live video mixer units
output2	The secondary video output	All Vizrt live video mixer units
output3	The third video output	Viz TC8000 only
output4	The fourth video output	Viz TC8000 only
input1- input(n)	Video inputs	All Vizrt live video mixers
net1, net(n)	NDI inputs	All Vizrt live video mixer units
ddr1, ddr2	DDR media players	All Vizrt live video mixer units
gfx, gfx2	Graphics players	All Vizrt live video mixer units
bfr1 - bfr15	15 buffers	All Vizrt live video mixer units

Additional parameters are available that perform the following:

Parameter	Description
xres	Force the resolution of the image to be this width. If this is omitted, this is computed from the source aspect ratio.
yres	Force the resolution of the image to be this width. If this is omitted, this is computed from the source aspect ratio.
q	JPEG image quality (range 50-100).

TABLE 1

In general, it is wise not to query frames at high image rates. Request them at a resolution that is close to what you need. For instance, to receive a 640x480 preview of the current output, you would get:

http:// NewTek live video mixer
SystemName/v1/image?name=output&xres=640&yres=480

Note: See also Section 7.3.9 for information on Vizrt live video mixer's WebSocket implementation.

7.3.7 GENERATING ICONS

It is possible to obtain an icon representation for any file that is on the system. This can be used, for instance, to generate icons for Viz LiveSet or Transition files that may be loaded into the switcher (with the names given to you by getting the Vizrt live video mixer state data). Similar to how inputs are read, you get an icon by specifying the path (required), the resolution of the longest side (optional), and the JPEG quality (optional).

For instance to get the icon for the LiveSet effect file at the location below:

C:\"NewTek live video mixer "\ProgramData\Newtek\Effects\LiveSets\
Alignment\Center.LiveSet

You could get the URL:

http://“NewTekVideoMixer“/v1/icon?filename=C:%5CProgramData%5CNewTek%5CEffects%5CLiveSets%5CAlignment%5CCenter.LiveSet&res=320&Q=90

7.3.8 GETTING TALLY AND OTHER SETTINGS

Vizrt live video mixers can provide XML-formatted lists of parameters and values representing various states of the switcher, buffers and effects. Combining this information with scripted shortcut commands, sophisticated interactions can be made between the Vizrt live video mixer and a web application, controlled from either side of the client/server relationship.

Requesting the URL below (where “NewTek live video mixer System” is the system name), returns a full description of all of the video inputs on the system, with information regarding whether they are currently being displayed on program or preview output (tally information):

http:// NewTek live video mixer SystemName/v1/dictionary?key=tally

An example returned XML result would be as follows:

```
<tally>
<column name="input1" index="0" on_pgm="false" on_prev="false" ndi_id="0"/>
<column name="input2" index="1" on_pgm="false" on_prev="false" ndi_id="1"/>
<column name="input3" index="2" on_pgm="false" on_prev="false" ndi_id="2"/>
<column name="input4" index="3" on_pgm="false" on_prev="false" ndi_id="3"/>
<column name="input5" index="4" on_pgm="false" on_prev="false" ndi_id="4"/>
<column name="input6" index="5" on_pgm="false" on_prev="false" ndi_id="5"/>
<column name="input7" index="6" on_pgm="false" on_prev="false" ndi_id="6"/>
<column name="input8" index="7" on_pgm="false" on_prev="false" ndi_id="7"/>
<column name="input9" index="8" on_pgm="false" on_prev="false" ndi_id="8"/>
<column name="input10" index="9" on_pgm="false" on_prev="false" ndi_id="9"/>
<column name="input11" index="10" on_pgm="false" on_prev="false" ndi_id="10"/>
<column name="input12" index="11" on_pgm="false" on_prev="false" ndi_id="11"/>
<column name="input13" index="12" on_pgm="false" on_prev="false" ndi_id="12"/>
<column name="input14" index="13" on_pgm="false" on_prev="false" ndi_id="13"/>
<column name="input15" index="14" on_pgm="false" on_prev="false" ndi_id="14"/>
<column name="input16" index="15" on_pgm="false" on_prev="false" ndi_id="15"/>
<column name="bfr1" index="16" on_pgm="false" on_prev="false"/>
<column name="bfr2" index="17" on_pgm="false" on_prev="false"/>
<column name="bfr3" index="18" on_pgm="false" on_prev="false"/>
<column name="bfr4" index="19" on_pgm="false" on_prev="false"/>
<column name="bfr5" index="20" on_pgm="false" on_prev="false"/>
<column name="bfr6" index="21" on_pgm="false" on_prev="false"/>
<column name="bfr7" index="22" on_pgm="false" on_prev="false"/>
<column name="bfr8" index="23" on_pgm="false" on_prev="false"/>
<column name="bfr9" index="24" on_pgm="false" on_prev="false"/>
<column name="bfr10" index="25" on_pgm="false" on_prev="false"/>
<column name="bfr11" index="26" on_pgm="false" on_prev="false"/>
<column name="bfr12" index="27" on_pgm="false" on_prev="false"/>
```

```
<column name="bfr13" index="28" on_pgm="false" on_prev="false"/>
<column name="bfr14" index="29" on_pgm="false" on_prev="false"/>
<column name="bfr15" index="30" on_pgm="false" on_prev="false"/>
<column name="ddr1_a" index="31" on_pgm="false" on_prev="false"/>
<column name="ddr1_b" index="32" on_pgm="false" on_prev="false"/>
<column name="ddr2_a" index="33" on_pgm="false" on_prev="false"/>
<column name="ddr2_b" index="34" on_pgm="false" on_prev="false"/>
<column name="gfx1_a" index="35" on_pgm="false" on_prev="false"/>
<column name="gfx1_b" index="36" on_pgm="false" on_prev="false"/>
<column name="gfx2_a" index="37" on_pgm="false" on_prev="false"/>
<column name="gfx2_b" index="38" on_pgm="false" on_prev="false"/>
<column name="ddr1" index="39" on_pgm="false" on_prev="false"/>
<column name="ddr2" index="40" on_pgm="false" on_prev="false"/>
<column name="gfx1" index="41" on_pgm="true" on_prev="false"/>
<column name="gfx2" index="42" on_pgm="true" on_prev="false"/>
<column name="v1" index="43" on_pgm="false" on_prev="false"/>
<column name="v2" index="44" on_pgm="false" on_prev="false"/>
<column name="v3" index="45" on_pgm="false" on_prev="false"/>
<column name="v4" index="46" on_pgm="false" on_prev="false"/>
<column name="preview" index="47" on_pgm="false" on_prev="false"/>
<column name="me_preview" index="48" on_pgm="false" on_prev="false"/>
<column name="me_follow" index="49" on_pgm="false" on_prev="false"/>
<column name="previz" index="50" on_pgm="false" on_prev="false"/>
<column name="web_follow" index="51" on_pgm="false" on_prev="false"/>
<column name="sound" index="-2" on_pgm="false" on_prev="false"/>
<column name="black" index="-1" on_pgm="false" on_prev="false"/>
  </tally>
```

Other system information can also be obtained in this manner. For example, getting the following URL returns information about the system, software, and session: http:// NewTek live video mixer SystemName /version

An example response would look like the following:

```
<product_information>
  <product_model>TC1</product_model>
  <product_name>TriCaster TC1</product_name>
  <product_version>7-0</product_version>
  <product_id>NCWL-WFKNJ8YAA-200918</product_id>
  <product_serial_no/>
  <product_build_no>7-0-180920</product_build_no>
  <machine_name>TC1</machine_name>
  <session_x_resolution>1920</session_x_resolution>
  <session_y_resolution>1080</session_y_resolution>
  <session_fielded>true</session_fielded>
  <session_frame_rate>29.970030</session_frame_rate>
  <session_aspect_ratio>1.777778</session_aspect_ratio>
```

```
    <session_color_format>CCIR709</session_color_format>
    <session_color_coding>NTSC</session_color_coding>
    <session_name>TC1 Session</session_name>
</product_information>
```

To offer another example, requesting the (example) URL below returns a wealth of switcher parameters.

http:// NewTek live video mixer SystemName/v1/dictionary?key=switcher

The result would include:

- All inputs, physical and virtual
- Switcher row sources for M/Es, including up to 4 rows if using a LiveSet
- Overlay information for main and M/E switchers
    - Overlay source
    - T-Bar position
- Currently loaded Effect (Transition or LiveSet)

An example response would look like the following:

```
<switcher_update main_source="BFR4" preview_source="gfx1" effect="Q:\Products\
      NewTek live video mixer _Content\Animation Stores\Output\Broadcast\Door
      Slam.effect">
  <tbar position="0.000000" speed="0.000000"/>
  <switcher_overlays>
    <overlay z_order_position="0" source="V2">
      <tbar position="0.000000" speed="0.000000"/>
    </overlay>
    <overlay z_order_position="1" source="BFR1">
    <tbar position="0.000000" speed="0.000000"/>
```

Or, requesting the URL below will return a list of the currently-assigned buffers for the main switcher row and the M/E rows, similar to the following example:

http:// NewTek live video mixer SystemName/v1/dictionary?key=buffer

:

```
<buffers>
  <main>
    <buffer selection="BFR4"/>
  </main>
  <me index="0">
  <row>
    <buffer selection="BFR1"/>
  </row>
</me>
...
```

The URL below returns a list of all the effects loaded into all the bins for each transition and overlay in the main Switcher and all M/Es.

http:// NewTek live video mixer SystemName/v1/dictionary?key=switcher_ui_effects
An example result follows:

```
<switcher_ui_effects>
<switcher name="main">
<effect_bin>
<effect0 effect="Fade"/>
<effect1 effect="Q:\Products\ NewTek live video mixer _Content\Animation
    Stores\Output\Broadcast\Door Slam.effect"/>
<effect2 effect="c:\Program Files\NewTek\...\Effects\Transitions\Fades\Non Additive
    Fade.trans"/>
<effect3 effect="c:\Program Files\NewTek\...\Effects\Transitions\Fades\Flash.trans"/>
<effect4 effect="c:\Program Files\NewTek\...\Effects\Transitions\Fades\Clouds.trans"
...
<effect7 effect="c:\Program Files\NewTek\...\Effects\Overlays\Iris\Hard\Rectangle
    (H).ofx"/>
<effect8 effect="c:\Program Files\NewTek\...\Effects\Overlays\Iris\Hard\Circle(H).ofx"/>
</effect_bin>
</key>
</switcher>
<switcher name="v3">
<effect_bin>
<effect0 effect="Fade"/>
<effect1 effect="c:\Program Files\NewTek\...\Effects\Transitions\Fades\Additive
    Fade.trans"/>
<effect2 effect="c:\Program Files\NewTek\...\Transitions\Fades\Non Additive
    Fade.trans"/>
<effect3 effect="c:\Program Files\NewTek\...\Transitions\Fades\Flash.trans"/>
<effect4 effect="c:\Program Files\NewTek\...\Transitions\Fades\Clouds.trans"/>
<effect5 effect="c:\Program Files\NewTek\...\Transitions\Fades\Noise.trans"/>
<effect6 effect="c:\Program Files\NewTek\...\Transitions\Iris\Hard\Circle(H).trans"/>
      ...
<switcher name="v8">
<effect_bin>
<effect0 effect="C:\Program Files\NewTek\...\Effects\LiveSets\User\Rock Stage Standing
    Flares 03 RAW\Rock Stage Standing Flares 03 RAW.LiveSet"/>
</effect_bin>
<key>
<effect_bin>
<effect0 effect="Fade"/>
<effect1 effect="c:\Program Files\NewTek\...\Effects\Overlays\Trajectories\Fly In\Fly In
    B.ofx"/>
```

The complete list of key values supported for dictionary use in the manner disclosed in the preceding examples follows below:

- shortcut_states
- tally
- ddr_playlist
- ddr_timecode
- buffer
- switcher
- buffer
- switcher_ui_effects
- audiomixer
- audio_bins
- audiomixer
- filebrowser
- macros_list
- ndi_sources

7.3.9 WEBSOCKETS

The WebSocket protocol is supported by most major web browsers and allows Vizrt systems to push data (inluding a great deal of system status information along with image previews and audio data) to the client. This allows improved responsiveness by notifying clients of state changes, removing the need for clients to peridically poll for changes.

CONNECTING

Vizrt systems support a number of different web socket connections. These can variously receive notifications of state changes or quickly execute shortcut commands. The process for establishing a web socket connection is the same regardless of the connection type. For example, to receive state change notifications, first open a connection to the Vizrt system with a web socket on port 5951 using the path "/v1/change_notifications" and the WebSocket protocol. Then listen for data coming in on that connection.

To keep the connection active for lengthy periods, you may send the web server ping frames periodically (every 15 seconds would be fine). Another common approach is to wrap the connection in a function that is called upon its closing.

The following is a list of available web socket addresses, which will vary depending on the system.

- v1/audio_notifications
- v1/change_notifications
- v1/shortcut_notifications
- v1/shortcut_state_notifications
- v1/timecode_notifications
- v1/vu_notifications

Opening a Connection in Javascript

```
var url = 'ws://' + ipAddress + 'v1/change_notifications';
var ws = WebSocket( url );
```

Hint: See also Exporting Macros in Section 7.2.1.

State Change Notifications

The "v1/change_notifications" web socket connection notifies the client of state changes (for example a clip being added to or removed from a DDR, or a change to tally). The only data sent over this connection is the name of the served state page whose content has changed. A subsequent HTTP request is required in order to obtain the actual updated state data.

For example, if the text received is "tally", the web page located at "/v1/dictionary?key=tally" has changed. Notification messages will pertain to one of the following key values, depending on the action on the device.

- shortcut_states
- tally
- ddr_playlist
- buffer
- switcher
- switcher_ui_effects
- audiomixer
- audio_bins
- macros_list
- ndi_sources

State Change Message Processing

The example code below creates a WebSocket that renews itself upon closing, has handler functions for onopen, onclose, and onmessage, and gives three stubs for processing received messages.

```
function createWebsocket() {
        var url = 'ws://' + ipAddress + 'v1/change_notifications';
        var ws = new WebSocket( url );
}

// create the socket the first time
createWebsocket();

// Attach event handlers. Standard WebSocket handlers are supported.

ws.onopen = function() {
    console.log("TriCaster WebSocket Opened");
}
```

```
// The incoming message data will be one of the states listed above.
ws.onmessage = function( msg ) {
      if (msg.data == "tally") {
        // do tally things
          }
          else if (msg.data == "switcher") {
            // do switcher things
          }
          // …
}

ws.onclose = function() {
      createWebsocket();   // Upon closing, reopen the socket
}
```

Video Previews

Some Vizrt products (such as TriCaster) also support the serving of video preview streams over web-sockets. The web socket location takes the form shown below:

http://SystemNameOrIPAddress/v1/video_notifications?name=NAME&xres=RESX&yres=RESY&q=QUALITY

(Parameters in the above example are described in Section 3.4, and work the same way.)

A web socket will be opened and JPG images will be streamed to it as fast as the network can send them. Each frame is sent in a single send operation, so you know the size of the JPG data from the size of the received packet.

Receiving/Sending Audio

Vizrt systems may also allow you to observe audio over a web socket. To do so, use the following format to create a connection:

http://SystemNameOrIPAddress/v1/audio_notifications?name=NAME

Supported audio names are listed below:

- output
- aux
- phones

There following caveats are important:

- Audio is uncompressed stereo 44.1 kHz, in signed 16 bit format. This requires approximately 180 Kb/s over the network so make sure you have a good enough connection.

- The implementation does not do dynamic audio resampling. Thus occasional audio glitches may be expected, since the client computer is not locked to TriCaster's audio clock.

TriCaster also supports serving VU data over web sockets. The web socket address takes the form shown below:

http://SystemNameOrIPAddress /v1/vu_notifications?name=NAME

Receiving/Sending Shortcuts

For some shortcuts, state change notifications that require a subsequent HTTP request could make an app feel slow or unresponsive. The "v1/shortcut_state_notifications" web socket connection not only notifies the client that a shortcut has changed, but includes its updated value as well, omitting the need for any HTTP request.

The "v1/shortcut_state" web socket can be used to quickly issue a shortcut command when performance is important (like updating a volume slider). No data is ever received by the client on this connection, it is only meant for sending data from the client to the system.

Issuing a Shortcut via WebSocket in Javascript

```
var url = 'ws://' + ipAddress + 'v1/shortcut_state';
var ws = WebSocket( url );
ws.send( 'name=<shortcut_name>&value=<shortcut_value>' );.
```

NOTES: A document named "AMP Video Server Setup", available from the Ross Video website, details how to configure a Carbonite system for AMP communication.

Chapter 8 FILES AND STORAGE

Some corollary of 'Murphy's Law' must state that "The more critical it is to a production, the greater the likelihood that important media will be delivered at the last possible moment and the wrong format, if it can be found at all." This section is devoted to reducing your stress level when this inevitably occurs.

Section 8.1 MEDIA FILE FORMATS

8.1.1 VIDEO CAPTURE

Current Vizrt live video mixer products support capture as Quicktime (.mov) files in Vizrt's own SpeedHQ (SHQ) format. In general, where other considerations permit, we recommend the use of these native Quicktime formats for high quality video capture. (Alternatively, the ENCODE feature on some products supports real time encoding as H.264 in an MPEG-4 (MP4) container, which may be helpful when smaller file size is important.)

The Quicktime format supports all Vizrt live video mixer features, including embedded timecode, and provides high quality capture suitable for almost any purpose. Most modern software applications can work with the Quicktime format (some applications may require installation of Vizrt Quicktime codecs).

Note: Vizrt live video mixer and Viz 3Play systems are able to play back Quicktime files that are still 'growing' (being actively captured). Some external software applications also enjoy the same benefits. (For example, Adobe's Premiere and After Effects applications support growing files courtesy of plugins included in the NDI Tools suite.)

8.1.2 VIZRT CODECS

Codecs are available for the Windows platform supporting both Vizrt's high quality AVI and Quicktime capture formats. The codecs can be found in the "Extras" folder of our live production systems or, alternatively, can be downloaded from the Vizrt website's Support pages.

Hint: Users of Vizrt live video systems need not separately install these codecs.

OS X PLATFORM

Reading Vizrt's Quicktime files on Apple platforms requires the installation of third-party tools (such as one of the Adobe® applications and plugins mentioned earlier, or VLC™ from VideoLAN™).

Section 8.2 IMPORT

Vizrt live video procucts are able to play back media files in many different popular file formats, but some of these require more system resources to play than others. With a view to making best use of precious resources, then, media files should ideally be prepared beforehand.

With the assistance of the Vizrt codec packs just mentioned, it is often true that media can be created in or converted to Vizrt-friendly formats that are easy to play back right in your favorite non-linear editor or compositing package. We can recommend Vizrt's own SpeedHQ Quicktime or AVI encoding as a good high quality format for use in any of our live production systems. (SpeedHQ options include support for files with embedded alpha channel, especially valuable for animations intended for use as overlays.)

Alternatively, Vizrt live production systems generally include dedicated *Import* modules. This module, typically provided in the *File* menu at upper left in the Live Desktop (see product manuals for details) provides a way to add multiple items to a queue for batch processing, including optional transcoding, as necessary.

Otherwise, most high quality Quicktime formats (other than ProRes) will work well. For HD files, you might consider trying the Quicktime PNG encoder (especially when an alpha channel is required).

Section 8.3 EXPORT

At times you may wish to export files recorded with a Vizrt live production in some other popular format. Of course, whether working with files captured to shared storage systems or copied to external media across a network, for example, you could perform transcoding entirely externally using your favorite conversion software. You may instead, though, wish to use your Vizrt system to do transcoding.

All current Vizrt live production platforms include an Export feature in the system's Startup pages. Most also provide a Publish Queue in the Live or Replay Desktop. Files can be added to the lists in these modules using a variety of method, even – in this latter case – during live production. The modules provide access to a deep set of transcoding tools and control over destinations for file output (including local and networked volumes, and ftp).

Section 8.4 ASSET MANAGEMENT

The integrated *Media Browser* native to Vizrt live production systems is a competent asset management system, enabling you to quickly locate and work with files related to your sessions, or external files.

Of course, more extensive media asset management systems provided by leading industry providers are also available within the Vizrt ecosystem, and may be directly supported by Vizrt products.

To utilize your favorite (supported) third-party asset management systems, you need only hold down the keyboard Ctrl key when invoking a file browser. For example, double-clicking a blank spot in a DDR playlist on a Vizrt live video mixer with Ctrl depressed will show your compatible custom asset management interface, rather than Vizrt live video mixer's native *Media Browser*.

Hint: Alternatively, you can open a standard system file explorer, by holding down the Shift key rather than Ctrl when adding files.

Asset management solutions can include outboard storage systems, so let's talk about this related matter.

Section 8.5 External Storage

Vizrt live productions systems provide substantial integrated storage for media used in your productions, by means of internal and removable drives. Of course, many broadcast environments have still larger requirements, making external storage solutions attractive. In addition, by virtue of their potential for massive capacity, fails-safe mechanisms, transfer speed, and shared access, external storage solutions can facilitate file ingest, shared access, media updates, and more.

Large storage solutions come in many varieties, including SAN (Storage Area Network), NAS (Networked Attached Storage) and others. Individual solutions may include dedicated MAM (Media Access Management) implementations, or not.

In general, we recommend the use of the NTFS file system, not least because it effectively manages files larger than four gigabytes (unlike FAT32, for example), but also because it fully supports the Vizrt video system features.

At times, though, you may prefer to employ another file system for media used for video capture or file sharing. If so, note that it's best if the actual "Session Volume" for a Vizrt live video mixer or Viz 3Play is still NTFS-formatted. Otherwise, links to media captured during the session that are automatically generated by the system may fail, forcing you to expend extra effort to locate them.

APPENDICES

APPENDIX A. DATALINK HARDWARE KEYS

This section lists the actual key names that are available for use with DataLink for the assorted brands of external equipment it supports. Mostly, the key names are self-explanatory, but we've added slightly more descriptive notes where appropriate. The list is grouped by manufacturer.

Note: the key names listed are shown inserted between percent (%) signs as a reminder, since this is how you will enter them onto your pages.

A.1 DAKTRONICS

A.1.1 BASEBALL

%DakClock% - Game Clock Time – "MM:SS.T"

%DakClockStatus% - Game Clock Status

%DakHomeHits% - Home Team Hits

%DakGuestScore% - Guest Team Score

%DakInning% - Current inning

%DakHhr% - Hour (from Clock Time)

%DakMin% - Minutes (from Clock Time)

%DakSec% - Seconds (from Clock Time)

%DakTen% - Tenths (secs/10 from Clock Time)

A.1.2 BASKETBALL

%DakClock% - Game Clock Time – "MM:SS.T"

%DakClockStatus% - Game Clock Status

%DakShotClock% - Shot Clock Time – "SS"

%DakHomeScore% - Home Team Score

%DakGuestScore% - Guest Team Score

%DakHomeFouls% - Home Team Fouls

%DakGuestFouls% - Guest Team Fouls

%DakHomeTOFull% - Home Time Outs Left – Full

%DakHomeTOPart% - Home Time Outs Left – Partial

%DakHomeTOTotal% - Home Time Outs Left – Total

%DakGuestTOFull% - Guest Time Outs Left – Full

%DakGuestTOPart% - Guest Time Outs Left – Partial

%DakGuestTOTotal% - Guest Time Outs Left – Total

%DakPeriod% - Current period

%DakHhr% - Hour (from Clock Time)

%DakMin% - Minutes (from Clock Time)

%DakSec%- Seconds (from Clock Time)

%DakTen% - Tenths (secs/10 from Clock Time)

A.1.3 FOOTBALL

%DakClock% - Game Clock Time – "MM:SS.T"

%DakClockStatus% - Game Clock Status

%DakPlayClock%% - Play Clock Time – "SS"

%DakHomeScore% - Home Team Score

%DakGuestScore% - Guest Team Score

%DakHomeTOFull% - Home Time Outs Left – Full

%DakHomeTOPart% - Home Time Outs Left – Partial

%DakHomeTOTotal% - Home Time Outs Left – Total

%DakGuestTOFull% - Guest Time Outs Left – Full

%DakGuestTOPart% - Guest Time Outs Left – Partial

%DakGuestTOTotal% - Guest Time Outs Left – Total

%DakQuarter% - Current quarter

%DakMin% - Minutes (from Clock Time)

%DakSec%- Seconds (from Clock Time)

%DakTen% - Tenths (secs/10 from Clock Time)

A.1.4 HOCKEY

%DakClock% - Game Clock Time – "MM:SS.T"

%DakClockStatus% - Game Clock Status

%DakShotClock%% - Shot Clock Time – "SS"

%DakHomeScore% - Home Team Score

%DakGuestScore% - Guest Team Score

%DakHomeTOFull% - Home Time Outs Left – Full

%DakHomeTOTotal% - Home Time Outs Left – Total

%DakGuestTOFull% - Guest Time Outs Left – Full

%DakGuestTOTotal% - Guest Time Outs Left – Total

%DakPeriod% - Current period

%DakMin% - Minutes (from Clock Time)

%DakSec%- Seconds (from Clock Time)

%DakTen% - Tenths (secs/10 from Clock Time)

A.1.5 SOCCER

%DakClock% - Game Clock Time – "MM:SS.T"

%DakClockStatus% - Game Clock Status

%DakShotClock%% - Shot Clock Time – "SS"

%DakHomeScore% - Home Team Score

%DakGuestScore% - Guest Team Score

%DakHomeTOFull% - Home Time Outs Left – Full

%DakGuestTOFull% - Guest Time Outs Left – Full

%DakGuestTOTotal% - Guest Time Outs Left – Total

%DakHalf% - Current half

%DakMin% - Minutes (from Clock Time)

%DakSec%- Seconds (from Clock Time)

%DakTen% - Tenths (secs/10 from Clock Time)

A.1.6 VOLLEYBALL

%DakClock% - Game Clock Time – "MM:SS.T"

%DakClockStatus% - Game Clock Status

%DakHomeServiceIndicator%

%DakHomeScore% - Home Team Score

%DakGuestScore% - Guest Team Score

%DakHomeTOFull% - Home Time Outs Left – Full

%DakHomeTOTotal% - Home Time Outs Left – Total

%DakGuestTOFull% - Guest Time Outs Left – Full

%DakGuestTOTotal% - Guest Time Outs Left – Total

%DakGameNumber% - Current game number

%DakMin% - Minutes (from Clock Time)

%DakSec%- Seconds (from Clock Time)

%DakTen% - Tenths (secs/10 from Clock Time)

A.2 DAKTRONICS CG

A.2.1 BASEBALL

%CGDakHomeScore% - Home Team Score

%CGDakGuestScore% - Guest Team Score

%CGDakInning% - Current inning

%CGDakInningText% - Current inning (text)

%CGDakInningDescription% - Inning Description (text)

%CGDakHomeAtBat% - Home At-bat indicator (0 or 1).

%CGDakGuestAtBat% - Guest At-bat indicator (0 or 1).

%CGDakHomeHits% - Home Team Hits

%CGDakHomeErrors% - Home Team Errors

%CGDakHomeLeftOnBase% - Home Team Left-on-base

%CGDakGuestHits% - Guest Team Hits

%CGDakGuestErrors% - Guest Team Errors

%CGDakGuestLeftOnBase% - Guest Team Left-on-base

%CGDakBatterNumber% - At-bat Player Number

%CGDakBatterAverage% - At-bat Player Average

%CGDakBall% - Ball count

%CGDakStrike% - Strike count

%CGDakOut% - Outs

%CGDakHit% - Hits

%CGDakError% - Errors

%CGDakHitErrorText% - Error (text)

%CGDakErrorPosition% - Error Position

%CGDakInningLabel1% - First Inning label

%CGDakInningLabel2% - etc.

%CGDakInningLabel3%

%CGDakInningLabel4%

%CGDakInningLabel5%

%CGDakInningLabel6%

%CGDakInningLabel7%

%CGDakInningLabel8%

%CGDakInningLabel9%

%CGDakInningLabel10%

%CGDakInningLabel11%

%CGDakInningLabel12%

%CGDakHomeInningScore1% - Home Score, First Inning

%CGDakHomeInningScore2% - etc.

%CGDakHomeInningScore3%

%CGDakHomeInningScore4%

%CGDakHomeInningScore5%

%CGDakHomeInningScore6%

%CGDakHomeInningScore7%

%CGDakHomeInningScore8%

%CGDakHomeInningScore9%

%CGDakHomeInningScore10%

%CGDakHomeInningScore11%

%CGDakHomeInningScore12%

%CGDakGuestInningScore1% - Guest Score, First Inning

%CGDakGuestInningScore2% - etc.

%CGDakGuestInningScore3%

%CGDakGuestInningScore4%

%CGDakGuestInningScore5%

%CGDakGuestInningScore6%

%CGDakGuestInningScore7%

%CGDakGuestInningScore8%

%CGDakGuestInningScore9%

%CGDakGuestInningScore10%

%CGDakGuestInningScore11%

%CGDakGuestInningScore12%

%CGDakHomePitcherNum%- Home Pitcher Player Number

%CGDakHomePitchesBalls% - Home Pitches, Balls

%CGDakHomePitchesStrikes% - Home Pitches, Strikes

%CGDakHomePitchesFoulBall%- Home Pitches, Foul Balls

%CGDakHomePitchesInPlay% - Home Pitches In Play

%CGDakHomePitchesTotal% - Total Home Pitches

%CGDakGuestPitcherNum%- Guest Pitcher Player Number

%CGDakGuestPitchesBalls%- Guest Pitches, Balls

%CGDakGuestPitchesStrikes% - Guest Pitches, Strikes

%CGDakGuestPitchesFoulBall%- Guest Pitches, Foul Balls

%CGDakGuestPitchesInPlay% - Guest Pitches In Play

%CGDakGuestPitchesTotal% - Total Guest Pitches

A.2.2 BASKETBALL

%CGDakClock% - Game Clock Time – "MM:SS.T"

%CGDakClockStatus% - Game Clock Status

%CGDakShotClock% - Shot Clock Time – "SS"

%CGDakHomeScore% - Home Team Score

%CGDakGuestScore% - Guest Team Score

%CGDakHomeFouls% - Home Team Fouls

%CGDakGuestFouls% - Guest Team Fouls

%CGDakHomeTOFull% - Home Time Outs Left – Full

%CGDakHomeTOPart%% - Home Time Outs Left – Partial

%CGDakHomeTOTotal%- Home Time Outs Total

%CGDakGuestTOFull% - Guest Time Outs Left – Full

%CGDakGuestTOPart% - Guest Time Outs Left – Partial

%CGDakGuestTOTotal%- Guest Time Outs Left – Total

%CGDakPeriod% - Current period

%CGDakMin%- Minutes (from Clock Time)

%CGDakSec% - Seconds (from Clock Time)

%CGDakTen%- Tenths (secs/10 from Clock Time)

A.2.3 FOOTBALL

%CGDakClock% - Game Clock Time – "MM:SS.T"

%CGDakHomeTeamName%- Home Team Name

%CGDakGuestTeamName%- Guest Team Name

%CGDakHomeScore% - Home Team Score

%CGDakGuestScore% - Guest Team Name

%CGDakQuarter% - Current quarter

%CGDakBallOn% - Current ball position

%CGDakDown% - Current down

%CGDakToGo% - Yards to go

%CGDakHomePossess%- Possession indicator (0 or 1).

%CGDakGuestPossess%- Possession indicator (0 or 1).

%CGDakPlayClock% - Play Clock Time – "SS"

%CGDakHomeTO% - Home Time Outs

%CGDakGuestTO% - Guest Time Outs

%CGDakMin%- Minutes (from Clock Time)

%CGDakSec% - Seconds (from Clock Time)

%CGDakTen%- Tenths (secs/10 from Clock Time)

A.2.4 HOCKEY

%CGDakClock% - Game Clock Time – "MM:SS.T"

%CGDakClockStatus% - Game Clock running status indicator

%CGDakHomeScore% - Home Team Score

%CGDakGuestScore% - Guest Team Score

%CGDakHomeTO% - Home Time Outs

%CGDakGuestTO%% - Guest Time Outs

%CGDakHomeShotsOnGoal% - Home Shots on Goal

%CGDakGuestShotsOnGoal% - Guest Shots on Goal

%CGDakPeriod% - Current period

%CGDakHomePenalty1_PlayerNum% - Home Penalty, player number

%CGDakHomePenalty1_PenaltyTime%- Home Penalty, time left

%CGDakGuestPenalty1_PlayerNum% - Guest Penalty, player number

%CGDakGuestPenalty1_PenaltyTime%- Guest Penalty, time left

%CGDakHomePenalty2_PlayerNum% - Home Penalty, player number

%CGDakHomePenalty2_PenaltyTime%- Home Penalty, time left

%CGDakGuestPenalty2_PlayerNum% - Guest Penalty, player number

%CGDakGuestPenalty2_PenaltyTime%- Guest Penalty, time left

%CGDakMin%- Minutes (from Clock Time)

%CGDakSec% - Seconds (from Clock Time)

%CGDakTen%- Tenths (secs/10 from Clock Time)

A.2.5 SOCCER

%CGDakClock% - Game Clock Time – "HH:MM:SS.T"

%CGDakHomeTeamName%- Home Team Name

%CGDakGuestTeamName%- Guest Team Name

%CGDakHomeScore% - Home Team Score

%CGDakGuestScore% - Guest Team Score

%CGDakHalf% - Current half

%CGDakHomeShotsOnGoal% - Home Shots on Goal

%CGDakHomeSaves% - Home Saves

%CGDakHomeCornerKicks% - Home Corner Kicks

%CGDakGuestShotsOnGoal% - Guest Shots on Goal

%CGDakGuestSaves% - Guest Saves

%CGDakGuestCornerKicks% - Guest Corner Kicks

%CGDakHomeFouls% - Home Fouls

%CGDakGuestFouls% - Guest Fouls

%CGDakHhr% - Hours (from Clock Time)

%CGDakMin% - Minutes (from Clock Time)

%CGDakSec% - Seconds (from Clock Time)

%CGDakTen% - Tenths (secs/10 from Clock Time)

A.2.6 VOLLEYBALL

%CGDakClock% - Game Clock Time – "MM:SS.T"

%CGDakClockStatus% Game clock running status indicator

%CGDakHomeGameScore% - Home Team Score

%CGDakGuestGameScore% - Guest Team Score

%CGDakHomeTO% - Home Time Out

%CGDakGuestTO% - Guest Time Out

%CGDakHomeServiceIndicator% - Home Service indicator (0 or 1)

%CGDakGuestServiceIndicator% - Guest Service indicator (0 or 1)

%CGDakHomeGamesWon% - Home Games Won

%CGDakGuestGamesWon% - Guest Games Won

%CGDakGameNumber% - Current Game Number

%CGDakHomeGameScore1% - Home Score, First Game

%CGDakHomeGameScore2% - Home Score, Second Game

%CGDakHomeGameScore3% - Home Score, Third Game

%CGDakHomeGameScore4% - Home Score, Fourth Game

%CGDakGuestGameScore1% - Guest Score, First Game

%CGDakGuestGameScore2% - Guest Score, Second Game

%CGDakGuestGameScore3% - Guest Score, Third Game

%CGDakGuestGameScore4% - Guest Score, Fourth Game

%CGDakMin% - Minutes (from Clock Time)

%CGDakSec% - Seconds (from Clock Time)

%CGDakTen% - Tenths (from Clock Time)

A.3 DSI KEYS:

A.3.1 BASKETBALL

%DSIClock% - Game Clock Time – "MM:SS.T"

%DSIShotClock% - Shot Clock Time – "SS"

%DSIMin% - Minutes (from Clock Time)

%DSISec% - Seconds (from Clock Time)

%DSITen% - Tenths (secs/10 from Clock Time)

A.4 OES

A.4.1 BASKETBALL

%OESClock% - Game Clock Time – "MM:SS.T"

%OESShotClock% - Shot Clock Time

%OESAwayScore% - Guest Team Score

%OESHomeScore% - Home Team Score

%OESHomeFouls% - Home Team Fouls

%OESAwayFouls% - Guest Team Fouls

%OESHomeTOFull% - Home Team Time Out - Full

%OESHomeTOPart% - Home Team Time Out - Partial

%OESAwayTOFull% - Guest Team Time Out - Full

%OESAwayTOPart% - Guest Team Time Out - Partial

%OESPeriod% - Current period

%OESMin% - Minutes (from Clock Time)

%OESSec% - Seconds (from Clock Time)

%OESTen% - Tenths (secs/10 from Clock Time)

A.5 TRANSLUX FAIRPLAY

A.5.1 FOOTBALL

%TLFPClock% - Game Clock Time – "MM:SS.T"

%TLFPQuarter% - Current quarter

%TLFPHomeScore% - Home Team Score

%TLFPVisitorScore% - Visiting Team Score

%TLFPDown% - Current down

%TLFPToGo% - To go (yards

%TLFPBallOn% - Ball on (yard line)

%TLFPFieldTimer% - Current field timer (SS)

%TLFPMin% - Minutes (from Clock Time)

%TLFPSec% - Seconds (from Clock Time)

%TLFPTen% - Tenths (secs/10 from Clock Time)

A.6 WHITEWAY

A.6.1 BASKETBALL

%WWPeriod% - Current period

%WWClock% - Game Clock Time – "MM:SS.T"

%WWAwayScore% - Guest Team Score

%WWHomeScore% - Home Team Score

%WWShotClock% - Shot Clock Time

%WWMin% - Minutes (from Clock Time)

%WWSec% - Seconds (from Clock Time)

%WWTen% - Tenths (secs/10 from Clock Time)

A.7 WHITEWAY RAINBOW

A.7.1 BASKETBALL

%WWRSportNum% - Sport Number

%WWRPeriod% - Current period

%WWRShotClock% - Shot Clock Time

%WWRAwayScore% - Guest Team Score

%WWRHomeScore% - Home Team Score

%WWRMinutes% - Minutes (from Clock Time)

%WWRSeconds% - Seconds (from Clock Time)

%WWRTenths% - Tenths (from Clock Time)

INDEX

N

P

R

S

T

V

W

CREDITS

Special thanks to each member of the hard-working R&D team who made this product possible.

Third Party Licenses:

This product uses a number of third-party software libraries under license. Related license requirements are defined in documentation installed on the product. To view these licenses, please click the Additional Licenses link provided in the Help menu on the Startup>Home page shown upon launching the product.