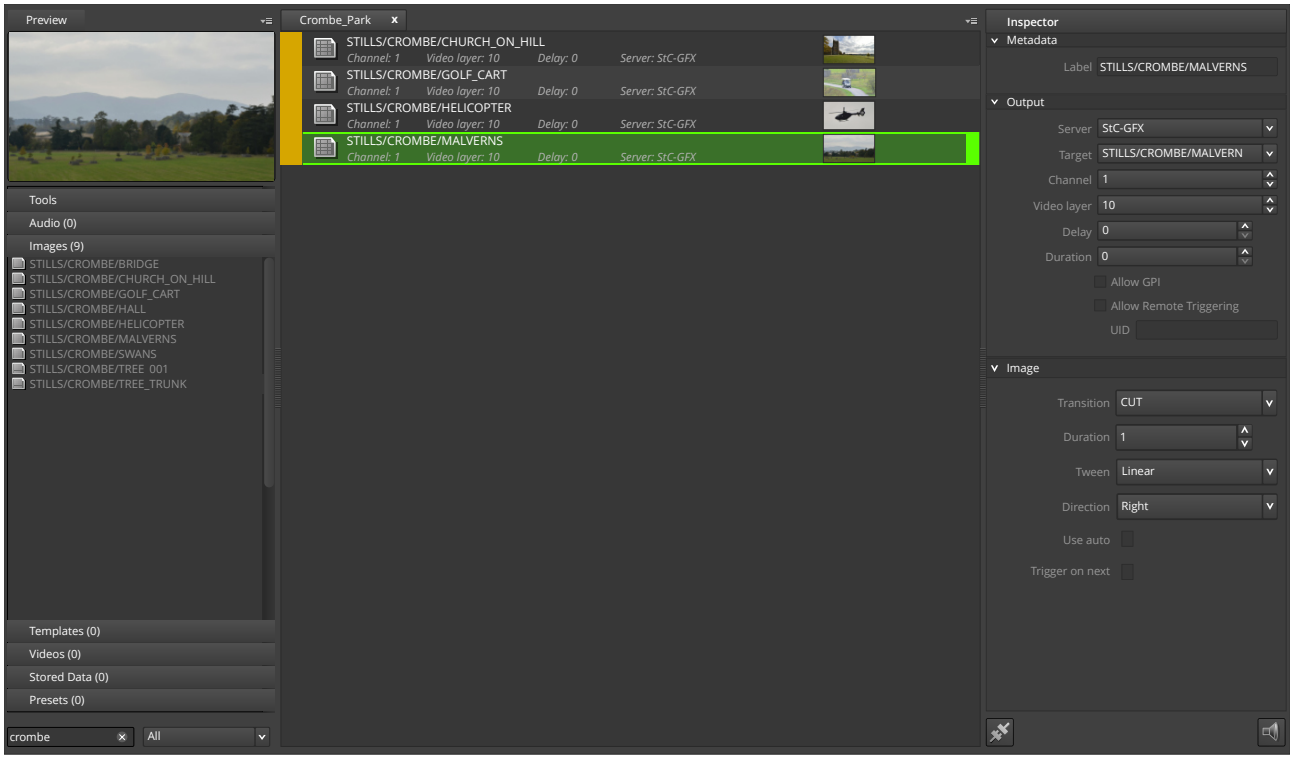
An Overview of CasparCG Graphics System



Andy Woodhouse

About this document

My thanks go to the developers of the CasparCG system in SVT, and for SVT deciding to make the software available on an open source licence. I also want to acknowledge the support provided to all CasparCG users by the members of the CasparCG Support Forum.

This is version 1.0 of the document

Andy Woodhouse

July 2021

Email: andy@amwtech.co.uk

Preface

I discovered CasparCG some years ago through a posting in the Blackmagic Design User Forum. Since that discovery I have installed multiple instances of CasparCG and introduced many individuals to its potential applications. I have created various templates, and used the deployed systems in many training environments.

My aim in publishing this document is that it will help other new users learn the capabilities of CasparCG and speed them to further investigations.

There has been no formal publication edit process, so I apologise for any errors in spelling or grammar that have slipped through.

CasparCG server can be deployed on both Windows and Linux operating systems, but for simplicity this document only covers the operations on a Windows host system. I have also decided to concentrate the descriptions on server version 2.3.3 Long Term Support working with client version 2.2 as these are the newest versions available at the time of writing.

Andy Woodhouse.

Contents

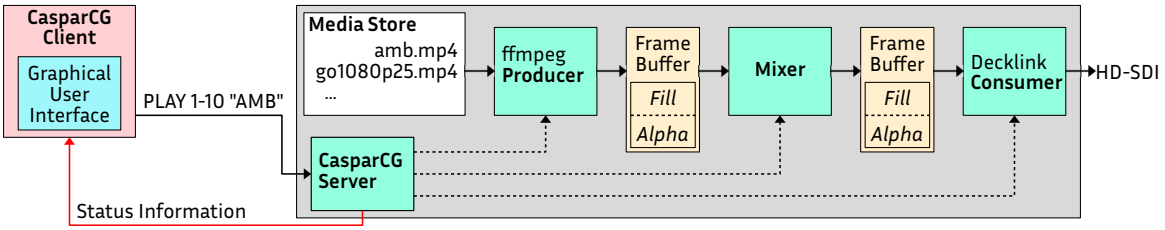
System Overview

CasparCG is a broadcast graphics and video playback system created by Sveriges Television (SVT). It was initially developed for use within SVT, but subsequently released as open source software under version 3 of the GNU General Public Licence.

CasparCG uses a client-server architecture. CasparCG ***clients*** send text-based control commands, known as AMCP, to a CasparCG ***server***.

The server software manages the replay of audio clips, video clips, rendering of templated graphics and combining multiple graphical elements into a composite bitmap. Templated graphics, such as lower thirds, combine a design layout (the template) with instance data.

An example process flow for video clip replay is illustrated below.



The client software has a graphical interface enabling the user to select a file, in this example called amb.mp4, and to designate the output channel number and layer for the content (more detail on channels and layers later in this chapter).

When the user presses the Play key the client sends the command string `PLAY 1-10 "AMB"` to the selected server. This command requests the server locate video media whose main name is AMB and to play the content on output channel 1 in layer 10. Note the command does not require the target file extension as part of the command.

The server parses the received command, looking in the media store for a matching file name. A status report is sent to the client indicating if the play process could be started (ie the file was found), or if there is some error condition.

The file name is passed to the ffmpeg producer that reads and decodes the source content, rendering the media into a frame buffer. The frame buffer has storage for both the fill content and for any associated alpha content. A full frame alpha channel is generated for files that do not have a stored alpha signal.

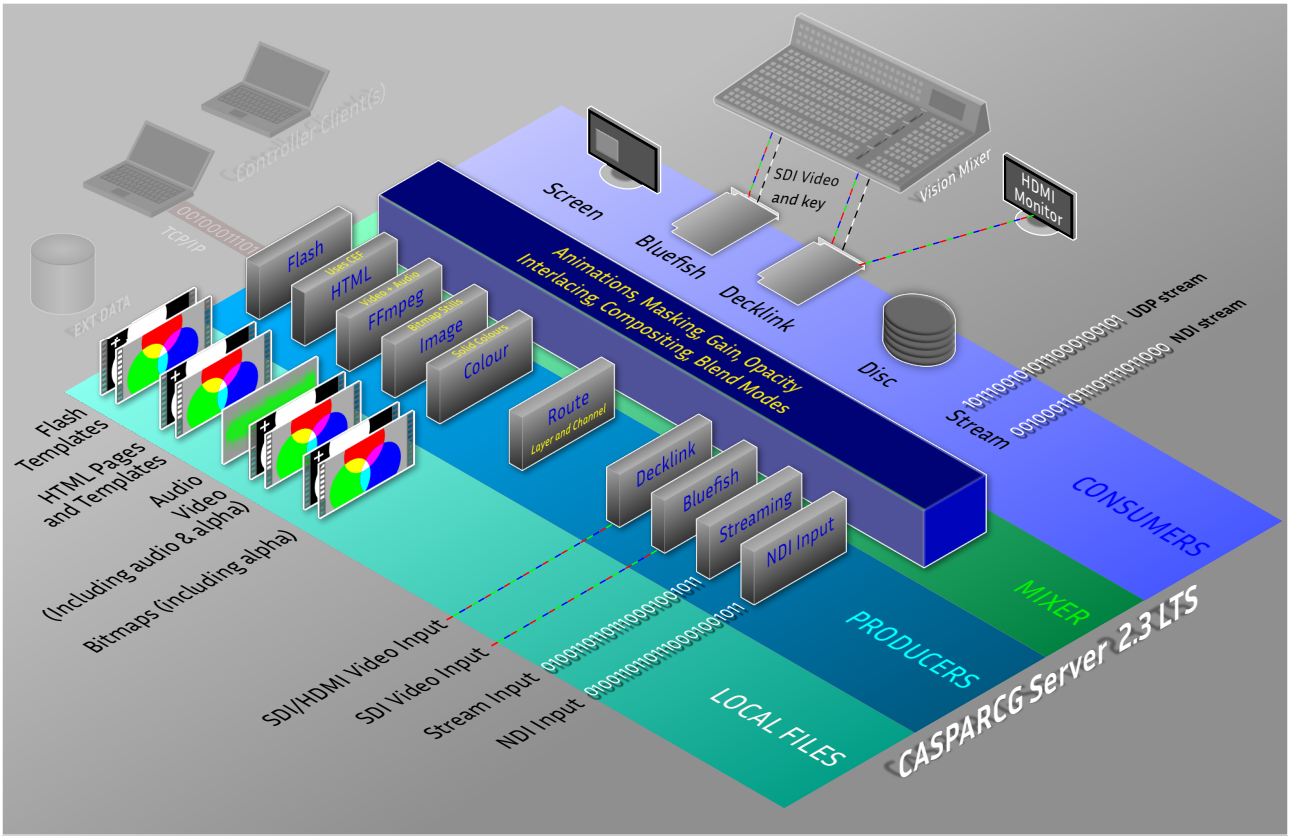
The frame buffer passes the decoded content to the Mixer process. The mixer applies any user specified size changes, position offsets or other manipulations such as opacity, combines the layer 10 content with any other layers of the assigned output channel, for example overlaying a lower third name strap. The composited output from the mixer is placed in a frame buffer.

The output of the mixer passes to one or more consumers for the channel. A consumer converts the bitmap from the mixer into a chosen output format, fro example SDI video and key. Multiple consumers can be used on a channel enabling the output as SDI whilst making a recording of that output.

In the above illustration, the consumer uses a Blackmagic Design Decklink card to output an HD-SDI output signal with embedded audio.

The server sends ongoing status about the active channels and the progress of the file replays to all connected clients. This status information enables the client software to display a progress meter showing the percentage of the clip played and a countdown to the end of media playout.

There are multiple producers and consumers. The number of producers and consumers may differ between versions of the server software. The following illustration shows the facilities in SVT version 2.3 server.



A server creates one or more logical ***channels***.Each channel is the output of a compositing process for multiple ***layers*** executed by the ***mixer*** module. Logical channel outputs are converted to physical outputs, such as SDI video, by one or more ***consumers***.

Media files are processed by the relevant ***producer*** module rendering a bitmap and associated alpha/key information into a frame buffer. The producer output is targeted at a user-specified ***layer*** of the final composite.

When the software was initially developed the ***Flash producer*** was the main mechanism used for rendering templated graphics. Most new applications of CasparCG now use HTML templates. The latest server version disables the Flash producer by default as Flash technology has reached end of life.

The ***HTML producer*** uses an embedded Chromium engine to display web pages or to implement rendering of templated graphics. The Chromium engine includes support for WebSocket technology, enabling a web server or web browser to push content directly to a display page. Specialist web-servers can be created using server side processing technologies, enabling full page graphics that continuously update, for example sports scores or weather information. A CasparCG client can still send update requests to a suitably authored HTML page, the page code in turn requests data from the web-server (for example to display a different football league).

HTML templated graphics pages can be created by any combination of manual coding, GUI created SVG data that is manually coded for animation, or using a GUI design and animation tool such as Adobe Animate, Google Web Designer or Tumult Hype (macOS only). Some manual coding is used to adapt the inherent animation facilities to support triggering by CasparCG commands.

Video and audio clips are processed by the ***ffmpeg producer*** supporting a wide range of file wrappers and codecs such as ProRes codec in .mov wrapper, DNxHD in a .mov wrapper, DNxHD in .mxf wrapper, AVC Intra codec in .mxf wrapper. Audio WAV files from a CD with 44.1 kHz sampling are sample rate converted by ffmpeg to 48kHz for use in a video environment.

The ***image producer*** renders bitmaps from still images stored as PNG, BMP, TGA, TIFF or JPEG. It recognises embedded alpha channels, using the fill and alpha in the compositing operations executed in the mixer module. The image producer auto-scales stills to fit the target channel size, enabling an Ultra-1 resolution source image to be shown in a channel operating in HD 1080i.

The ***scroll producer*** is part of the image producer function. It uses images that have one dimension that matches a channel output dimension. If the matched dimension is image width, the scroll producer operates the image as a roller caption. If the matched dimension is image height, the scroll producer uses the image as a crawler caption.

The ***colour producer*** creates full-screen solid colour displays. The colour property is specified by both the red, green and blue amplitudes with an overall opacity (alpha) amplitude. Using the mixer blending facilities allows complex tinting operations of stills or moving video.

The ***Decklink producer*** supports the input of live SDI video via a Decklink card. This supports record operations or picture-in-picture operations.

The ***Bluefish producer*** supports input of SDI via a Bluefish444 video card.

There are multiple IP ***Stream producers***. Native Newtek NDI inputs are supported on server version 2.3_LTS. Several MPEG-based streams can be used as live inputs using the ffmpeg producer.

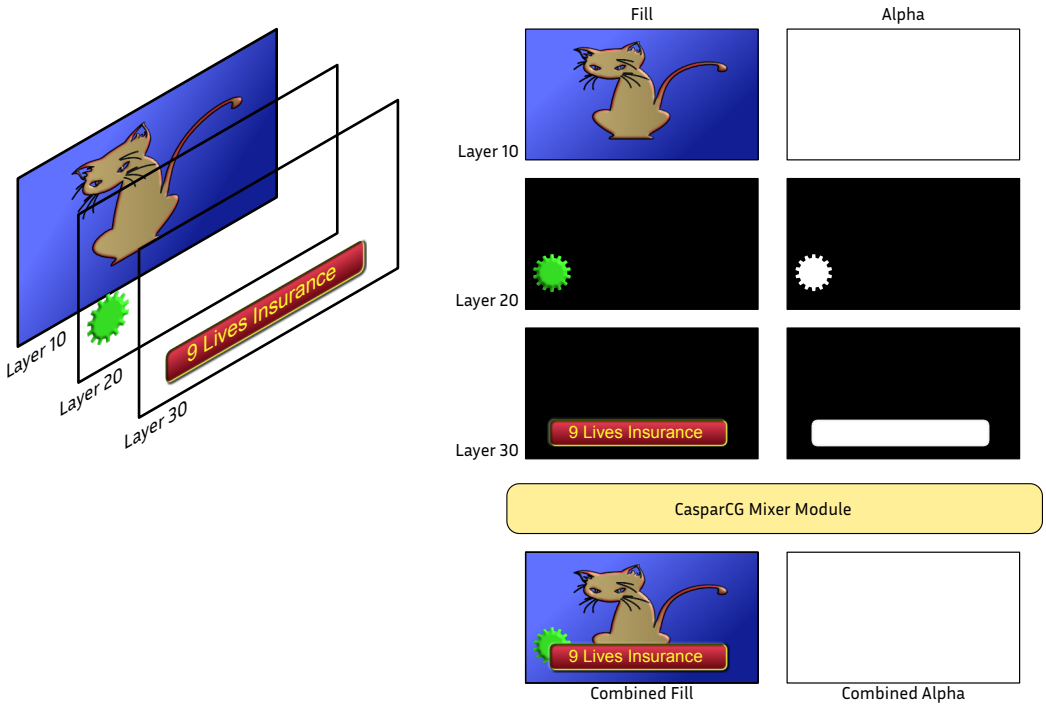
The ***Route producer*** sends either a selected layer or the composited channel output to a video layer in another channel.

CasparCG supports the use of Virtual Channels that have a defined spatial and temporal resolution, but that are not routed to a consumer. Virtual channels are used with the route producer to support various production requirements. Examples include sending a moving graphics background to multiple output channels ensuring temporally aligned outputs for downstream vision mixing; viewing a live input source on a designated channel; and sending an extra high resolution image to multiple output channels that spilt the image across the channel outputs to provide a video wall operation.

The producer's frame buffers pass content to the ***mixer*** module. The mixer implements various image manipulations including size and position, brightness, contrast, opacity and saturation. The transforms are applied independently to each active layer of the channel. The manipulated layers are combined to create the composite output. The mixer module implements the combination using both alpha data for a layer and user-set blend modes (multiply, darken etc).

The mixer module includes an audio mixer enabling complex mixing processes for the various layers of a composited output.

Layers are numbered from 0 to 9999. Layer 0 is at the lowest level of the combining stack and 9999 at the top. The layering process is illustrated below.



The left side shows a graphic made from 3 layers. Layer 10 contains the picture of the cat, layer 20 has the cog-wheel, and a layer 30 contains a strap with overlay text. The graphics in layers 20 and 30 contain large transparent areas defined by their alpha channels. The fill and alpha contents for each layer and the resulting output are shown at the right-hand side of the illustration.

Although this example uses static elements, each layer can be a dynamic element, such as video, with an associated alpha channel. The alpha channel controls both overall visibility and transparency of elements in its application layer.

The outputs of the mixer are converted to channel outputs by ***consumers***.

The ***screen*** consumer creates a video display on the computer graphic display of the server computer. The display can be created in a moveable window, or set borderless full-screen on a defined output screen. This style of consumer can be used to drive a display embedded in a piece of scenery.

The ***system audio*** consumer directs the channel audio to the audio driver in the computer. This is convenient for development servers, but less use in a production installation. An operational hazard is system alert and alarm noises are also sent to the same audio output.

The ***Bluefish*** consumer is associated with a Bluefish444 SDI output card. There may be two outputs for a channel, one with the fill video and the second with the key enabling a lower-third caption to be keyed in a downstream vision mixer. A single card can provide the output for multiple channels.

The ***Decklink*** consumer is associated with a Blackmagic Design Decklink SDI board. There can be multiple SDI outputs for a single channel producing fill and key signals. Decklink cards commonly offer an HDMI output for direct connection to a video display. A single Decklink card can provide multiple channels of SDI output, or the card may be operated in split mode with some BNC connectors operating in output mode, and other BNC connectors used for inputs.

The ***file*** consumer sends the channel output to a disk file. The resulting file can be used in an edit session, or become the master recording for the studio.

The ***stream*** consumer supports output of a channel in an MPEG transport stream encapsulated in a UDP transport layer. This option must be used with care as MPEG coding, especially H264, is processor intensive.

The ***NDI*** consumer allows CasparCG to output a Newtek NDI stream to other equipment. The data rate for such an output is quite high, perhaps 150 Mbit/sec per output channel. Using the NDI stream output may require extra TCP/IP interfaces to support this operation.

Multiple consumers can be assigned to a single channel. For example, using a file consumer and a Decklink consumer allows the channel to be viewed whilst a recording is created. The consumers can be added and removed by commands from the CasparCG client.

A CasparCG server computer may be fitted with multiple physical outputs. The server must be informed which card output is used for a given CasparCG channel. The resource allocation uses a reserved file, **casparcg.config**, stored in the same folder as the server software. This file has sets of XML tags that define the locations for the media, the allocation of hardware to channels, the channel resolution, as well as the TCP/IP ports used for various operations.

```
<channels>
  <channel>
    <video-mode>1080i5000</video-mode>
    <consumers>
      <decklink>
         <device>1</device>
         <key-device>2</key-device>
         <embedded-audio>true</embedded-audio>
         <latency>default</latency>
         <keyer>external</keyer>
         <key-only>false</key-only>
         <buffer-depth>3</buffer-depth>
      </decklink>
    </consumers>
  </channel>
</channels>
```

A segment of casparcg.config for a single channel is shown in the box to the left. The section between the `<channel>` and `</channel>` tags defines the properties of the channel. This tag pair are the first entry inside the overall channel information container (<channels> </channels>) making this the channel definition for CasparCG channel 1.

The video mode tags define the resolution for the channel, in this example high-definition operating as 1080i/25. The number after the interlace definition letter uses four digits so that the NTSC related scan standards can be defined using an integer parameter.

The `consumers` tags surround the specification of all the consumers used for the channel. In this example a Decklink card is assigned to the channel using card number 1 for the fill and card number 2 for the key. The card numbers are enumerated in the Blackmagic Design Decklink configuration tool.

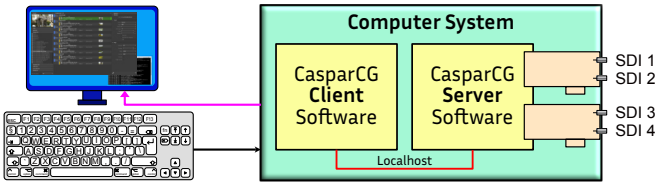
A ***CasparCG Client*** is any computer programme that sends ***AMCP control commands*** to one or more CasparCG servers. The server returns a status report to the client indicating success or failure of the command, or providing status information requested by the command. The server also outputs a stream of status data wrapped into Open Sound Control (OSC) messages and transmitted over UDP.

SVT provide an 'official' or 'standard' client for three operating system hosts - Windows, macOS, and Linux. Users can also develop their own control solutions known as ***custom clients***.

The SVT client uses the name ***rundown*** for a sequenced list of CasparCG commands. Rundowns can be stored on the client computer or in a shared network folder. Multiple rundowns can be loaded in a client, switching the active rundown is achieved by a single mouse click.

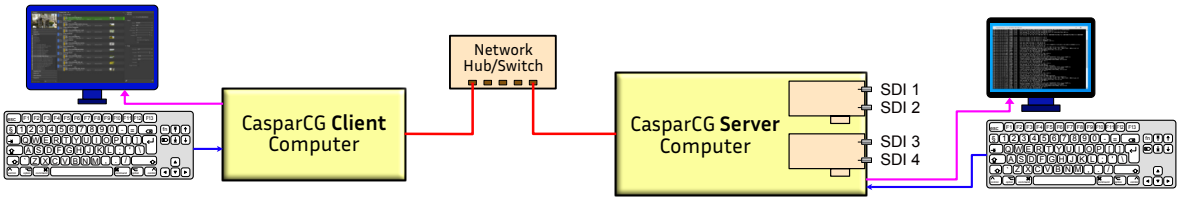
The operator steps through the elements in the rundown, issuing each command by selecting it in the list then pressing a function key such as F2 to start a playout operation, or F1 to stop the playout.

A single client instance can connect to multiple servers, and multiple client instances can connect to a single server. The combination of connections is established as needed to deliver the operational flexibility required.

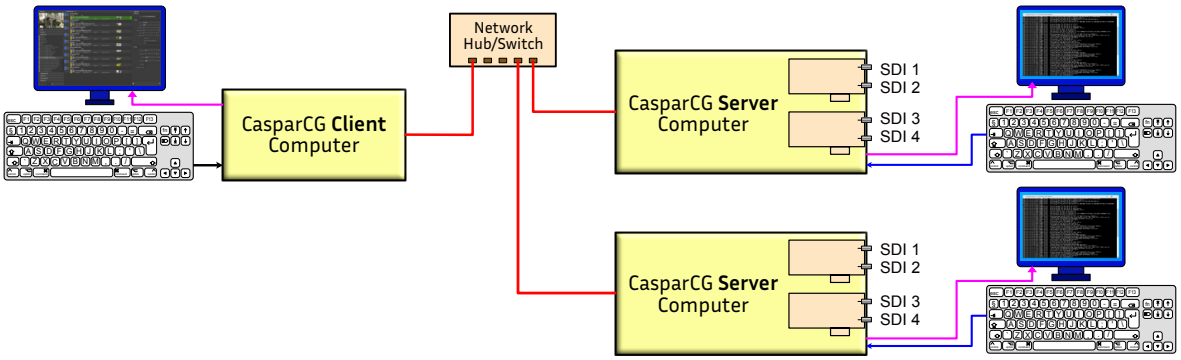


The simplest system uses a single computer running both the sever and the client as shown at the left. The control connection between the two software blocks uses localhost (127.0.0.1) networking.

A functionally equivalent operation can be implemented with the server software one one computer and the client software on a second machine as shown below.

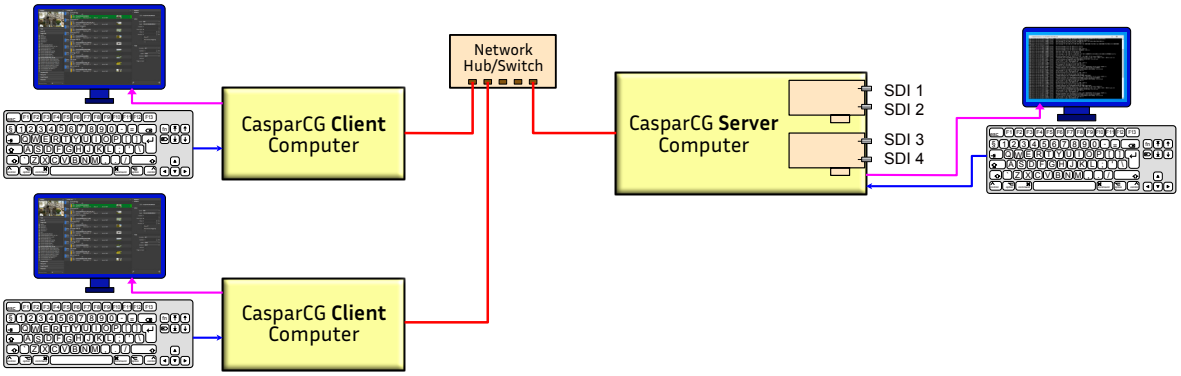


A single client connected to multiple servers is shown below. Using multiple servers can provide more output channels than a single computer, or servers can be paired to provide a main and reserve output.



Where servers operate as main and reserve users must make suitable arrangements to ensure the media is mirrored to both servers.

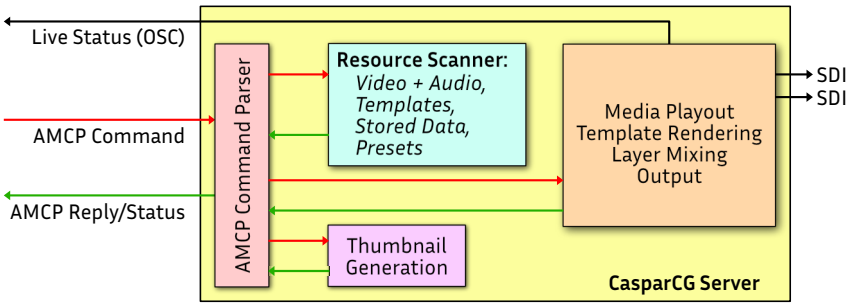
Multiple clients can connect to a single server. Supporting multiple connections to a single server allows one operator to manage graphics displays or lower thirds whilst a second operator manages clip replays.



For some news and sports applications one client provides manual control of some channel outputs whilst a second, custom, client pulls data from a sports information server to update event statistics on a chosen channel output.

CasparCG Software Versions

CasparCG is open-source software. Multiple code branches exist because individuals or organisations have altered the code to provide the specific facilities they need.



The overall server function is implemented by several processes. One process parses commands from clients, invoking the relevant processing. One process scans the media paths, returning any lists required by the client. Another process creates and manages the storage of thumbnail images used in the SVT client preview window. The primary playout process emits status responses to the client in response to user commands. It also emits a stream of status data using OSC (Open Sound Control) messages wrapped in UDP packets.

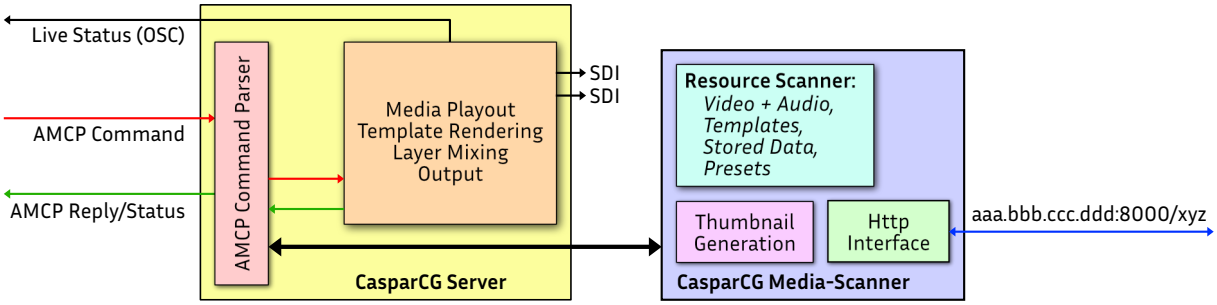
When this document was written there were two significant code branches. Key differences between the branches are the processing of the thumbnails, the management of media lists, and the detailed structure of the OSC messages.

SVT Code Branch

At the date this document was written the SVT software versions were:

server:	2.3.3_LTS (LTS = Long Term Support)
matched client:	2.2

This version requires a server computer graphics card with OpenGL version 4.5 or higher. SVT recommend *n*VIDIA Quadro graphics cards, although other graphics cards work.



The server software has been divided into two programs, one program provides the media scanning processes and thumbnail generation leaving the core server program to manage the playout processes. The core server uses the partner program to return lists of media requested by the client software. The media scanning element also supports user queries via a restful http interface. In the above digram the computer running the CasparCG server system has a tcp/ip address of aaa.bbb.ccc.ddd, and the media scanner program listens for traffic on port 8000. A limited set of http get requests are supported on this port including:

/tls	return a list the templates
/cls	list the media files
/cinf/<name>	return information about the named file.

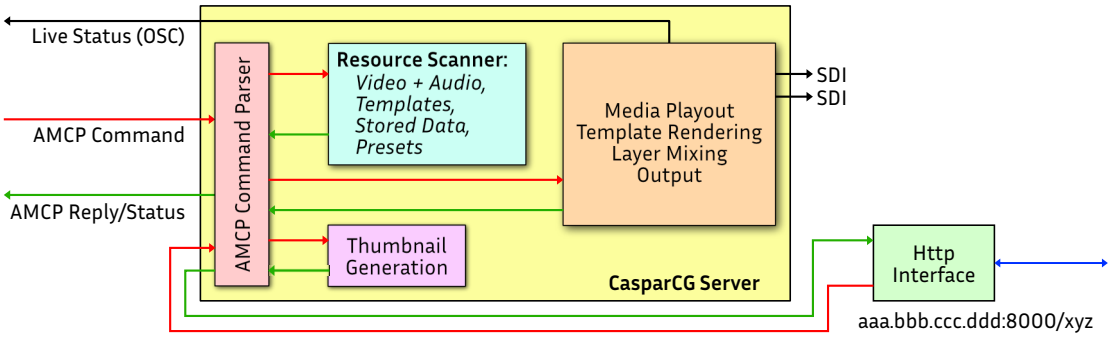
See this web page for a more detailed list of supported http endpoints.

NRK Code Branch

NRK created this code branch because they required time-based triggers that allow them to use CasparCG in their open-source *Sofie* news automation playout system. At the time of writing the current software versions were:

server:	2.1.12_NRK
matched client:	2.0.9

This version requires a server computer graphics card with OpenGL version 3.0 or higher, again *n*VIDIA Quadro cards recommended. There is a single executable that provides the media scanning and server functions. A media scanner program is available that hooks into the server and provides a restful http interface for media scanning. The process flow is illustrated below.



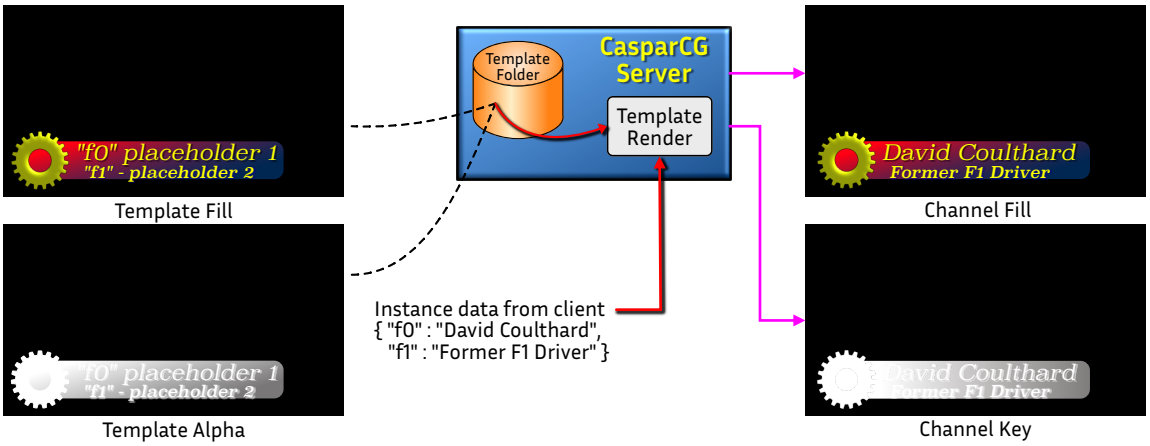
The NRK scanner module offers more http get endpoints than the LTS version of the server, mostly added in support of automation operations. The list of endpoints is available here.

Templated Graphics

A ***graphics template*** defines a preset format for the presentation of graphic and text information. A template is converted into a rendered bitmap by combining the template format with ***instance*** data. The instance data replaces named placeholder fields within the template.

The rendered results can be designed as static or dynamic. Static templates are commonly used when an operator pushes a button on a vision mixer to control the times when graphics inserts occur. Movements in a dynamic graphic draw the viewers attention to the updates or new information. Both static and dynamic templates require a scripting language that manages the insertion of new or updated instance data into the placeholders within the template.

Early versions of CasparCG server used Adobe Flash with ActionScript (AS3) as the scripting language. Subsequently HTML template support was added using an embedded Chromium engine that uses Javascript as the scripting language.



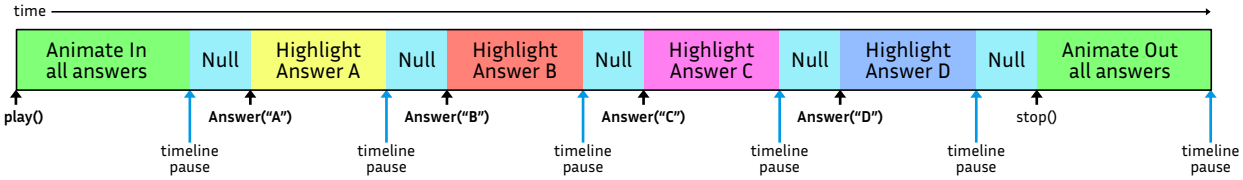
The placeholder fields in the above example are called **f0** and **f1**. Instance data is sent by the CasparCG client as part of a ***CG Play*** or ***CG Update*** command. Instance data is encapsulated in either an XML or JSON wrapper. In the above example the instance data is in a stringified JSON wrapper. A well-designed template auto-detects the encapsulation method and extracts the included instance data.

The template producer creates two bitmap outputs - fill and key. The template fill and key are used by the CasparCG internal mixer as it combines the template layer with other active layers in the channel, such as a video clip or more template layers. A composited fill and composited alpha/key are created for the channel. The two signals can be output as SDI signals via a Decklink or Bluefish consumer, enabling a keyer in a vision mixer to combine the graphics with live sources such as a camera output.

Each HTML template requires user provided Javascript functions to process the commands and instance data from the client. Five Javascript functions are required:

Function Name	Action
play()	Called when a CG PLAY message is received from a client. The play function initiates the display of the template content. The action may be a cut or a timeline transition.
stop()	Called when a CG STOP message is received from a client. The stop function initiates the removal of the template content from the display.
update()	Called when either a CG PLAY or CG UPDATE are received from a client. When a CG PLAY command is received the update() function is called before the play() function. The CG UPDATE allows new instance data to be inserted whilst the static graphic content remains on display.
next()	Called when a CG NEXT command is received from a client. Moves the timeline playout point from the current position to the next keyframe position. The CG Invoke command allows a user-defined function to be called by the client. For example the movement speed of a graphic could be set via a a function named **rollspeed(new_speed)** where new_speed is a used provided value. The name of the function to call is passed as a parameter of the CG INVOKE command. Multiple invoked functions can be used in a template.

An example of the invoke function operational use is a quiz show that displays a list of four possible answers. When the contest announces their choice, the invoke function is used to highlight the correct answer. An example timeline block for the quiz example is shown below. The null blocks are just spacing that permits later adjustment of animation durations.



When the operator presses the CasparCG Play key (F2) the server process first calls the update() function to place the answers in their display elements. It then invokes the play() function to animate the question and answers onto display. At the end of the initial animate process the timeline auto-pauses playout.

When the contestant gives their answer, for example answer B, the CasparCG operator sends an Answer("B") invoke command to the server. The invoked Answer() function jumps the timeline playhead to the start of the Highlight Answer B segment and starts the animation, playing until it reaches the next pause mark.

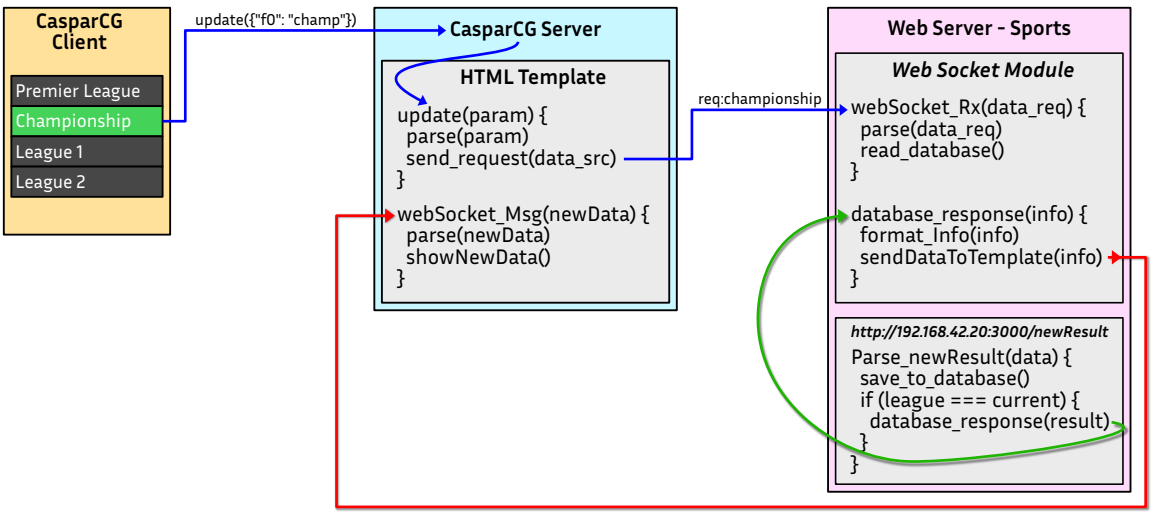
Finally, the operator sends a stop command to the server which invokes the Javascript stop() function. This animates the question and answers off the screen.

Template placeholder fields require identifying names. A news programme with a lower-third story ident may require two placeholder fields identified as ***Name*** and ***Designation***. When an operator instructs the SVT CasparCG client to add a identifier field name to the instance edit form it auto-inserts placeholder field names ***f0***, ***f1***, ***f2*** etc. These default field names can be over-typed to the user specified placeholder names. Many templates use the default idents to speed entry of instance data into the client entry form. The required field names need documenting so that the graphic operator knows the field names to enter.

There are multiple methods available to design and code a template. These methods include manual coding of the HTML page, free-to-use visual layout tools such as ***Google Web Designer***, and commercial design tools such as ***Tumult Hype*** (macOS only) and ***Adobe Animate***.

The visual editing packages support insertion of user-defined Javascript libraries or code blocks such as the CasparCG interface functions. Detail of how a template is authored is outside the scope of this document.

One feature of the Chromium HTML engine is built-in support for *WebSocket* technology. WebSockets allow the template to directly communicate with a web server. The data channel can be used both to request data from the web server and to have data pushed from the web server. An example data flow is illustrated below.

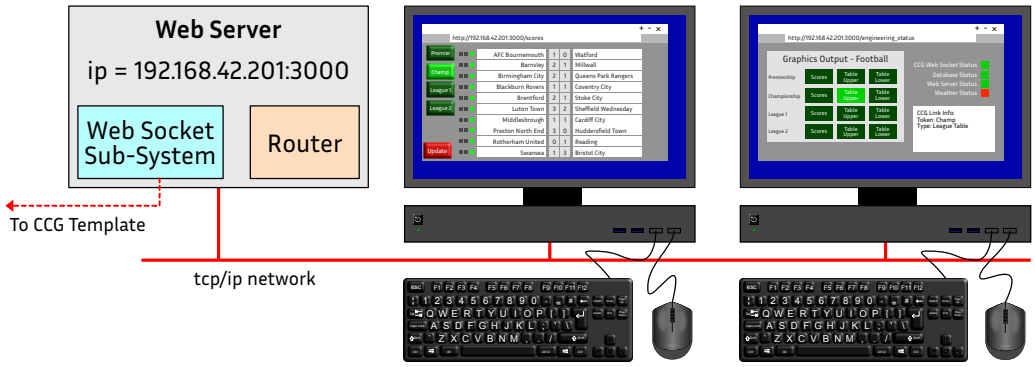


The illustration shows how a template can receive a request from a client, using the web socket system to request a large data set from a web server. This example fetches the football scores from the web server.

The CasparCG client issues a standard update command requesting a template data refresh. In this operation the instance data passed to the client is used as an index to request the actual data from the web server.

In the example, the parameter passed from the client has a field name of f0 and a value of "champ" (shorthand for championship). The CasparCG software receives the update command and passes the name and value to the update function. The update function uses the value to create a request for the actual data, sending the request to the web server via the web socket connection.

The web server reacts to the message initiating a request to fetch data from the results database. A database read operation is normally programmed an asynchronous process. When data is returned from the database the processing code calls function database_response(info). This callback function formats the information ready for the template to use, using the web socket connection to send the data to the template. The template updates the placeholder fields with the new results data.



The web server includes a conventional html page server. A computer browser can connect to that page server to allow manual entry of data, for example by accessing an html page at ***http://192.168.42.201:3000/scores***. The template and the web server can be programmed such that an updated result is pushed to the template for immediate display. The web server can also connect to other data sources via database queries or http *get* requests.

Other standard browser pages, for example ***http://192.168.42.201:3000/engineering_status***, can be configured to monitor the state of the template and web socket connection.

Some programmes, for example News, need a wide range of templates providing a consistent look and feel to all forms of name straps, story straps, live location identifiers etc. Rather than use lots of individual templates it is possible to use the power of the Javascript system to have a single template file that shows the element or elements required. Before the programme starts the template is played, providing an empty output.

All templated graphics are then controlled by the newsroom automation system. It sends a single command to the update process in the template, but that update can have several sub-fields separated by a printable character or character sequence that is never required in the graphic text. The template manages the displays such that if a two-line lower third is active and a single line display is needed the template does the remove old, display new transitions. Assume the field split id is set to use two up-arrow characters **^^**. The newsroom computer can request a two-line strap sending a command such as:

```
cg update 1-10 {"f0":"story^^Boris Johnson^^Prime Minister"}
```

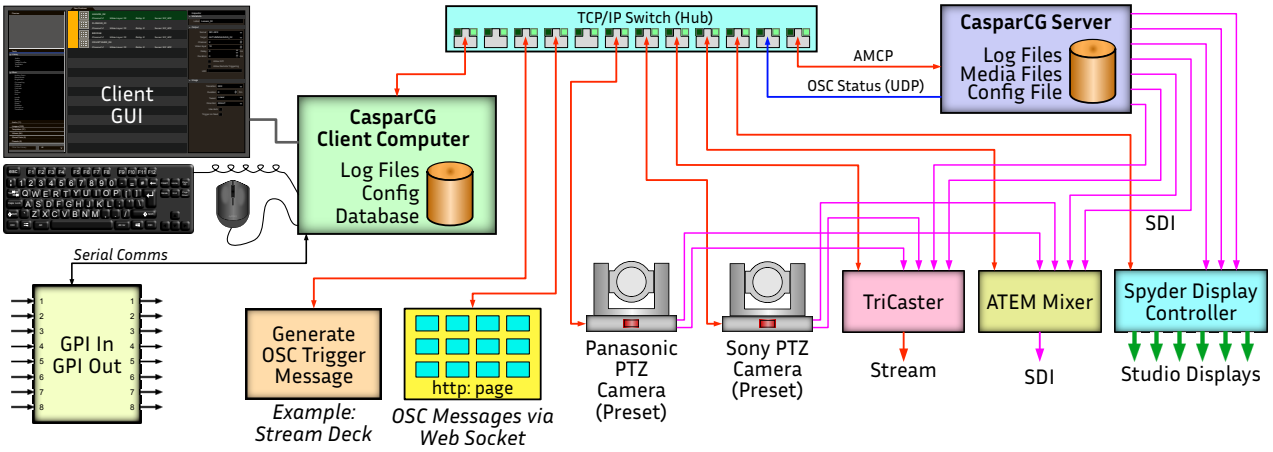
The Javascript update function checks the field id is "f0" then splits the value into sub-strings using **^^** as the divider. The key requirement for such control is a careful definition of the protocol that is used to select display elements and how the whole display or a sub-set can be cleared (for example removing a name strap but keeping the live location text).

The control operation can be extended to use web socket connections between a controller and the template. When the template is initialised from a CasparCG client it connects to another process that also has web socket support. Web socket libraries are available for many programming languages facilitating custom control solution development.

Such a custom solution can enable a weather presenter to use a hardware push button to sequence through a selected set of displays, or a presenter to use a browser on a tablet computer to summon a wide range of graphics.

CasparCG SVT Client

The SVT client enables an operator to create and edit rundowns, and manage their replay. The client can control multiple CasparCG servers, but can also interact with other items of broadcast equipment. This control connectivity is illustrated below.



The client uses a multi-column graphical user interface (GUI) where the centre column shows the list of items in the rundown. Playing items uses the keyboard function keys and cursor movement keys. It is operational best practice to set the client GUI to run at maximum screen size, ensuring that the client screen retains input focus for keyboard events.

The client software uses an SQLite database to store the configuration properties, such as the tcp/ip addresses of controlled items, and the thumbnail pictures of the video and stills media on connected CasparCG server units.

The client generates plain text logs of the actions executed by the client, each entry is time stamped using the host PC clock time.

Controllable Hardware

CasparCG Server

The client establishes and maintains links to the CasparCG servers listed in the configuration. The client continually monitors the link status and shows error indications on the client GUI if communications are not operating.

The client requests the list of media and template items stored on each linked server. Updates of the media lists can be done using manual requests or set to run at user-defined intervals. The client requests thumbnail pictures of any new media elements.

The client sends AMCP control messages to connected servers. The server checks the command syntax returning brief status data about the command (OK, Fail etc), or sending more extensive data such as the list of media file names.

The server emits a UDP data stream describing the current operational status of all channels in the server. The status is encoded into OSC (Open Sound Control) address structures. The OSC messages are sent to all registered clients, plus external recipients defined in the server configuration. The client uses the OSC data to show media progress bars for video clip playout. External OSC status processors can create status displays, for example on a web page, enabling a programme director to see the names of current media and their timeline progress.

GPI Control

Any item in the rundown can have GPI triggering enabled. CasparCG clients use an RS232 serial connection to a GPI hardware interface, typically implemented using an Arduino microcontroller. The client interface supports eight input contacts that are allocated to command functions in the client configuration. The configuration sets the triggered action and active GPI transition edge for each input. The triggered action options include Stop, Play, Pause, Load, Next and Clear.

The rundown command set includes a GPI output instruction. This enables the client to trigger an external hardware unit such as a lighting desk or control an element on a vision mixer such as the DSK enable. The duration of the GPI output pulse is set in the client configuration. The GPI output data is sent via the serial link to the hardware interface.

OSC Control

When the client is installed or the configuration is reset Open Sound Control, OSC, message support is enabled. The client listens for OSC *control* messages, matching the address in the OSC message with the remote control enabled element address fields in the active rundown. An example control message from an external unit is:

```
/control/clip01/play 1
```

The user enters **clip01** into the UID address box of the rundown item to trigger. When the control message is received the CasparCG client will select the item with UID containing `clip01` address and issue the play command for that item.

This message address capability allows an external “Shot Box” interface to be used as the control device. Example hardware devices are the Elgato *Stream Deck* and the P.I. Engineering *X-Keys* units.

The client listens for OSC messages via multiple IP ports. One port, typically 6250, is for general server status messages. A second port, typically 3250, is used for control messages and a third port, typically 4250, is used for Web Socket transport. This Web Socket transport support enables a web page on a computer or tablet to send the control messages direct to the client.

Pan Tilt Zoom Cameras

The SVT client has limited control of Pan Tilt Zoom cameras from Sony and Panasonic. The manufacturers PTZ control software or hardware is used to create multiple presets in the camera memory, including the speed of movement from current positions to the preset values. The SVT client instructs a camera to recall a numbered preset.

TriCaster

NewTek make a range of Vision mixers called ***TriCaster*** which make extensive use of software control. The SVT client can interact with a TriCaster to select sources on the preview or transmit banks, select a network video source, play a macro, and trigger a transition on the background or DSK keyers. These control facilities enable keying of lower thirds to be managed by the CasparCG operator when the vision mixer operator is busy on other programme demands.

ATEM Mixer

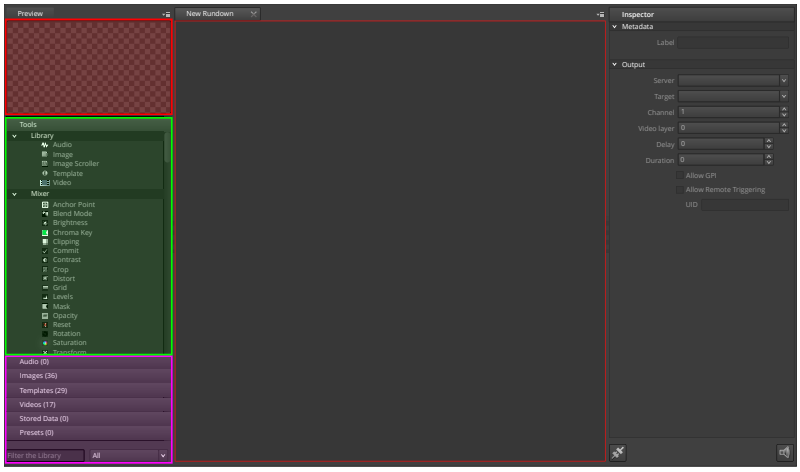
Blackmagic Design manufacture a range of ATEM vision mixers from simple desktop units to four mix/effect processors. The SVT client has controls for source selections, keyer action, playing macros and control audio properties such as gain and pan.

Spyder Videowall Processor

Christie Spyder units are processors that manage displays of pixel based graphics onto a linked set of displays forming a video wall. Some broadcasters use Spyder units to control the displays of graphic sources sent to video walls and displays in a news studio. Christie tools are used to create presets that the SVT client can recall.

Client GUI

The SVT standard client is available for three host operating systems - Windows, macOS and Linux. The client uses a graphics support library from *The QT Company* ensuring the same “look and feel” on all operating system hosts. When the client software opens, the user interface looks similar to the picture below.



The user interface has three information columns.

The leftmost column displays information about media and playback control tools. This document has added coloured highlight boxes to the picture to identify the segment functions within this column.

The red rectangle is the media preview display. The green rectangle shows the list of media and command tools. The magenta rectangle is the library listing selection and library search filters.

The centre column shows the rundown media and command list or lists. Rundowns can be saved or loaded from the ***client*** disc. Multiple rundown sets can be loaded in the client enabling swift selection of a set of commands by clicking the rundown name tab at the top of the column.

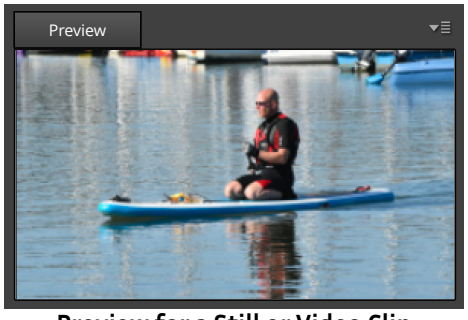
The rightmost column is the information inspector and editor. It displays item or group properties, and is used to edit property values such as the target output channel and video layer.

The left and right columns may be enlarged to expose more information edit space, or collapsed to increase the screen area available to the rundown column. The entire right side bar of the leftmost column, or left side bar of the rightmost column are dragged to alter the column width.

Hover over the column edge to be moved and the edge drag cursor appears (see icon at left).

There is a minimum column display width, ensuring all core information is visible. Continuing to drag towards the edge of the screen causes the column width to collapse into a hidden mode. Even when collapsed, there is a knurled patch visible at half the client window height. The looks like the closely spaced lines shown to the left of this text. Minimising the library and inspector columns during playout is good operational practice. This aids keeping focus on the rundown and hence keystrokes are directed to the rundown control processing.

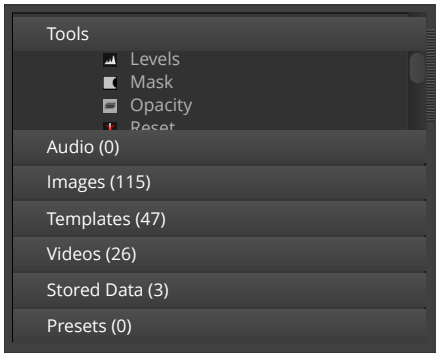
The Preview panel in the left-hand column is optional. It is strongly recommended this panel is configured as visible in Studio C operation. Display of the preview window is controlled by a selection box in the general tab of the client configuration. When a user selects a still or video in the one of the library tabs, the preview window shows either a small version of the still image, or the poster (first) frame of the video clip as illustrated below.



Preview for a Still or Video Clip

The server system creates the preview picture and sends it to the client where it is optionally stored in the client database.

There is a small triangle adjacent to four horizontal lines in the top-right corner of the preview window. Clicking the arrow reveals a drop-down menu enabling the user to show or hide the preview window, and to choose fill or alpha channel preview mode.



The **Tools** tab lists all commands and playout control tools. The example to the left shows some of the layer mixer tools. Tools are available to load video clips, load stills, load html pages, load templates, adjust the mixer properties such as opacity, and control external units such as a Tricaster or an ATEM vision mixer.

The **Audio** tab lists the filenames for any audio files (such as *sting.wav*) in the CasparCG server media store. The preferred audio sample rate is 48kHz, but CasparCG, on-the-fly, re-samples WAV audio that was sampled at 44.1 kHz enabling replay of audio extracted from an audio CD.

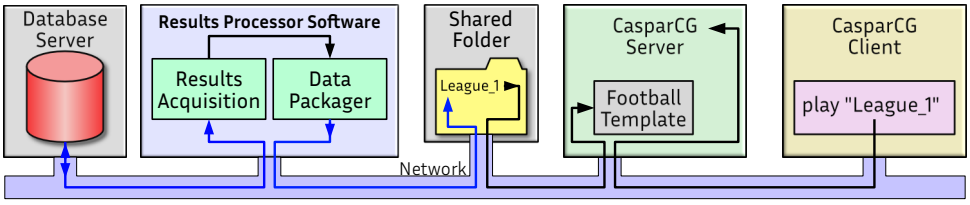
The **Images** tab lists filenames recognised as still pictures. The commonly used image file extensions are BMP, PNG, TGA, TIFF and JPEG.

The **Templates** tab shows the graphic template names of files stored in the server templates folder. CasparCG supports two mechanisms for dynamic displays - Adobe Flash and HTML. Flash technology is end-of-life and should ***not*** be enabled in the Studio C installation. HTML templates for CasparCG include several javascript functions called when a template is played or stopped, and when the text content requires updates.

The **Videos** tab shows files in the CasparCG server media folder that are recognised as video clips. The name display does *not* include the file extension. Any file recognised by ffmpeg can be played in CasparCG provided it uses standard broadcast sizes and audio sample rates. If the audio is not sampled at 48 kHz, clicks, pops, bangs or silence may occur.

CasparCG supports use of alpha (key) channels in channel compositing and output. If a video or stills file inherently contains an alpha channel it is used by the layer mixer, and the alpha content is available to output on an SDI connection to a vision mixer. CasparCG also supports the alpha channel presented as a second file. It recognises the association via a file name convention requiring **_a** added to the name section of the alpha file. For example if the fill video file is called *myTitles.mov* the alpha channel file must be called *myTitles_**a**.mov*.

The **Stored Data** tab lists any data sets stored on the server. A lower-third name strap combines the design layout (the template) and instance data (the name to display). The instance data is sent as part of the CG_*play* or CG_*update* command to the server. It is easy for the client operator to enter and check small amounts of text such as a name or a headline. The Stored Data mechanism supports graphics that need large amounts of dynamic information, such as a set of football match teams and scores.



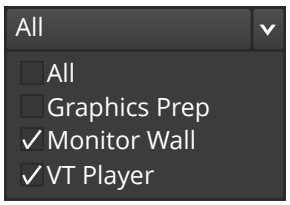
The mechanism is illustrated above. An external computer process gathers the data, often from a database, packaging it in the same fashion used for inline template data transfers. The packaged data is stored in a folder shared by the external computer and the CasparCG server. The client sends a template “play” command to the server, passing the external data set name to the server.

The **Presets** tab lists the names of user-saved command sets on the **client** computer. The presets feature supports recall of sets of AMCP commands, for example a headline sequence, for use in a rundown.

Instance data, such as the clip name, can be updated in the rundown, but the proven sequence of commands in the preset is re-used to minimise potential entry errors.

Below the library content tabs is a **library filter** box. The example at the left of this text shows the box before filter string entry. When a string is typed into the filter box and the return key pressed, the displays of files in the tabbed lists show only names that contain the filter text letters.

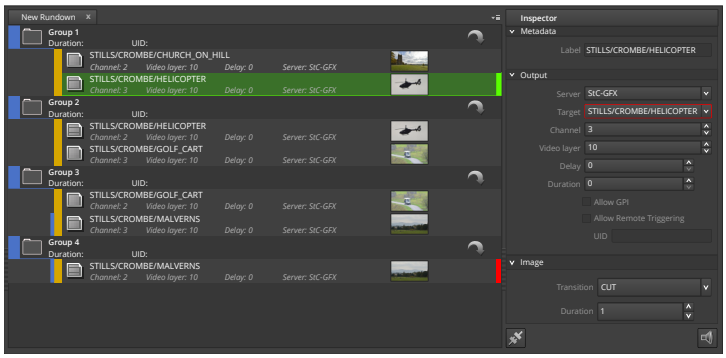
Assume there is a saved a set of files about rivers stored in a sub-folder called ***river***. The search text **river** will match that folder name, limiting the name lists to just that folder plus any other files containing the filter sequence. Case is ignored in the searches. To remove the filter either manually delete the string entry or click on the cross in the filter box, then press the return key to refresh the library listing.



To the right of the filter string entry box is a second filter control. A single client can connect to multiple server units. The example at the left shows a client with three named servers. Currently the server filtering is set to show the library content of both the Monitor Wall server and the VT Player server.

The central panel of the client screen shows the contents of the currently active rundown. A **Rundown** is a sequence of videos, stills, templates and control commands to play out under user control. Rundown lists can be stored on the *client computer* disc for later recall and/or modification. Multiple rundowns can be open in the client, enabling the user to quickly switch between sequences via the named tabs at the top of the rundown list area.

The **Inspector** panel at the right-hand side of the client window is used to view and change item properties. The inspector shows the properties of the currently selected item in the rundown.



The **Output** section of the inspector shows the target CasparCG server name, channel and layer. The server is selected from a drop-down list of server names.

Server names shown in the server list, and their associated IP addresses are defined in the client configuration dialog.

A stored rundown file includes the ***server name*** in the item data. Using identical server names in ***all*** client configuration files within an operational area allows the rundown file to be copied and run on a different client computer with no additional editing.

The **Target** box is the name of the media item in the server media folder. It is a drop down box that can display a media list. The library filter controls effect the content of that displayed list. The **Channel** and **Video layer** properties were described earlier in this document. **Delay** sets the time interval between pressing an action button and when the command is sent to the server. This delay feature is used when building timed sequences of events within a group.

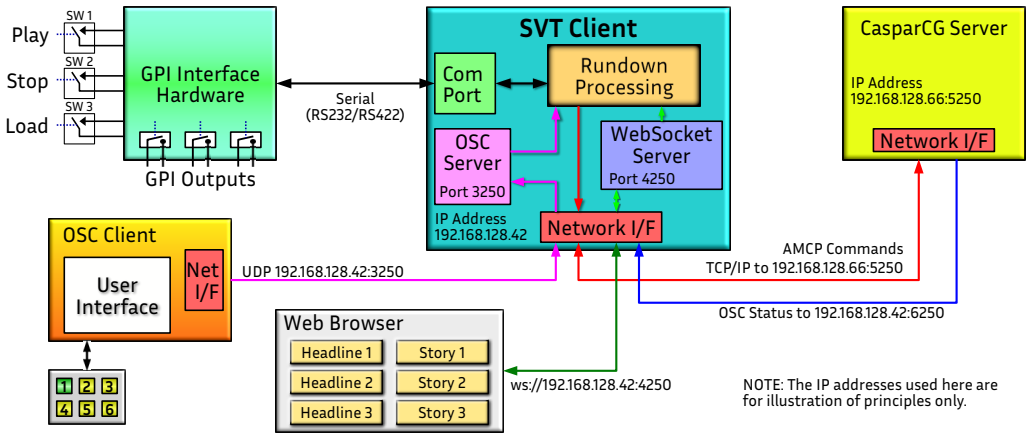
If the **Duration** is set to a non-zero value the media will be removed from display after the time set in the duration box. Delay and duration entries can be configured to show in either milliseconds or frames using the client configuration.

An event, such as play or stop, can be triggered using a keyboard function key on the client computer or by using an external control. The SVT client supports two external control mechanisms:

- Contact closure (GPI) - affects the current selected rundown item.

- OSC (Open Sound Control) messages. These can target a specific item for shot-box style operations, or remote control item selection and set the play/stop/pause state.

The external control options are illustrated in the diagram below.

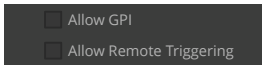


GPI contact closures are sensed in an external hardware unit such as a microcontroller. The contact close information is encoded into a short ASCII message sent across a serial connection to the SVT client. The client configuration defines the mapping between the GPI input number, the active transition edge, and the action initiated on the selected rundown item. The serial protocol is described in Appendix A.

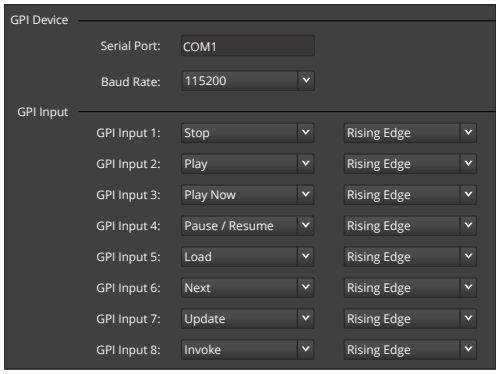
An external OSC client can send messages to an OSC server hosted by the client. The default sets the OSC server to listen for traffic on port 3250, but this can be changed by the client configuration. When a recognised message is received the message is passed to the Rundown Processing software for action. The OSC system and the CasparCG OSC message address schemes are described in Appendix B.

An Elgato Stream Deck LCD button unit combined with open-source BitFocus Companion software can be used as an OSC client. This supports both shot-box operations and sequential playout via the rundown.

Modern web-browsers support ***WebSockets***, a technology that provides a full-duplex high speed communication between client and server. The SVT client includes a WebSocket Server enabling a browser and html page to connect to send OSC messages to the client via the WebSocket connection. The OSC operation via websockets is described in Appendix B.



Rundown items that are intended to receive GPI triggers must be enabled by ticking an Allow GPI selector box in the Item Inspector.



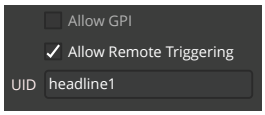
The default settings for both the GPI device and input actions are shown in the screen grab to the left.

The GPI configuration defines the active level transition and the action triggered by that transition. This allocation is universal on the client computer.

A simple microcontroller, such as an Arduino, can be used to convert physical button presses into the serial bit stream sent to the client computer.

There is a rundown command that triggers a GPI output bit. This provides a simple control trigger to external equipment such as a lighting chaser controller used on a quiz programme set, or actioning something on the vision mixer such as enabling the DSK.

The output GPI is connected via the same serial port used for GPI input triggering. The duration of the GPI output bit is configured in the client configuration, GPI tab.



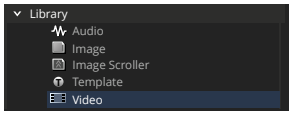
OSC control of an item is enabled by a tick in the Allow Remote Triggering box in the inspector. In the UID box enter the control name that the external OSC client will send. The software that sends the OSC command includes the chosen trigger name (headline1 in this example) as part of the control message:

```
/control/headline1/play 1
```

The 1 in the example command is an integer parameter. For many day-to-day operations it may be advisable to use very simple UID's such as a number. External trigger units do not then need reprogramming as different rundowns are selected for replay, just allocate the trigger ID in the SVT client item properties.

Adding Media and Commands to the Active Rundown

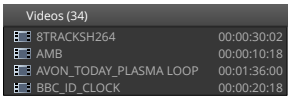
There are two methods to add media or other control commands elements into the rundown.



Method 1

Insert the base command from the Tools tab. The screen grab at the left shows the list of library media selection tools.

Either drag and drop the item/tool into the rundown, or double-click the item/tool. Use the Target field in the item inspector to choose the media item. Set the target channel and target layer.



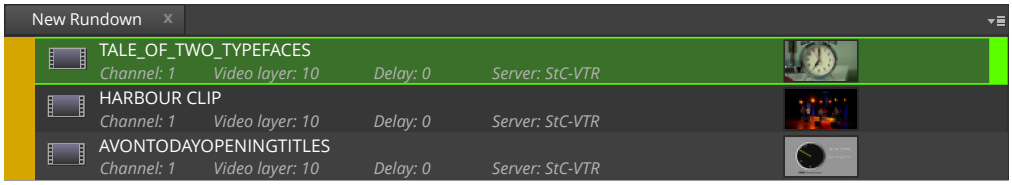
Method 2

Use the Library browser to locate the media item, checking the preview display to confirm the media choice.

Either drag and drop the item/tool into the rundown, or double-click the item/tool. Use the Target field in the item inspector to set the target channel and layer.

If items have not inserted at the required position in the rundown, their position can be moved by either mouse click and drag, or by key strokes (**Ctrl**+**Cursor Up Arrow** or **Ctrl**+**Cursor Down Arrow**).

Selecting a rundown item from the library automatically fills in the property inspector's **Target** field. The screen grab below shows a rundown after three items have been added. The topmost item 'TALE_OF_TWO_TYPEFACES' has been clicked to make it the active selection, showing it's properties in the inspector.

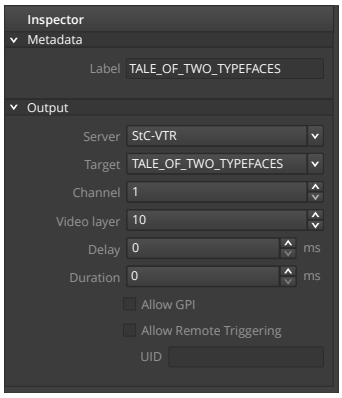


Summary properties are available in each item in the rundown list. As fields are edited in the inspector, information changes are reflected in the item rundown display. When a library double-click or drag and drop is used to add the item to the rundown the label display in the top line of the rundown entry is set to the clip name. When a raw tool is inserted to the rundown the tool type is shown in the label (Video, Image, Template etc). Use the Inspector Label field to change to an operationally more relevant name.

If the properties are not shown in the items of the rundown, the client is operating in *Compact View* as show below.

The viewing mode is toggled by the Toggle Compact View entry in the Rundown menu section of the client application menu.

The Item Inspector for a rundown item shows the file name, the allocated output channel, the video layer, the delay between pressing the command button and the command being issued, and the server that will receive the command.



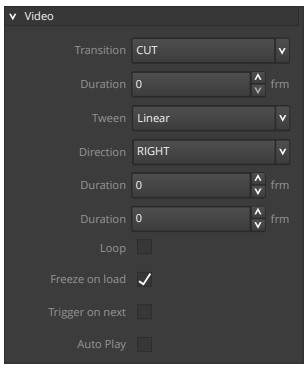
The Inspector display for the Tale_of_Two_Typefaces clip is shown at the left. The **Label** field is set as the clip name because the item was dragged from the library into the rundown. This field can be edited as may be required, for example providing an operator hint such as USE F6 (UPDATE) KEY.

The **Server** field is a drop down box. Select the target server from the list. The **Target** field is the item file name. For clips and images this is the file name to play, for templates it is the template file name to use.

The **Channel** and **Video layer** define which output channel and layer will carry the media. Video and stills default to Channel 1 Layer 10, templates default to channel 1 layer 20.

The **Delay** value adjusts the time offset between the play command button being triggered, and when the command is sent to the server. When an individual item is played the delay field is normally set to 0. The client can group commands together (more details later) enabling an action such as Play to be sent to multiple items using a single keystroke.

An example of using the delay property is playing a clip and overlaying a lower-third caption credit. The clip item and the caption item are grouped, playing the clip in layer 10 and playing the caption in layer 20 of the chosen output channel. The video clip Delay is set to 0, and lower-third Delay is set at 5000 ms. When the group Play command is activated, the clip will start immediately, but the caption display is delayed by five seconds. The caption layer can also be set to stop after a user-selected time (Duration field).



The item properties section of the inspector changes slightly with item type. The example shown at the left is the property display for a video clip.

The **Transition** field has a drop-down list of the available transitions, including Cut, Mix, Push, Slide and Wipe. The **Duration** of the transition is set in ***frames***, not milliseconds. **NOTE**: with server version 2.3 LTS a fade duration of 0.5 seconds is entered as a value of 25 (for 1080i50 channel).

The **Tween** field controls the transition profile. This is a very extensive list of options, but the most used are linear, EaseInOutQuad and EaseInOutSine, the latter giving the flattened S shaped profile commonly used in vision mixer T-bar transitions.

The **Seek** field allows the clip to be loaded with the chosen frame displayed as the first frame in the clip. A non-zero value in this field is commonly used to skip over a clock or ident at the start of a clip. The **Length** controls the number of frames that are played before the clip shows a freeze frame or re-loads (loop mode play). Server version 2.3 LTS requires numbers are entered as the number of *frames* for a progressive mode output channel, or as the number of *fields* for an interlaced mode output channel. Thus a seek of 1 second is entered as 50 on a 1080i/25 channel.

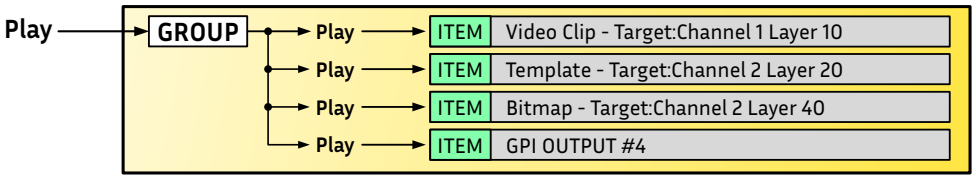
The **Freeze on load** tick box default setting is defined in the client configuration. When enabled, a video loaded with the F3 key will show the initial frame of the video. If the freeze on load is not active a blank frame is shown, with the video loaded in the background ready for play. Having the frozen still is helpful to many directors as they can see the clip is the one they want to play next.

The **Trigger on next** tick box changes the function key used to initiate play from the F2 key to F5 key.

The item **Auto play** tick box is disabled, unless the item is part of a group that has the group Auto play enabled.

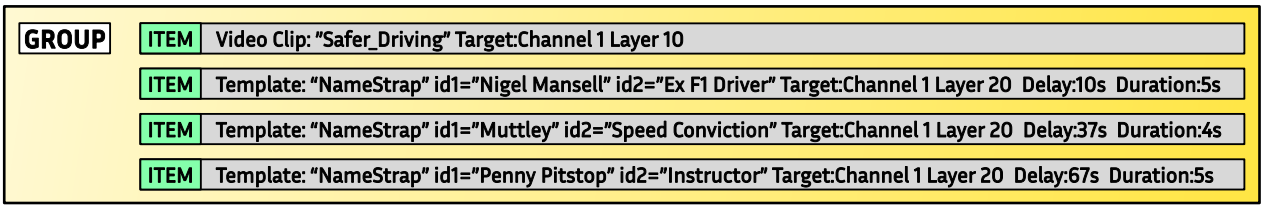
Groups

The group tool, as it's name implies, allows multiple commands to be treated as if there was just a single entity. A CasparCG command, such as play [F2], directed at the group container is sent to all members of that group, as illustrated below.



The group-level command is sent sequentially to group items, with minimum delay between each command. Layer mixer commands have a **defer** tag. Commands with the defer flag asserted are stacked in the server until an **Execute Deferred** command is received. All deferred commands then start operation in the same blanking interval.

Combing a group container with delays set on the component enables sequenced playout of items or actions. Consider a clip replay that needs some lower-third credits shown during playout. An example of such a group command is illustrated below.

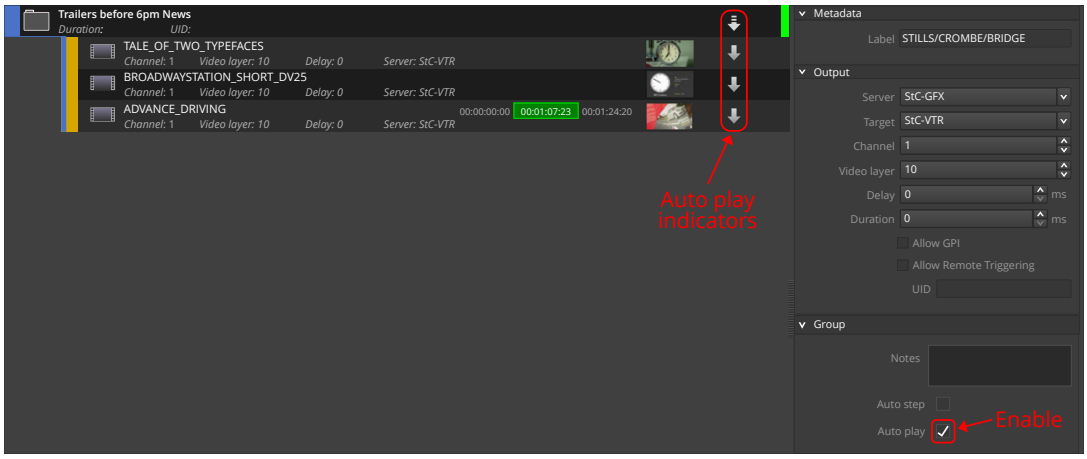


The video clip, item 1, starts playing as soon as the group play command is issued. The client waits 10 seconds before it issues a play command to item 2, a template called NameStrap that displays two lines of text. The text to display is sent as part of the play command. Five seconds later the client sends a remove (Stop) command to the same template. The same play and stop process is applied to items 3 and 4 at their specified offset times from the start of the clip.

The items in a group can be audio-visual commands, or device control triggers sent to external devices. The CasparCG client can issue GPI output commands, send an OSC command string, instruct a Panasonic or Sony Pan-Tilt-Zoom cameras to recall a numbered preset, and issue commands to ATEM and TriCaster vision mixing devices.

Group Auto Play

A group can also be set to auto run a series of video clips using **Auto play**. An example of an auto-play enabled group is shown below.



All video clips are allocated to the same server and channel. The videos are added to the group, then Auto play mode is enabled for the group via the tick box in the Group Inspector. This sends the Auto play enabled flag to the members of the group.

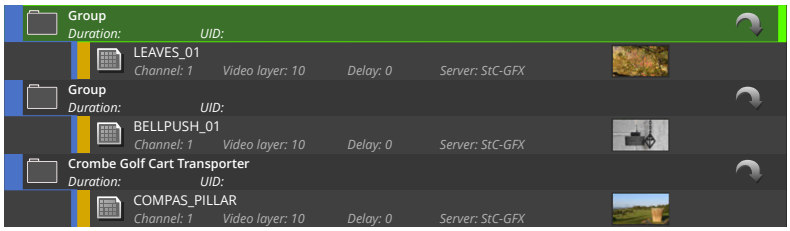
The SVT rundown client display shows Auto play mode is active by adding a large downward pointing arrow to the group box and to each item in the group.

When the group is selected and a Play action initiated, the client loads the first video in the list. The second video in the list is loaded in the background section of the channel. When the currently playing video reaches the end of file or reaches the set length, the background clip auto transitions to become the foreground clip. The transition settings on the background clip define the transition - Cut, Fade Wipe etc.

The client loads the next clip in the group list into the background section of the channel, and the auto play operations continues to the last clip in the group.

Group Auto Step

The Group facility also has an **Auto step** mode. The group is selected by mouse click or keyboard. When an action command, such as play, is issued on the group the command is executed by all elements in the group, then the next item in the rundown is selected. An example of the client with 3 groups set for Auto step is shown below.



The auto step enable is indicated by the looping arrow in the group header.

This mode allows the operator to step through a sequence of display commands without manually selecting the next item in the play list.

Auto step is commonly used in Studio C operation to advance through the sequence of stills used during the Avon Today studio exercise.

Creating a Group

Add one or more items to the rundown, either by double clicking or dragging a tool or media item. Select the items that are to be in the group using the mouse click and standard selection modifier keys (Shift and Alt).

Method 1

With the mouse above one of the selected items use a right click to show the context menu, selecting Group.

Method 2

Use **Ctrl**+**G** key combination.

The client creates a Group identifier (folder icon at the left) and shows the group members by adding a left-side indent to each item. It is best practice to name the group immediately after it has been created. Select the group header, then use the inspector to set the **Metadata Label** to a meaningful name.

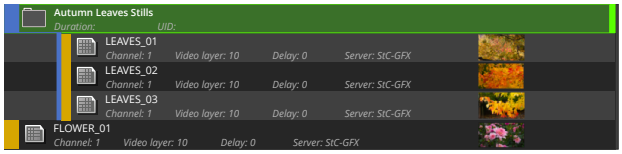
Deleting a Group

To delete the group and all items in the group, select the group then either use the Delete Key or right-click the group and select Remove from the context menu.

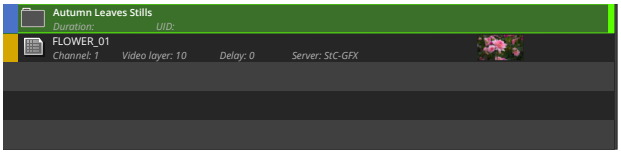
To delete *just* the group identifier and keep the items in the rundown, select the group then either use **Ctrl**+**U** key combination or right-click and select **Ungroup** from the context menu.

Expanding and Collapsing a Group Display

A group display can be set to show the group header with all the grouped items, or show just the group header. These display options are illustrated below with a rundown containing a group with 3 items followed by a stand-alone item. On the left we see all of the group contents, on the right we see the group collapsed to make maximum screen display space.



Group Display - Items Revealed



Group Display - Items Hidden

The expand and collapse operations are actioned using either the mouse or a keystroke. A mouse double-click over the group header *toggles* the expand status. With the group selected, **Cursor Left Arrow** collapses the display, **Cursor Right Arrow** expands the display.

Editing a Group

Add Item to Group

Method 1 - Using a mouse double click

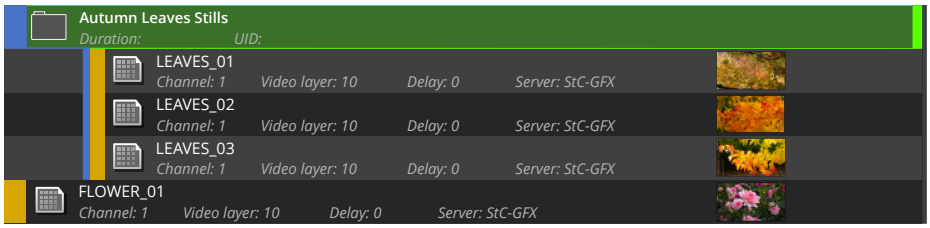
Expand the group if it is currently shown collapsed. Use the mouse to select an item in the group. This sets added content to insert immediately below the selected item. Locate the item to add by browsing the tool or library list. Double click the item to add it to the group. Repeat the process if more items are to be added.

Method 2 - Using Drag and Drop

Expand the group if it is currently shown collapsed. Locate the item to add by browsing the tool or library list. Click and drag the new item over the current group item display. The new item is inserted immediately *after* the group element under the mouse pointer when the left mouse button is released.

Method 3 - Add the item Immediately below the Group

Select the item to add, such as FLOWER_01 in the example below, then use **Ctrl**+**Cursor Right Arrow** to add the item to the group.



Post-Addition Editing

The newly added item is auto-selected by the insert process, showing the item properties in the inspector. Set the item properties such as the channel, layer and transition mode.

Remove Item from Group

Method 1

Select the item using the mouse, drag the item above below the group and release the mouse button.

Method 2

Select the item with the mouse cursor, right click to display the context menu and select Ungroup. The item moves to the line below the group.

Method 3

Select the item then use **Ctrl**+**Cursor Left Arrow**. The item moves to the rundown line below the group.

Moving items within the Rundown or a Group

Items can be moved to a new position using either the mouse or the keyboard.

Mouse: Select the item holding the left mouse button whilst dragging the item to a new position. When the mouse button is released the item moves to the slot below the item under of the cursor pointer.

Keyboard: Select the item using mouse or cursor movement keys. Then use **Ctrl**+**Cursor Up Arrow** to move the item upwards, or **Ctrl**+**Cursor Down Arrow** to move the item down.

After any item is selected in the rundown the selection highlight can be moved up or down the rundown or group using the **Cursor Up Arrow** or **Cursor Down Arrow** keys.

Presets

Some programmes have complex video and graphics replay sequences that are used on almost every episode. Entering the media items into the rundown and editing their transition properties can be a relatively complex operation with all the testing needed. The ***Preset*** allows a proven item or block of items to be stored as a named entity. The preset is recalled for subsequent episodes, and the media names or template data fields edited to meet production requirements. But all of the transition properties, channel and layer allocations are retained from the recording of the preset.

Creating a Preset

Use the standard tools to create the list of media to playout, external triggers to be output, and instance data for templates. Prove the operations by rehearsing the sequence. Use the mouse to select the first item of the block that will be recorded as a preset. Move the mouse over the final item of the block, press the shift key then click the left mouse button to highlight the block.

Right-click the mouse over the block, selecting **Save as Preset...** from the context menu. Enter a name in the pop-up box and click the OK button.

Using a Preset

Highlight an item in the rundown. The recalled preset actions will add to the rundown immediately following the highlighted item.

Open the presets tab in the library window column. The filtered list of preset names is shown. If the needed preset name is not shown, check the filter settings.

Use your preferred method to copy media from the library list into the rundown - double click, drag and drop, or right-click and select **Add Item** from the context menu.

Select each item in the recalled preset and select new targets and update template data as required.

Deleting a Preset

Select the preset in the library. Either press the Delete key, or right-click and select Delete from the context menu. **Beware - there is no undo mechanism available**.

Archiving a Preset

Use the New Rundown item in the client file menu or use the Ctrl+N shortcut. The new rundown auto selects as the focus for new item entry. Recall the preset from the library.

Use the Save As... entry in the client file menu or the shortcut Shift Ctrl+S key combination. Select a folder then name the rundown and click Save. Local practice dictates the folder and rundown naming convention.

Copy a Preset to Another Client

Follow the process to archive the preset to copy. Save the copy on either a shared network folder to make the rundown file accessible on the second client. Alternately use a USB pen drive and “sneaker net” to move the file to the second client computer. Open the copied rundown on the target client computer, select all the elements of the rundown then create the named preset.

Playing Items in a Rundown

The selected rundown items can be activated by a mouse click in the client menu or by pressing the keyboard function keys. The function keys operation changes very slightly between media and templated character generation items. The key actions are summarised in the tables below.

Still and Video Replay

Key	Key Name	Description
F1	STOP	Immediately stop replay. Remove the video or still from the layer of channel.
F2	PLAY	Load the video or still if needed. Start playback actions using set transitions.
F3	LOAD	Load the video or still producer and await further instructions. When Freeze on load is set the target first frame is shown.
F4	PAUSE/ RESUME	Pause or resume replay.
F5	NEXT	When the **Freeze on load** and **Trigger on next** flags are active in a clip, this key starts playback.
F6	UPDATE	-
F7	INVOKE	-
F8	PREVIEW	If a preview channel is defined in the client configuration pressing the preview key shows the first frame or still on the designated preview channel.
F9	unused	-
F10	CLEAR	Clear layer as set by output control segment of media item.
F11	CLEAR LAYER	Remove and reset parameters on selected channel. Mixer parameters are **not** changed.
F12	CLEAR CHANNEL	Remove all sources of video from all layers in the channel. Reset mixer properties to default values.
Shift F2	PLAY NOW	Forces the delays entered into an item of a group to be ignored.

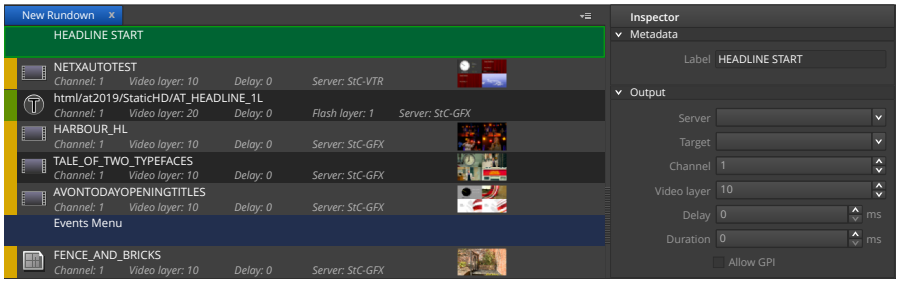
Templated Graphics

Key	Action	Description
F1	STOP	Fires any exit transition then removes the producer from the layer when out transition is complete.
F2	PLAY	Load character generation producer and template, then call the routine that runs opening animation.
F3	LOAD	-
F4	PAUSE/ RESUME	-
F5	NEXT	Start next stage of animation.
F6	UPDATE	Send new text to character generation template without using an input animate. It may use an animation to remove old text and add the new text.
F7	INVOKE	Calls a function in the character generation template. User provides function name and any parameters as part of the command.
F8	PREVIEW	-
F9	unused	-
F10	CLEAR	Clear
F11	CLEAR LAYER	Clear Video Layer
F12	CLEAR CHANNEL	Clear Channel
Shift F2	PLAY NOW	Forces the delays entered into an item of a group to be ignored.

Visually Divide a Rundown into Blocks

It is helpful to have a rundown divided into identifiable blocks - such as headlines, interview 1, music item etc. The CasparCG client supports named blocks via a coloured separator bar.

The separator bar is part of the **Tools Other...** command group. It can be accessed in the library Tools window, or via a right-click in the client rundown area. The separator bar defaults to a solid red background with no metadata. When the bar is added to a rundown the Metadata label text is shown in the separator.

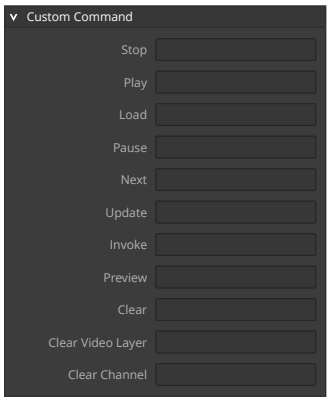


The example rundown at the left has two separator bars visible, one at the top and one near the bottom (dark blue background). The text in the first bar is a mirror of the Metadata label text. There is a limited set of colours available for the separator. The colour can be selected from the list available in the right-click **Colorize Item** context menu. The colour may also be applied to media or control items in the rundown.

Custom Commands

One of the challenges in playing through a rundown without operator error is the need to use different keys to action some sections of the rundown. The **F2** function key can be used to load and immediately play a video clip, or play an already loaded clip. Most programme directors want to see the first frame on an insert clip before calling for it to be cut to the output of the vision mixer. This can require the CasparCG operator to load the clip using **F3**, subsequently playing the clip using **F2**. Whilst this should not be a major issue for an experienced operator, new users my find swapping keys more challenging.

The **custom command tool** provides a mechanism that enables a rundown sequence played using just one function key. The complexity of this solution is the rundown creator has to know some fine detail of the AMCP control protocol used by a CasparCG server.



Maps to:
F1
F2
F3
F4
F5
F6
F7
F8
F10
F11
F12

The monochrome screen grab at the left shows the properties inspector for a newly added custom command. The dark boxes immediately to the right of the white text are where the AMCP command text is inserted.

The light green box to the right shows how the function keys map to the boxes. Function key F9 is **not** used.

The mapping enables any function key to be used to issue any AMCP command - thereby enabling the F2 key, normally used as a **play** command, to issue a **load** command.

Details of the AMCP language are available in the CasparCG online wiki at https://github.com/CasparCG/help/wiki/AMCP-Protocol

Each command description contains a mandatory element - the command verb. Required elements are shown in rectangular brackets **[ ]** and optional elements are enclosed in curly brackets **{ }**.

The parameter descriptions include the type of information that is expected in that field such as an integer or a string. An abbreviated version of the **play** command is:

```
PLAY [video_channel:int]{-[layer:int]|-0} {[clip:string]}
```

The commands are case *insensitive*, so play and PLAY are treated identically. The minimum version of the play command is:

```
play 1
```

Which would cause a video file previously loaded on layer 0 of channel 1 to start playing. Why layer 0? The command syntax shows the verb and video channel must be present in the command to the server. The second element in the command is shown in curly brackets, and hence is optional:

```
{-[layer:int]|-0}
```

The vertical bar, `|`, just before the `-0` is shorthand for an **OR** function. If there is a layer number present, for example `15`, that value has precedence.

The second optional parameter is a string type and is the core name of the file to play. When adding files to the CasparCG media folder edit the names to remove spaces, replacing them with either a hyphen or an underscore. This is because the command parser in the CasparCG server uses whitespace characters to detect the tokens present in the command line, so spaces in a file name might be interpreted as parameter separators, causing the file to be not found.

Bad	*Better*	*Best*
`Monday Story 001`	`Monday_Story-001`	`Mon20201207_StationOpening`

If the file name does contain spaces the file name in the command must be surrounded by quotation marks:

```
play 1-10 "Monday Story 001"
```

The command can take extra parameters such as an instruction to LOOP the clip, seek a start point that is not the first frame etc. The extra parameter structure is common to three commands:

`loadbg` Loads a producer in the background layer and prepares file for playout.

`load` Loads a producer in the foreground layer and prepares file for playout.

`play` Moves clip from background to foreground and starts playing it.

The full syntax of the play command is:

```
PLAY [channel:int]{-[layer:int]} [clip:string] {[loop:LOOP]}
{[transition:CUT,MIX,PUSH,WIPE,SLIDE] [duration:int] {[tween:string]|linear}
{[direction:LEFT,RIGHT]|RIGHT}|CUT 0} {SEEK [frame:int]} {LENGTH [frames:int]}
{FILTER [filter:string]} {[auto:AUTO]}
```

So to start a looping playback of file AMB.mp4 in channel 2 layer 10 starting with a 12 frame linear mix the command is:

```
play 2-10 amb loop mix 12
```

Using the same video mix in time but starting at frame 100 then playing for 250 frames before restarting the loop the command becomes:

```
play 2-10 amb loop mix 12 seek 100 length 250
```

Appendix C is cheat sheet showing graphs of the available transition easing functions.

The filter element is a reference to an ffmpeg filter, details of available filter functions is available at https://ffmpeg.org/ffmpeg-filters.html.

Some of the commands can be very long - but the custom command entry box will scroll if the entered string exceeds the screen display box width.

There is a simple “trick” that can be used to ensure the correct file name spelling is used in a custom command. Double click on the wanted item in the library listing to add an item to the rundown. Use the Inspector to access the Target field. Copy the content of the target field into the custom command. Remove the item that was added from the library listing.

Building a New Rundown

Most programmes use a combination of moving video, templated graphics and stills. They may also need to send triggers to external devices. Creating the first rundown instance for a programme usually requires at least one copy of the programme script.

Work through the pages of the printed script, adding notes that identify all video clips, stills and captions. Pay particular attention to which output channel the director has allocated for each media item, and include that allocation on the marked-up script. Where video clips do not yet exist, for example news items still being edited, select a standard media item as a placeholder in the rundown.

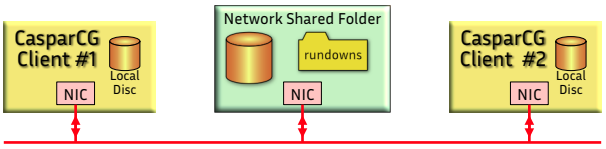
Identify the graphic templates that are needed. If new templates are required allow sufficient development time for template creation. It is helpful to maintain a document showing examples of each template in use, including the field names used in the template.

When all media elements are available, open the SVT client and a new rundown tab is created. If the client is already active, a blank rundown is created by selecting the New Rundown in the SVT client File menu, or by using the shortcut keyboard command **Ctrl N**.

Most of the detail of how to add media items to the rundown were described earlier in this document, in the section called “Adding Media and Commands to the Active Rundown”. Work through the marked-up script adding all required elements to the rundown. Most operators find the simplest way to build a video section of a rundown is to use the library search facilities as the media preview panel helps confirm the media selection.

If the rundown might be controlled by an external process, such as a web browser or OSC client, use the main menu bar Rundown sub-menu to Allow Remote Triggering. A red lightening flash is added to the rundown name tab to show the enabled status.

Finally, save the rundown using the **Save As...** Item in the File menu, or use the shortcut key combination of **Shift Ctrl S**. Select an appropriate name and folder location to store the rundown.



If more than one client may need to access the rundown, for example to split graphics replay and clip replay operations into two blocks from an existing rundown, using a network accessible store enables all interested clients access to the single resource. If the network share is not mirrored, keep a copy of the rundown on the client computer hard dive.

Sharing a rundown on multiple clients is most effective where the client configuration uses the same names for the controlled servers. The name lookup table is part of the client configuration screen. An illustrative example is shown in the screen grab above. The elements that need to match are the Name column and the associated Address and Port column.

Name	Address	Port	Description
StC_GFX	192.168.42.21	5250	Studio C Graphics Server
StC_VTR	192.168.42.31	5250	Studio C VTR Server

Check the rundown entries by playing through the items, checking the intended media or template that the media are displayed on the allocated target channel using the assigned level. Store the edited rundown using the **File Save** menu element or use the shortcut **Ctrl S**.

Day to Day Operations

This chapter examines building and editing rundowns for day-to-day operational work in a production studio. The mechanics of adding media and editing the properties are covered in the earlier chapter about the SVT client.

A typical programme requires both pre-programme events and programme transmission/recording events. The pre-programme phase prepares and starts moving graphic backgrounds, it may output a programme ident slate to identify any recording.

Adding Video Media to the CasparCG Library

CasparCG only reads media from the allocated media drive, usually mapped to a physical drive in the server configuration. The new media must be copied to the media store for playout. Good operational practice also maintains a archive store that enables subsequent replay of the media on other servers.

Whilst many media files will replay if there are spaces in the name, occasionally the spaces cause parsing errors in the AMCP command. A simple problem avoidance strategy is replacing any spaces with either a hyphen or an underscore.

Once media is added to the store the client needs to discover that content. This is achieved by using either the Library Refresh menu item or Ctrl-R key combination. The library refresh function can be set to run at user defined intervals.

Insert the newly delivered clip into the playlist, then start the replay by pressing the F2 key. Check the clip video replays without quality issues, also checking the audio is both is audible and distortion free.

Import Existing Still Images

The CasparCG Server can replay many types of still images. Five commonly encountered types are PNG, JPEG, BMP, TIFF and TGA.

The CasparCG Image replay software will attempt to scale any input still to fit the active channel output dimensions. For high-definition operation files should ideally be 1920 x 1080 samples. Standard-definition sizes, 1024 x 576 or 720 x 576, will scale to the high-definition output, but may look slightly soft.

Usually HD resolution images are fast to recall but images with very high resolution, say over 4000 samples per line, add visible delay between pressing the play key on the keyboard and seeing the image on the output.

Best workflow practice is therefore to conform a still, such as one from a digital camera or phone camera, to a 16:9 shape via cropping, then scale the image to 1920 x 1080 . Save the file as a PNG or JPEG image, but do ***not*** use spaces in the file name.

Move the prepared images to the CasparCG Server computer either over the computer network or via a pen drive. Use the file browser on the server to copy your images into the CasparCG Media folder of the server computer.

Go to the CasparCG Client display and use Ctrl R to refresh the library listing held by the client. The new images should now be available in the library listing, but it may take a short time before the preview images to be available in the client preview panel.

CasparCG and Alpha channels

The CasparCG system is designed to support alpha channels in any replay media compositing operation. The alpha channel may be activated in either of two mechanisms:

1. Using a file format that supports an alpha channel. Example formats are ProRes codec in a MOV wrapper for video, and PNG and TGA for still images.

2. Using a second file with **_a** added to the primary name. So if there are two files, one called wilty_titles.mov and the second called wilty_titles_a.mov, CasparCG auto-selects wilty_titles_a.mov file as the alpha channel for wilty_titles.mov. The file name association also works for still images that do not include an alpha channel.

Adding Video Clips to a Rundown

The full list of files on the sever media store can be long and need much scrolling to find a needed item. Use the library filter tools for name and CasparCG server selector to limit the list size. After the clips are found and added to the rundown clear the library filters.

When a video clip is added to a rundown the default selections set the first played frame to be the first frame of the clip. If the clip includes a countdown clock (eg as commonly specified for network programme delivery) the first displayed frame can be set as the first frame of actual content by entering a value into the **seek** field of the clip item.

Using a drag and drop or double clip to add the clip to the rundown automatically sets the name of the server in the Target field of the inspector. Set the output channel number to match the programme script allocation, setting the channel layer to be used. The default layer is 10 and this works for simple replays. Enable or disable the Freeze on load box in the Video segment of the item inspector to match local working practices.

Adding Stills to a Rundown

Various programmes requires sequences of stills forming a events menu. This section examines options for adding such sequences to a rundown. Stills are added to the rundown using the same processes used for video clips.

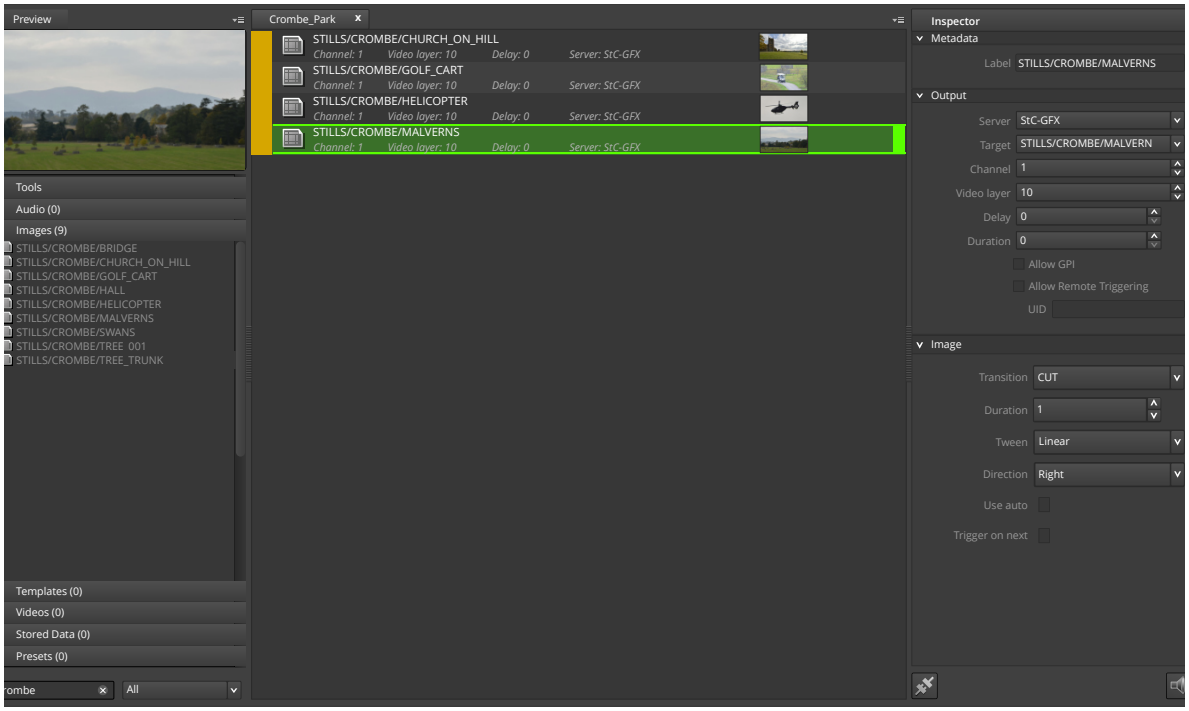
The detail of adding stills to a rundown depends if the director wishes a preview of the next still to be transmitted.

Stills Replay using only Transmission Output

In the left-hand column of the client, select the Images tab. Enter or clear the filter string as appropriate. Careful management of the server media sub-folders creates library selection boxes that need minimal scrolling of the library window.

Selecting a still in the Images list usually shows the still in the client Preview window. If the preview does not appear this is because the preview for that still has not yet been created at the server and sent to the client. If a still has been added to the media store, but the name is not visible in the client library list this may be because the library listing needs a refresh. Use the Ctrl R key combination or the menu Library Refresh item to initiate a client update.

After some images have been selected the client window may be similar to the example below.



As each image is added to the rundown, it is allocated a default channel number, video layer, transition type and Tween (transition) type. Set the Channel to the output designated for still outputs.

Images can be added to the rundown in any order. It is easy to re-position a still in the sequence, or to make a duplicate entry, for example when the image is needed in a headline sequence and as a main programme item.

Unwanted images are removed from the rundown by selecting the image then pressing the **DELETE** key.

Changing the image position in the rundown order is effected by either selecting the item and using the mouse to drag the image to a new position, or by single clicking the still to select it, then pressing and holding the control key whilst using the up or down arrow keys to move the position of the image in the rundown.

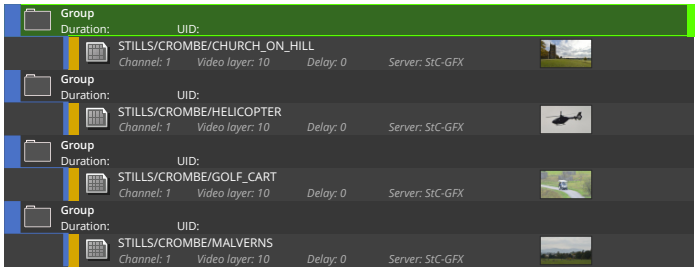
The position change techniques also work for a multi-element selection (click then shift-click or multiple ctrl-clicks) followed by drag or Ctrl Up/Down arrow.

When a mix transition between two slides is required, select the entry for the slide that will fade in, setting its transition mode to MIX, then set the duration of the mix and the Tween type. Commonly used Tween modes are Linear (same as EaseNone), EaseInOutSine and EaseInOutQuad. The available tween transitions are illustrated in Appendix C. The same detail edit process is used for other transitions such as Push, Slide and Wipe.

During rundown playout the operator selects the still to be load, then presses the F2 key to play it onto the output. The next item is manually selected, either by mouse click or by cursor down arrow.

Auto Advance, Single Channel Output

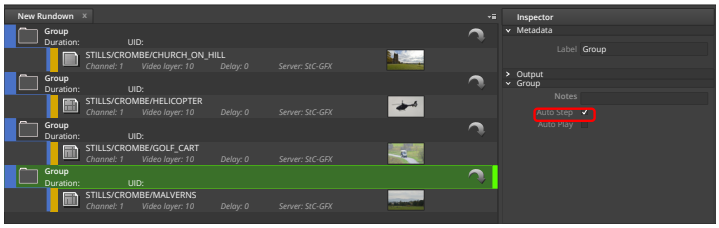
This mechanism enables an item to be played with the selection cursor automatically advancing to the following item in the rundown. This allows the playout operator to concentrate on issuing the play commands when the director calls for a stills change. Initially build the rundown as described in the preceding transmission output only section.



Rundown showing Images in Groups

Select the first still item in the rundown, then press the control and G keys. This puts the image into a Group. Select each image in turn and add it to its own group. After adding several groups, the rundown will look similar to the example shown to the left.

The indent (movement to the right) of each media item beneath the group header is the indication that the item is a member of the group. The advantage of the group wrapper for stills replay operations is that the group can have an ***auto-step*** property enabled. The location of the group auto-step property in the inspector is indicated by the red rectangle highlight in the client screen shot below.



Groups with Auto-Step Enabled

Select each group in turn, setting the Auto step property. It is good practice to edit each group's Metadata Label property to identify the image in the group. This metadata label identifies the stills when the group display is collapsed. A double-click on the group bar collapses or re-expands the group display.

Playout Process

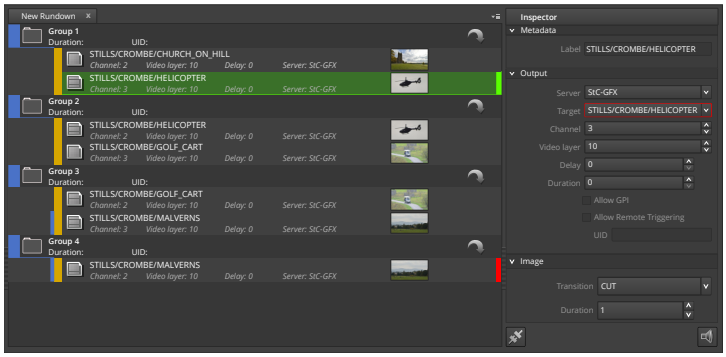
1. Select the first group in the rundown using the mouse.

2. Press the F2 key to transition the image to air. The Auto step mode selects the following group ready for a play command.

Auto Advance with Tx and Pv Outputs

For some programmes, it is desirable to have a preview of the next slide in a replay sequence. This process requires two CasparCG server outputs. This processes is supported using the Group mode with those two output channels. Assume for this description that channel 2 is transmission output and channel 3 is the preview output. All groups in the slide sequence apart from the last now have two images; the image to send to transmission and the image to send to preview.

The client display is similar to that shown in the following illustration.



Group Mode with Tx and Pv Channels

In this example, Group 1 contains two images. The first image in the group (CHURCH_ON_HILL) is set to output on channel 2 (Tx), but the Inspector shows us the second entry in the group (HELICOPTER) will output on channel 3 layer 10.

The second slide for each group can be inserted by selecting the first slide of the group in the rundown thus setting the target for item inserts, then selecting the second slide in the library and dragging or double clicking, followed by changing the output channel in the inspector.

Alternately the second slide can be copied from the following group by using the keyboard. There are four steps:

1. Select the slide in the second group, then press ctrl D to duplicate the image reference. Leave the duplicated reference selected.

2. Use ctrl left arrow to take the slide out of the second group

3. Use ctrl up arrow to move the reference above the second group

4. Use ctrl right arrow to join the reference to the group above the source of our copy.

5. Set the Output Channel to number 3.

The above process is repeated for each group in the rundown. The final group needs only the transmission channel image.

Using Mix Transitions

It is good practice to delay the change of the preview output until any transmission output mix is complete, then cut the new preview still onto the preview output. The human visual cortex is adept at detecting changes. It can detect the cut of the preview output even when the observer is not focussed on that section of the vision stack. Hence the director “knows” they can request the next still change as soon as they need it on air.

This postponed preview output change mechanism is implemented by setting a value for the Delay property of the preview still. Set the preview delay time to the same value as the transition duration for the transmission still.

Playout Process

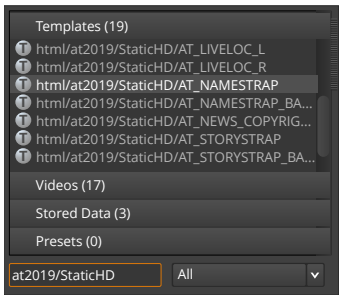
This is the same as used for the "no preview" output operations:

1. Select the first group in the rundown using the mouse.

2. Press the F2 key to transition the image to air. The Auto step mode selects the following group ready for the next play command.

Using Templated Graphics

Templated graphics can be used on any CasparCG server channel either as overlays shown on top of other content, or output as SDI fill and key for combining in a vision mixer keyer. The following descriptions assume the templated graphics is to be output on server channel 1. .



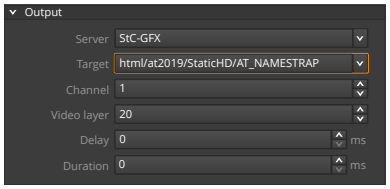
The templates available on the server or servers are listed in the Template tab in the left column of the standard client. If an expected template name is not shown in the library list remove any strings in the filter control and select all servers. Use the library refresh (*Ctrl R* on Widows) if the expected template is still not in the library list.

Best practice is storing templates in folders that are named to match the programme strand. This naming convention enables the folder name to be used as a filter string for library searches, as shown in the example at the left.

There is no preview picture for a template. Template creators should be tasked with providing a document, suitable for electronic display or print, showing the name of the template, an example of the template in action, and a list of placeholder names. It is also possible to program templates that show their placeholder names on the selected channel if a suitable named key and value are sent in a play command, or via an invoke function.

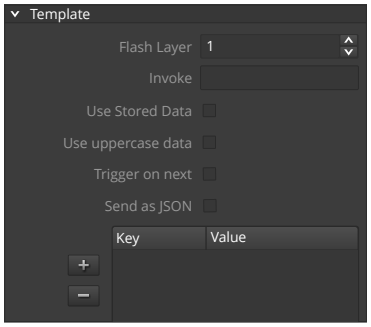
Select the required template in the library list. Either double-click the template name, or drag the template into the rundown sequence part of the client window. Either of these actions set the source server name, selects channel 1 as the output and sets the layer to 20.

Select the template instance in the rundown, activating the inspector for that item. In the Output tab, confirm the server name and the channel number are the intended settings.



By default, templated graphics are set to use layer 20. This places templated graphics in front of video replays running in their default layer - layer 10. It may be necessary to change the default layers for some complex graphics applications. Leave the **Delay** and **Duration** fields at zero for captions that will be inserted and removed by the vision mixer operator.

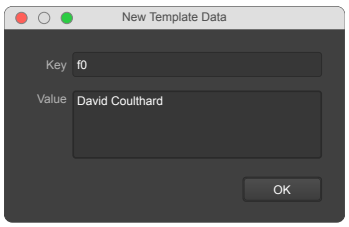
Setting Instance Data



Selecting a template in the rundown sequence shows the Template properties panel in the lowest section of the inspector. An example template properties panel is shown at the left of this text.

There are several control fields followed by an entry box headed with the labels **Key** and **Label**. The instance data is entered in this area.

Click the + button to pop up the New Template Data entry box. The Key field is initialised with the next available ***f*** plus ***number*** (eg f0 as shown in the example New Template Data box). The key name can be edited as required by the template, but many templates are created to use keys named f0, f1 etc for speed of entry or editing.



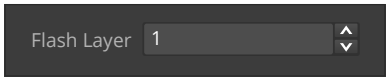
Note the entry box is closed by clicking on the **OK** button. This is because the box allows the entry of multi-line text, each line terminated by a carriage return.

When a key already exists in the template properties display, the Value is edited by double clicking the Key or Value entry in the list.

As each template is activated on an output channel using the ***Play*** command (pressing function key F2) the instance key names and values are formatted into a string sent to the assigned server. The script

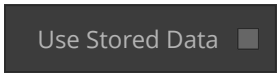
code in the template extracts the keys and their values, inserting them into the placeholder fields of the graphic.

There are several control fields that may need editing.

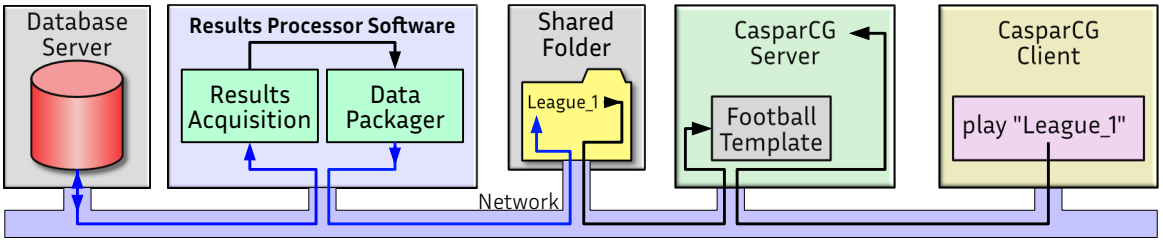


Flash Layer. This field is only relevant to Flash templates and allows the data to be directed at one of several layers within the Flash Action Script code. Leave value at 1 for HTML template use.

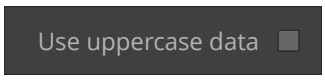
Invoke. This field can contain the name of a function in the template code that is called when the template I***nvoke*** key (function key F7) is pressed. Leave this field blank unless the invoke process is in use.



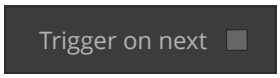
Use Stored Data. When selected, this tick box tells the server to fetch template data from the CasparCG data folder location defined in the server configuration file. This process is very useful where a large amount of data must be passed to a template, for example the scores for the entire set of clubs in a specific football league. The external data is typically stored in the designated folder by an external process as illustrated below.



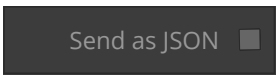
The name of the stored data is passed to the template as the value element of placeholder name f0. The CasparCG server code fetches the data and presents it to the **update()** javascript function as if the data were included in the ***play*** or ***update*** command.



Use Uppercase Data. Instructs the client to convert all text fields into upper case before sending the command and data to the server.



Trigger on Next. When this flag is enabled the play command is not sent to the server until the Next key (function key F5) is pressed. This allows the graphics operator to prepare the next caption for display, then have an external system send a GPI or OSC trigger to the client firing the **next** operation.



Send as JSON. This flag changes the encoding of the data between the client and the server.

It is only relevant to a CasparCG HTML template. XML is the default encoding used in the data transfer. A single f0 data field with a value of **My Name** would be sent as:

```
<templateData><componentData id="f0"><data id="text" value="My Name"/></componentData></templateData>
```

Setting the JSON flag converts the data transmission into:

```
{"F0":"My Name"}
```

The scripting language for HTML, JavaScript, includes many functions for JSON processing. XML support is a relatively complex process in JavaScript, so using JSON can simplify template programming.

In practice, both example transmission modes have to be wrapped in quotation marks to delineate the data segment of the overall command. This requires that a backslash escape character is used ahead of the embedded quotation marks making the JSON format transmission string into:

```
"{\"F0\":\"My Name\"}"
```

Templated Graphics - Client Function Keys

There are seven function keys used for templated graphics replay control.

PLAY
Used to initiate playback of a template with instance data. May use animations to control the display of user provided data.

STOP
Used to initiate the removal of template from the display. May use animation to remove the data and other graphic elements.

NEXT
Timelines can contain programmed animation pauses. For example a results graphic needs to progressively reveal sections as the presenter describes the information. The animation timeline pauses at the end of the first phase. The user sends a ***next*** command to instruct the template to play on to the next pause.

UPDATE
Used to change one or more data elements but leave other graphics elements unanimated. Frequently used to update one or more text fields whilst leaving underlying graphic elements unchanged.

INVOKE
Used to call a specified function in the template. This allows random hops around a timeline or switching of context.

CLEAR LAYER
Clears the layer in which the template is currently active WITHOUT running any exit transitions.

CLEAR CHANNEL
Delete all layers in the selected channel. No output animations are run.

Grabbing Still Images from a Live Source

Many CasparCG installations have the ability either SDI or network transported inputs. The inputs can be used to grab stills into the media store. The following description uses an SDI input as the example source to illustrate the grab operation, and it assumes CasparCG channel 3, previously used as a Stills preview, can be used to monitor the grab.

There are 5 steps to the grab process:

1. Use the router or mixer aux bus to deliver the SDI signal to the Decklink or Bluefish card.
2. Route the SDI input channel to an SDI output channel to enable viewing of the input source.
3. Instruct CasparCG to take an image capture from the source. Repeat this capture operation as many times as needed.
4. Remove the route from the input channel to the output channel (STOP the route).
5. Instruct CasparCG to remove the live stream from the input.

There are two methods available for the grab event in step 4 - ***print*** and ***add image***.

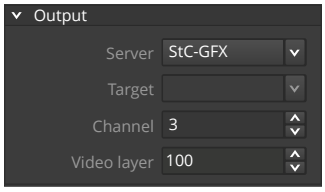
Preparing to grab images

Route the desired grab source to the CasparCG server input. In this example Decklink connector 8 is the input source.

Do not mix the programme rundown and the grab operation. Create a new rundown using the File Menu, New Rundown, or use the shortcut key combination Ctrl N. Make the new rundown the active rundown. There are two methods to set the required input:

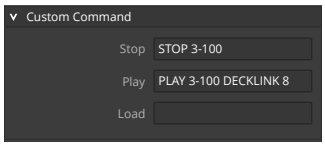
Method A

Add an instance of the Custom Command to the empty rundown. Edit the metadata label to indicate the purpose of the command - for example **Grab Input Control**. Set the properties of the output and custom commands using the Inspector panel:



i.) Select the CasparCG server (StC-GFX in this example).

ii.) Set the channel to 3 and set the video layer to 100. Using layer 100 leaves lower layers unmodified by the grab operation.



iii.) Enter text into both the Stop and the Play command boxes:

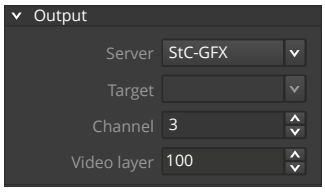
Stop box: **STOP 3-100**

Play box: **PLAY 3-100 DECKLINK 8**

Case does not matter in the commands, lower case text works identically.

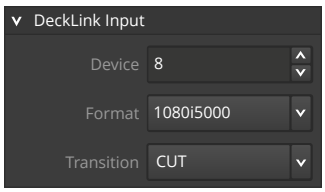
Method B

Add an instance of the Decklink Input command to the rundown. This command is available in the Tools Other section of the tool palette, or it can be inserted using a right click over the rundown display, then selecting Tools → Other → Decklink Input from the context menus. Use the Inspector panel to:



i.) Select the server (StC-GFX in this example)

ii.) Set the Output properties. Channel 3, Layer 100.



iii.) Set the Decklink Input Properties. Device 8 and input format 1080i5000

The input format is important. If the SDI input format does not match the value in the Format field there is a black video input.

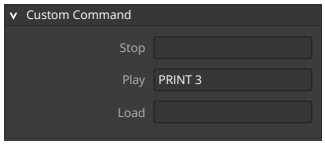
Capture a still using AMCP Print Command

The print command captures a single frame, storing the captured data in a PNG file. The file name is auto-created from the capture date and time, expressed in a modified ISO 8601 date and time format.

The name format is **yyyymoddThhmmss.png** where **yyyy** is the year, **mo** is the month number (01 to 12), **dd** is the day number, **T** is a field separator, **hh** is the hour (00 to 23), **mi** the minute (00 to 59), and **ss** is the second at which the print command ran.

The captured file is placed in the root of the CasparCG server ***media*** folder. Files captured using print should be renamed to more human readable versions, remembering not to use space characters in the name. Good practice also moves captured stills into a programme related media sub-folder.

The CasparCG SVT client does not have a pre-defined tool to issue the print command. Use a custom command instance to run the print command.



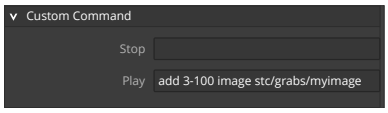
Add a custom command to the rundown in use for grab operations. Set the metadata label to say Capture Still or similar ident. Set the server to the required destination.

To capture the preview output on channel 3 use the string PRINT 3 in the Play box of the custom command properties edit form. Each time the F2 key is pressed a still is captured from the signal feed on output 3. To capture a different channel edit the number after the PRINT command.

Capture a still using Add Image Command

An advantage of the add image mechanism is captured picture is named as part of the capture instruction. The following descriptions assume the video input has been routed on BNCS and the SDI input routed to channel 3 layer 100.

Prepare the capture commands



One capture command is needed for each image that is grabbed, or a single capture command can be used, editing the file name prior to the grab.

Add an instance of Custom Command to the rundown. Select the server using the properties Inspector. Enter text into the Play box of the custom command using a format similar to:

```
add 3-100 image stc/grabs/myimage
```

This command, when run using the F2 key, adds the image producer to the channel and layer selected in the command, captures the still to a file named in the command, then removes the image producer.

When the file name includes a path, as shown in the above example, the path is *relative* to the CasparCG Media folder on the server, and that path ***must already exist***.

IMPORTANT: Do ***NOT*** use spaces in the filename.

Restore standard outputs when captures are complete

When all captures are complete, select the command used to route the source video, then press the F0 Stop key to clear the use of the channel and layer used for stills preview. If there is active content on one of the lower layers in the channel the lower layer or layers will show on the channel output.

If the grabbed images are required in the future, the images should be copied from the media folder or sub-folder onto an archive store.

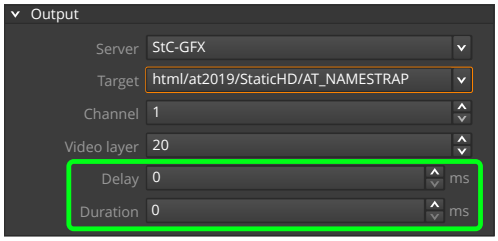
Finally use the library refresh command, Ctrl R, to ensure the client library lists are up to date.

Advanced Operations

This chapter describes some advanced modes of control.

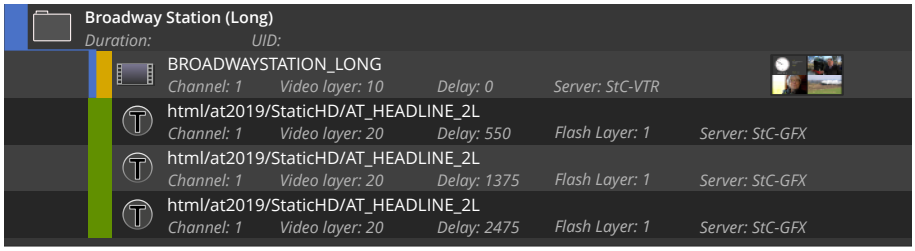
Auto Timed Caption Replay

Edited news stories may need multiple captions added during clip transmission. The insertion and removal of the caption could be done manually under the guidance of the production assistant who is timing the programme.



The CasparCG standard client can be configured to insert and remove captions on a channel at times relative to the start of the clip. The mechanism uses a group to combine the clip and associated templates. The delay and duration fields of each template are set in the template Output panel inspector. These fields identified by the green rectangular ring on the example panel at the left.

The template is instanced as many times as needed for the clip, and the video replay command and template instances are grouped together. The **Delay** value gives the time, relative to the start of the video clip, when the caption starts ins input animation. The **Duration** sets the time after the start of the input animation that the caption will start its output animation.



In the above example, the video clip called **BROADWAYSTATION_LONG** uses three captions, hence there are three template entries in the group, each entry has a delay relative to the start of the video clip. The duration field for each template is not shown on the template summary on the rundown.

The delay and duration times can be entered as frames or milliseconds; the entry mode is defined in the CasparCG client configuration.

The user selects the group item and starts playout using the F2 play key. The client issues template Play and Stop commands at their defined times.

Virtual Channels

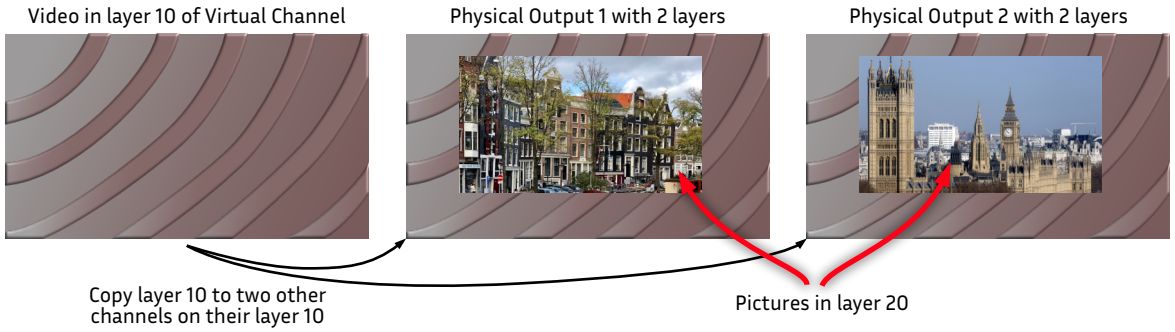
CasparCG supports the use of ***Virtual Channels***. Such channels have defined properties such as the video standard, but are not allocated to any output consumer.

Virtual channels can receive and execute AMCP control commands. The commands are executed just as if there was a connected physical output. Virtual channels have access to the internal mixer/keyer, so they can add and remove templated graphics above a video clip or create a picture-in-picture overlay.

CasparCG server supports the use of both channel routing and single channel layer routing. The virtual channel can be routed to one or multiple physical channels. This mechanism supports applications such as sending a moving background to appear behind picture overlays, and splitting a high-resolution video clip to feed multiple in-vision monitors such as a monitor wall.

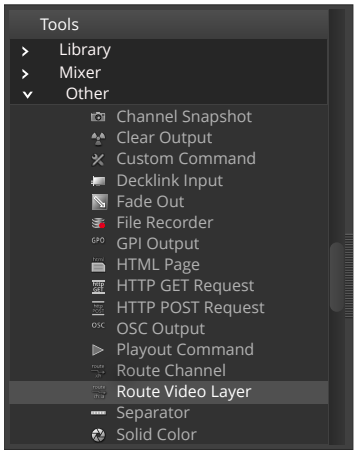
A Common Background for Multiple Output Channels

The illustration below illustrates how a virtual channel can be sent to two physical channels. Each physical channel performs layer mixing to overlay a reduced size still over the common shared background.



Using the route process ensures the two backgrounds of output 1 and output 2 remain in sync with each other, enabling a vision mixer to transition between the two outputs with no temporal offsets on the background areas.

Virtual channels are defined in the CasparCG Server configuration file and are created when the server software starts.

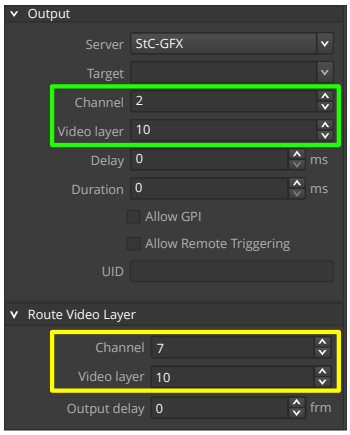


The CasparCG SVT client has a command available that enables a channel or layer route without the operator needing detailed knowledge of the underlying AMCP command.

The illustration to the left shows the Tools tab of the client, with the Other tools section expanded. The route video layer command is highlighted by the lighter grey bar.

The selected command, route layer or route channel, is inserted into the rundown by double-clicking the tool or by click-dragging the tool to the rundown panel.

Select the route command in the rundown panel to inspect the source and destination routes.



The route properties panel for a channel layer route is illustrated at the left. A green rectangle outline shows the copy ***destination*** settings, and the yellow rectangle outline shows the ***source*** channel and layer.

When the route command is played (key F2) it makes the route sending the AMCP command:

```
PLAY 7-10 route://4-10
```

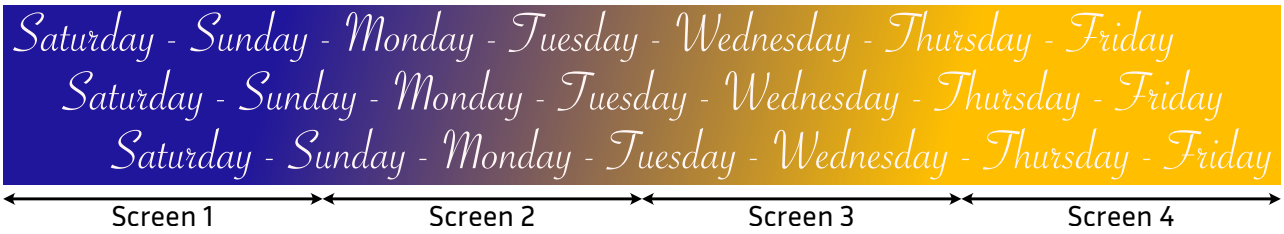
Before or after the route is activated the virtual channel can be given a command to play an image or a video that then appears on the routed layer of the destinations of the route or routes. There is a small latency (1 frame) added by using the route layer mechanism.

The example route illustration near the top of this page shows extra layers added to form the production requirement composite result, using the CasparCG server's mixers ability to change the size and position of a layer element.

Creating a Monitor Wall

A high-resolution virtual channel can be used to drive multiple monitors placed such that they form an in-vision monitor wall. Assume four in-vision monitors are needed as part of the set, and that the video content flows from left to right on the video wall. The video wall content is played in a high-resolution virtual channel using channel routing plus the channel mixer to create the physical set of outputs that drive the monitors. The example that follows is based on using an Ultra-1 HD source creating four HD screens.

The video for the wall is created using a software tool such as Adobe After Effects or Apple Motion to create a file that is 7680 x 1080 pels - the size of four full-HD screens adjacent to each other. This file can be designed as a seamless loop, or it can be rendered with sufficient duration that a single play lasts longer than the programme that uses the clip. A simple example of the quad-size source is shown below.



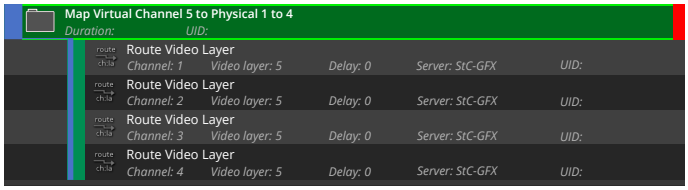
The second stage process uses a video edit package to create an Ultra-1 HD video clip. Two copies of the super-wide source are placed one above the other using a position offset on the lower copy to keep the right-hand side of the source. The resulting Ultra-HD clip is as shown below, where a light green dotted box has been added to show the display targets for the four quadrants.

CasparCG server plays the Ultra-HD clip in a virtual channel, using layer routes and mixer size and move commands to each of the four physical channel outputs.

A production could require one or more layers added in front of the wall content. Video clips and stills default to layer 10, hence a good operational choice is placing the video wall content on layer 5 to avoid later "finger trouble" when adding the inlay elements. It may be easiest to use the same layer number for both the virtual and physical channels.

Source			Destination	
Channel	Layer		Channel	Layer
5	5	→	1	5
5	5	→	2	5
5	5	→	3	5
5	5	→	4	5

Assume that the physical outputs are channels 1 to 4 and the virtual channel is number 5. The 4 routes needed are shown in the adjacent table.

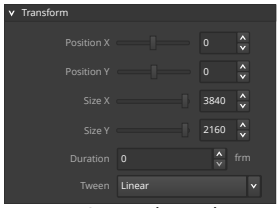


Placing the four route commands in a group provides a simple mechanism to activate the four routes by a play command directed at the group. After creating the group edit the group label to identify the function of the group.

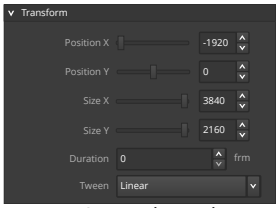
The inherent auto-fit operations in CasparCG mean that the routed Ultra-HD signal is sized to fit the HD resolution output channels. The size change is auto-implemented by the graphics mixer in CasparCG.

That default re-size operation is modified by further commands from the CasparCG client, instructing the mixer not to reduce the size but to apply position offsets such that the quadrants of the source are mapped onto the HD outputs of the physical channels.

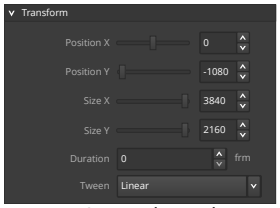
Four transform commands are required, one for each physical output. Placing these transforms in a group allows speedy application of the offsets and sizes, and the commands can be included in the routing set-up group. The transform properties for each channel are shown below.



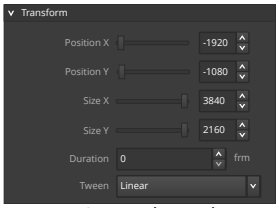
Settings - Channel 1



Settings - Channel 2

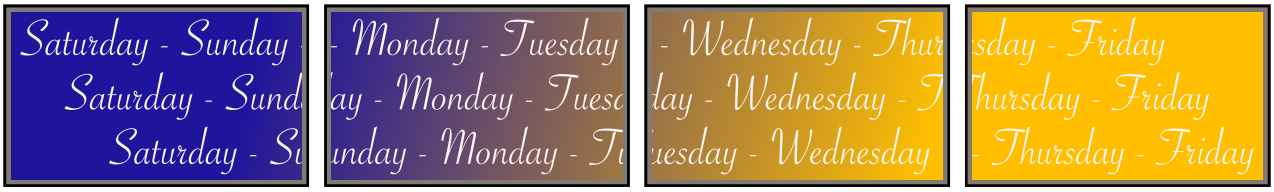


Settings - Channel 3



Settings - Channel 4

When the video is played in the virtual channel, it creates a video wall display similar to the one shown below.



It may be desirable to separate the configuration of the video wall from the playback of content on that wall. This is simple to implement by using multiple rundowns - one having the configuration commands and the second having the content playout commands.

Mixer Commands - Defer Action

The CasparCG server mixer supports a large range of image manipulations including adjustments of size, contrast, rotation, position cropping, masking and changing layer blend modes. A mixer transitions can be instantaneous (cut), or set to complete over a user specified time. The available mixer commands are listed in the tools tab of the SVT client.

Multiple manipulations on a single channel layer send a sequence of commands to the server, each command acknowledged by the server. This sequential transmission can cause a problem where some transformations intended for simultaneous start actually start at offset times. The biggest issue is the time offsets are not consistent when a command sequence is re-run. The random offset can be avoided by using the ***Defer*** facility in the mixer commands.

In the mixer command transform properties inspector panel there is a **Defer** tick box. Set the Defer flag on all the transformations that are in a sequence group. When the server receives a deferred mixer command it places that command in a stack awaiting a "go" command.

The sequence of commands is activated by a ***Commit*** command. When this command is processed in the server it will action all of the items in the delayed process stack. This mechanism ensures a reproducible effect. The commit item is available in the Tools panel, Mixer command set.

End Credits - Stacks:Rollers:Crawlers

There are at least three methods to create and output end credits:

1. Use still pictures containing the text and associated alpha channel.

2. Use templated graphics with scripted control of colours, typeface and fount properties.

3. Use a video editing package or motion graphics package to create a video file with an associated key video.

The final credit is normally a static full frame picture that contains the copyright data and production company information.

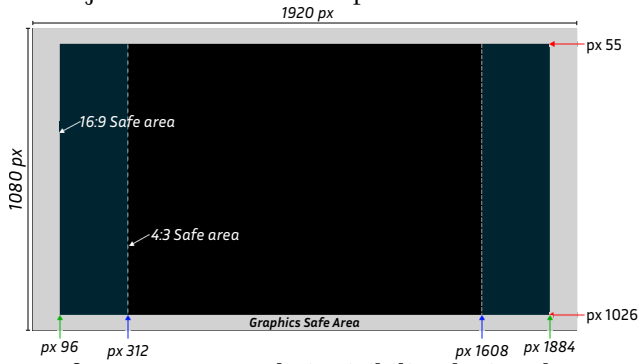
Credit slide preparation uses widely available still-picture editing software such as *Gimp* (free, open source), *Adobe Photoshop* (commercial licensed) or *Serif Affinity Photo* (commercial licensed). The credit slide or slides are saved both in the editors native format, to support further edits, and as a bitmap file such as TGA, TIFF or PNG. It is also possible to use presentation software such as Powerpoint or Keynote (macOS)as the editor, exporting the slides as bitmaps. Credit slide replays use the still picture replay methods described earlier in this document.

Roller and crawler end credits use a slightly different CasparCG image producer called the ***image scroller***. This producer requires an input slide with one dimension that matches the native display width (for rollers) or height (for crawlers).

Any alpha channel is either embedded in the still, for example in a PNG file, or is saved as a second file. CasparCG auto recognises external alpha channel files using a simple naming convention which adds **_a** to the end of the filename stem. Note that it is better to use uncompressed or lossless compressed files for credits, with PNG offering small files with lossless data compression and an embedded alpha channel.

Credit Stacks

A typical programme needs several stills slides to show all the credits, hence the **credit stacks** label for this segment. Credits should use the safe area guidelines appropriate for the geographic area of transmission. The following description uses the UK technical delivery standard specifications used by the major broadcasters. The positions of the 16:9 and 4:3 safe areas for high-definition productions is illustrated below.



Credit guidelines commonly state programme credits must be 4:3 safe to support programme sales to areas still using 4:3 transmission chains.

These constraints allow network playout presentation to use an end-credit squeeze that creates space for summary data about other programming shown at one side of the screen. Keeping the credit roller or credit stills inside the 4:3 safe zone ensures their visibility during the squeeze process.

Planning overall caption layout is an essential starting point ahead of any data entry. Some programmes need a set of horizontal guide lines used as the baseline for text boxes. Other programmes can use a single multi-line text box whose properties, such as fount (font) size and line spacing, are well defined. Where different fount sizes are required for a job title and the persons name grid lines provide a better solution.

It is sensible to create one or more `~~` templates for the credit layout required, including the design for the end board which has the copyright date and any production company logos. Save the template for later use, either as a named template for the text editing programme or as a file that is copied and renamed before data entry occurs. Write protecting the file that is copied to make new instances provides a degree of protection against “bad finger” operation.

Rundown creation requires finding the programme credit slides in the CasparCG server media folder. Storing the set of credit stills in a named folder simplifies both search process and archiving processes.

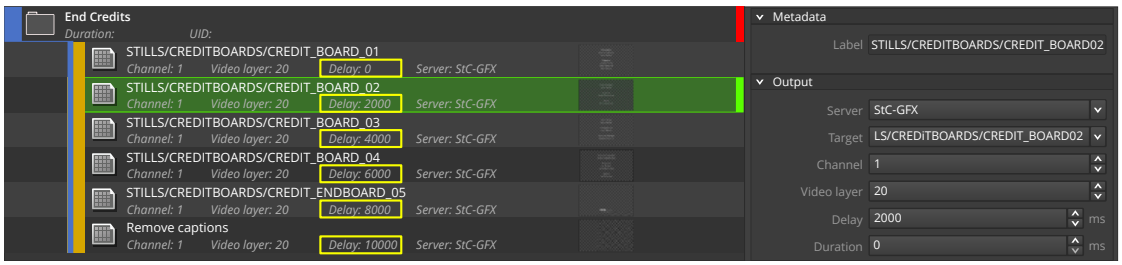
Do ***NOT*** use spaces in the folder and file names as this can cause problems for the file location process in the CasparCG. Use underscores or hyphens as word separators.

The CasparCG client library lists need updating once the credit slides are stored in the server media folder. Use either a mouse click in the client programme menu **Library** element or use the keyboard shortcut **Ctrl R** to update the lists of available media.

Use the library filter tools to locate the captions in the stills tab, loading the slide set into the rundown.

Full frame credits over a black background can be played out as stills into the vision mixer. If any of the credits will be shown over the studio output using the mixer DSK route follow good operational practice by including a ***Clear Channel*** command used ***before*** the mixer DSK is enabled.

Replaying a sequence of end-credit stills can use manual playout of each slide. Alternately the credits can be grouped together with user entered delays, relative to the first caption, for each caption in the group. A single play command addressed to the group sequences the credit stills playback. The group mode is illustrated below. The delay values in milliseconds, highlighted here in yellow edged rectangles, differ for each member of the group.



The above rundown example contains four credit stills plus the end board that has the production logo and copyright date. The final entry in the group uses the special file name **EMPTY** to cause a final blank output. Setting a mix transition mode for this final slide fades out the end board.

Rollers

A simple credit roller can be played using the image scroller tool. The still image file must be full screen width (1920 samples for HD). The height of the image is as large as required to hold the full credit list.

This is illustrated by the adjacent example roller caption. When a roller is played by CasparCG server the top of the image starts off screen, just below the bottom of the screen display area, then the movement is started.

The movement continues until the final line of the image has scrolled off the top of the screen display area.

Preparing a roller slide is, generally, relatively simple using a text layer in a bitmap editor.

A simple workflow starts by creating the list of credits as lines in a simple text editor, saving the data in case further edits are required.

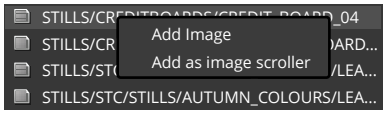
Open the bitmap editor and create a new slide 1920px wide by, say, 6000 px high. The final height is adjusted once all text is visible in the bitmap. Select the text tool, placing a text box starting near the top of the bitmap and extending to the bottom of the bitmap.

Set the typeface/fount(font), the text height, and select centred text entry. Copy and paste the text from the text editor into the text field of the bitmap editor. Edit the position and format to create the desired results. A full size black background layer may

help to see the captions as they are checked and positioned. The background can be retained for a full screen credit system, or it can be hidden before the final bitmap export.

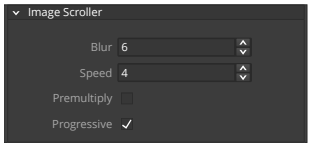
Finally edit the canvas size to leave just a small amount of canvas below the text list. Save the bitmap file in native file format.

Apply any filters, such as a drop shadow, required for the desired display then export the bitmap as a PNG file storing this file in the CasparCG server media folder for the programme.



Refresh the CasparCG client library.

Locate the roller slide in the Stills tab. Select the slide, then right click, selecting **Add as image scroller** from the pop-up menu.



Set the client output properties to the allocated channel and playout layer. The image scroller defaults to layer 10. Use a **channel clear** command before the credit replay commands in case residual text from a templated graphic is present on layer 20.

Use the Inspector panel to set the Image Scroller properties. Experiment to find the optimum **Speed** and **Blur** values. Adding some blur tends to make the roller less juddery, but it also softens the text.

Higher scroll speeds tend to need higher amounts of blur to avoid caption judder. Also test the effects of the Progressive flag on output quality. The speed parameter can only use positive integer values.

Crawlers

Creating a crawler uses the same workflow as the roller, but the fixed size for a crawler is the ***height*** of the bitmap file - 1080 lines or px for high-definition. Adding the crawler to the rundown and adjusting the blur and speed parameters is identical to the process described above for a roller caption.

If the crawler text is to display above a graphic strap there are two simple mechanisms to set up the strap. Method one includes the strap in the bitmap passed into the scroller. Method 2 uses the graphics layer capabilities of the CasparCG, with the strap placed in a lower layer of the channel than the scroller bitmap with alpha. A group is created for the replay, with the opening and closing animations of the strap managed by standard CasparCG transition and delay controls.

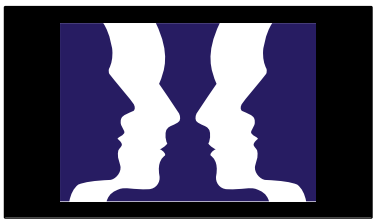
Static Opening Title

Some programmes do not use video clips for their opening title sequence. They may use shots of the studio overlaid with text from the character generator as their programme title. Such titles can use either templated graphics or stills with alpha channels.

Using templated graphics provides easy editing of programme name or presenter name via the template inspector panel. The slight complexity when using templated graphics method is the time and specialist knowledge required to create the HTML/Javascript code, although this may be mitigated by use of a visual design tool such as Google Web Designer. Subsequent style changes also need specialist knowledge and preparation time.

Where the programme name is known a short time in advance of the studio session it is usually quicker to use a still bitmap with alpha channel. The stills producer output passes via the channel mixer where size reduction and placement of the title can be adjusted using the CasparCG mixer transform tools.

If the title uses a graphic logo as well as name text, two stills and two layers provide highly flexible options for the layout, including on-screen dynamic changes. Some examples using two layers are shown below.



Layer 30 - Full frame

Layer 40 - Full Frame + Alpha

Combined Output

Resize Example 1

Resize Example 2

Output of Mixer Keyer

The logo is placed in layer 30 with the text name in layer 40. The layer combination places the text in front of the logo when the layers are combined as shown in the top row right hand side. Using mixer transform commands allows the two layers to be independently sized and placed creating the options shown by the first two examples on the lower row of pictures. The reduced size title elements can be layered above studio pictures as shown in the lower right picture.

Logs and Log Housekeeping

CasparCG servers and clients each keep logs of their actions. There is no automatic purge of the logs inherent in either server or client. Monitoring and limiting the storage of log files requires a local management process. One possible solution is to use a windows scheduled application that deletes log files that are older than a user-selected age.

CasparCG Server Logs

CasparCG servers store log files in a folder whose location is specified in the server configuration file. Server configuration files are stored in the same folder as the casparcg.exe executable. The default configuration file is **casparcg.config**. It is possible to use a different configuration file name by passing the full name of the configuration file to the server using a command line parameter. For example to use configuration file **casparcg_SD.config** use command line:

```
casparcg.exe casparcg_SD.config
```

The use of other file names is complex in server version 2.3.X_LTS as the media scanner task only looks for casparcg.config. It is simpler to keep master versions of multiple configurations, then use a batch file to copy the required configuration naming the copy *casparcg.config*.

The configuration file uses XML to define many operational properties. There are two elements that are significant for the log process → **log-level** and **log-path**. The block below is a section from an example configuration file. The red coloured tags shown here are the log file related items.

```
<log-level>info</log-level>
<paths>
  <media-path>M:/CasparCG/CCG/media/</media-path>
  <log-path>C:/CasparCG/CCG/log/</log-path>
  <data-path>M:/CasparCG/CCG/data/</data-path>
  <template-path>M:/CasparCG/CCG/template/</template-path>
  <thumbnail-path>M:/CasparCG/CCG/thumbnail/</thumbnail-path>
  <font-path>M:/CasparCG/CCG/font/</font-path>
</paths>
```

The log-level text controls the detail of reports stored in the log. The minimum amount of data is stored when the log-level is set as fatal. The supported values for log-level settings are shown in the table below.

log-level	Log content includes
fatal	Events that cause the server to have unrecoverable errors.
error	Events that are unable to successfully complete the request.
warning	Events that may have operational consequences. This may indicate a missing hardware feature, or an internal timing issue.
info	Lists all events. This level allows support teams to check that the server is receiving requests and includes the source tcp/ip address of the requests.
debug	Intended for use when developing the server software and wanting to check the sequence of sub-functions involved in a request.
trace	Very fine detail of each operation. Very large logs are created. This level is mostly used during server software development.

The log includes the chosen level plus the information from the higher levels in the above table. Thus a log-level setting of info will cause the log to store messages from the warning, error, and fatal levels.

The log-level defined in the configuration file sets the log-level when the server starts operation. The level can be changed during operations using an AMCP control message. The command is takes the form:

```
LOG LEVEL info
```

Log messages are displayed in the server's text window and stored in the daily log file. The log file location is set by the <log-path>...</log-path> token value of the configuration file. It is good practice to store the log on a different drive to the one that holds the media. The log file name uses the format **caspar_yyyy-mm-dd.log** where yyyy is the year, mm the month, and dd the day of the month.

The panel below shows an example of the extracted from a log file with the log-level set at **info**.

```
[2021-02-19 13:59:04.314] [11724] [info]    Received message from 127.0.0.1: LOAD 1-10 "MAE_BRADBURY/HARBOUR_CLIP" CUT 1 Linear RIGHT\r\n
[2021-02-19 13:59:04.322] [11552] [info]    ffmpeg[Harbour_Clip.mov|720x576p25.00|0/1512] [video-decoder] DV (Digital Video)
[2021-02-19 13:59:04.323] [11552] [info]    ffmpeg[Harbour_Clip.mov|720x576p25.00|0/1512] [audio-decoder] PCM signed 16-bit little-endian
[2021-02-19 13:59:04.324] [11552] [info]    ffmpeg[Harbour_Clip.mov|720x576p25.00|0/1512] [audio-decoder] PCM signed 16-bit little-endian
[2021-02-19 13:59:04.330] [11552] [info]    transition[empty=>ffmpeg[Harbour_Clip.mov|720x576p25.00|0/1512]] Initialized
[2021-02-19 13:59:04.331] [9188]  [info]    Sent message to 127.0.0.1:202 LOAD OK\r\n
[2021-02-19 13:59:04.335] [7716]  [info]    [frame_muxer] deinterlace_bob 720x576i50.00
[2021-02-19 13:59:05.170] [11724] [info]    Received message from 127.0.0.1: LOAD 2-10 "LOADTEST01" CUT 1 Linear RIGHT\r\n
[2021-02-19 13:59:05.193] [10600] [info]    ffmpeg[Loadtest01.mxf|1920x1080p25.00|0/3000] [video-decoder] VC3/DNxHD
[2021-02-19 13:59:05.197] [10600] [info]    transition[empty=>ffmpeg[Loadtest01.mxf|1920x1080p25.00|0/3000]] Initialized
[2021-02-19 13:59:05.197] [9188]  [info]    Sent message to 127.0.0.1:202 LOAD OK\r\n
[2021-02-19 13:59:05.211] [8876]  [info]    [frame_muxer] simple 1920x1080i50.00
[2021-02-19 13:59:17.706] [11724] [info]    Received message from 127.0.0.1: PLAY 1-10\r\n
[2021-02-19 13:59:17.706] [9188]  [info]    Sent message to 127.0.0.1:202 PLAY OK\r\n
[2021-02-19 13:59:21.730] [11724] [info]    Received message from 127.0.0.1: PAUSE 1-10\r\n
[2021-02-19 13:59:21.731] [9188]  [info]    Sent message to 127.0.0.1:202 PAUSE OK\r\n
[2021-02-19 13:59:26.202] [11724] [info]    Received message from 127.0.0.1: PLAY 2-10\r\n
[2021-02-19 13:59:26.202] [9188]  [info]    Sent message to 127.0.0.1:202 PLAY OK\r\n
```

Each line of the log includes four elements - the date and time of the log entry, the thread_id that generated the entry, the category level of the entry, and the log message.

In the example log above, the first line shows a message was received from tcp/ip address 127.0.0.1 (Localhost) and that message requested a video clip to be loaded for playout in layer 10 of channel 1. The \r\n at the end of the line indicates the command ended with the required terminating sequence of carriage return (\r) and line feed (\n). The next three line of entries were issued by the ffmpeg producer reporting properties of the clip. The fifth line was issued by the mixer sub system stating it is ready to transition from an empty layer to the video content. The sixth line is the message returned to the client, in this instance a success report.

SVT Client Logs

The SVT client, like the CasparCG server, keeps logs of actions and received messages. The client also needs configuration data such as the tcp/ip address of CasparCG servers and other controlled broadcast kit such as switchers and cameras. The client configuration data is stored in an SQLite database.

The database and the client log files are stored in a CasparCG specific folder called **.CasparCG** which is held in the users home folder - typically ***C:\Users\User_Name***. The configuration is stored in .CasparCG\Client\Database.s3db and the logs are in .CasparCG\Client\Logs

The log file name uses of the format **Client_yyyy-mm-dd.log** where yyyy is the year, mm the month and dd the day of the month. An extract from a log is shown in the box below.

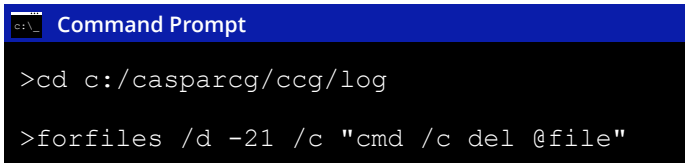
```
[2021-02-25 16:50:08.053] [3288] [D] Starting CasparCG Client 2.0.8.012c50b49a
[2021-02-25 16:50:08.124] [3288] [D] Using SQLite database
[2021-02-25 16:50:08.768] [3288] [D] Listening for incoming OSC messages over UDP on port 6250
[2021-02-25 16:50:08.808] [3288] [D] Sent message to 127.0.0.1:5250: VERSION SERVER\r\n
[2021-02-25 16:50:08.809] [3288] [D] Sent message to 127.0.0.1:5250: INFO\r\n
[2021-02-25 16:50:08.813] [3288] [D] Received message from 127.0.0.1:5250: 201 VERSION OK\r\n
[2021-02-25 16:50:08.830] [3288] [D] Received message from 127.0.0.1:5250: 200 INFO OK\r\n
[2021-02-25 16:51:11.873] [3288] [D] Sent message to 127.0.0.1:5250: PLAY 1-10 "STILLS/STA/MISC/BLETCHLEY_LAKE_01" CUT 1 Linear RIGHT\r\n
[2021-02-25 16:51:11.898] [3288] [D] Received message from 127.0.0.1:5250: 202 PLAY OK\r\n
```

Each line of the log includes four elements - the data and time, the thread id, the category level, and the log message text. The category levels are [D] for debug, [W] for warnings, [C[for critical, and [F] for fatal.

Deleting Log Files

Deleting old log files can be done manually on every server and client computer. View the Windows screen output from the host computer. Use the Windows file manager to navigate to the relevant log folder. Use the list display tool to show the log files in date order. Select the unwanted older files, fully deleting them from the system (ie do ***not*** leave the deleted files in the recycle bin).

Windows operating system includes some tools that can speed the deletion of older files, both manually and through a scheduled run process. The **forfiles** tool can be run in a DOS command window. This tool uses various command line switches to act on files in the currently selected directory. For example to delete files that are 21 days or more older, open a DOS window and enter commands as shown in the box below, selecting the relevant directory in the first command line.



The first line changes the working directory to the folder holding the CasparCG server log files. In this example the log file location is the one defined in the example configuration file. The second line uses the **forfiles** command to look for files that are more than 21 days old then runs a delete operation on each such file.

It is also possible to combine the path to scan into a single command line as shown below:

This command can be saved in a batch file for ease of running.

Windows includes a task scheduler that can run an event daily, weekly, or monthly. The description below should work for both Windows 7 and Windows 10 host computers to set up a once per week purge of old logs.

Use the taskbar search tool to find and run **taskschd.msc**. In the right-hand side pane select the Create Basic Task... Wizard and run it. Enter a task name, for example ccglogpurge, and a description for the task, then click on the Next> button. Set the trigger to Weekly, then press Next>. Choose a day of the week and time when the task is to run, and set the Recur to every 1 week. Do **not** use Sunday 0100 to 0300 to avoid issues when daylight savings start and stop. Press Next>. Set the Action to Start a program, then press Next>. Fill in the Program/Script and arguments boxes with the data based on the entries shown in bold below (the path setting is from the example configuration).

Multiple entries can be created for each set of log files that need management. If the host computer is not continuously powered it is also possible to run the purge process described above whenever the computer is powered.

CasparCG Server Configuration

Each CasparCG server requires customising to deliver the end-user requirement. Server deployments may use:

- A single-screen consumer output
- One or more network-based outputs (eg NDI)
- Multiple SDI outputs
- A hybrid combination of output types.

Each physical and virtual channel of a server needs initial spatial and temporal resolution values. The spatial and temporal properties can be adjusted whilst the server is running via AMCP commands.

When the CasparCG server software starts, it reads a configuration file stored in the same folder as the CasparCG server executable code. The default configuration file is called **casparcg.config** but multiple configuration files can exist, each having the **.config** file extension. On startup, the CasparCG server software checks the command line supplied. If the command line has no parameters the server uses the default configuration file, otherwise it assumes the first parameter is the name of the configuration file to use.

Note the media scanner system used in server version 2.3.X also needs to access the configuration file. Use a batch file to copy the desired configuration file, naming the copy as *casparcg.config*.

CAUTION: An incorrect configuration file can stop server operation. Before starting any configuration edit ensure there is an up-to-date copy of the current configuration file. If the edited configuration then proves unworkable the file copy provides a mechanism to restore functionality.

Loading Server Software

New versions of the server or client software are released occasionally. These may provide bug-fixes or add new features that are operationally desirable for Studio C use. The installation process is relatively simple, requiring manual installation of couple of software dependancies if the host computer has not previously been used for CasparCG.

The software dependancies are:

1. Microsoft C++ run-time support library
2. Microsoft .NET framework V4.0 or later (normally already present on a Windows 7 or Windows 10 Operating System).
3. Driver software for any Blackmagic Design Decklink or Bluefish444 SDI cards
4. NDI iVGA output software for CasparCG servers prior to version 2.3. Obviously this software is only needed if NDI outputs are required. Version 2.3 has built in NDI version 4 producers and consumers.

Downloading any needed Dependancies

Both the NRK and LTS windows versions of the server are built using Microsoft Visual Studio. The NRK branch uses Visual Studio 2015 and the SVT branch uses Visual Studio 2017. There is a common C++ runtime available for VS2015, VS2017 and VS2019. When this document was written the common runtime library (vc_redist.x64.exe) was available from Microsoft [here]. Before downloading and installing the support library check if already exists on the computer, as another task may have loaded it.

To check if the runtime library software is already loaded, run the Windows Settings tool, select Apps... from the displayed set of options, then select Apps & Features from the left-side menu. Scroll the list to the Microsoft section and examine the files. Look for *Microsoft Visual C++ 2015-2019 Redistributable(x64)*.

The Blackmagic Design Decklink drivers and other utilities are part of a package called Desktop Video which is available from the Blackmagic Design website support pages [here]. Generally the latest version of desktop video is the one to use. When this package is installed it adds the Decklink configuration tool, the Media Express capture tool, and a disk speed check tool to the host computer applications. Again check the computer to see if a recent version is already present. Look for Blackmagic Desktop Video. If present, a single click shows the product version.

Downloading of the iVGA driver needed for NDI output in CasparCG version 2.1.xx_NRK involves a lot of searching as many links in search engines are broken. CasparCG version 2.3_LTS branch (or later) has built-in support for NDI inputs and outputs.

IMPORTANT: If downloads are required, follow local policies that ensure downloaded files are free of unwanted content such as viruses, spyware, trojans and adware.

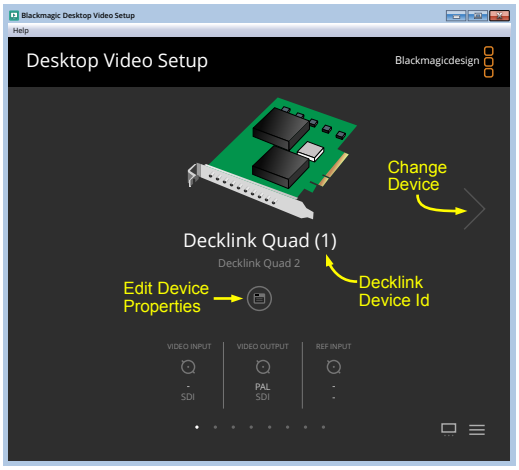
Each dependancy has a simple wizard to guide people through the install process. The Decklink diver system will ask for permission to re-boot the computer. When the re-boot occurs there may be several instances of a Blackmagic Design pop-up asking to search for the latest drivers. Allow the installation using the *search the host machine for drivers* option.

The Decklink install process may also show pop-up menus if it finds the board firmware is not current, and it wants permission to run the firmware updater.

Once the drivers are available the Decklink cards require several property values setting, especially allocation of SDI outputs on a multi-output card.

Example of Blackmagic Design Desktop Video Set Up

This software is installed as part of the Desktop video installation. It can be started using a shortcut in the Windows Start Menu. At task startup it shows a screen of the style shown below, without the yellow arrows and text.

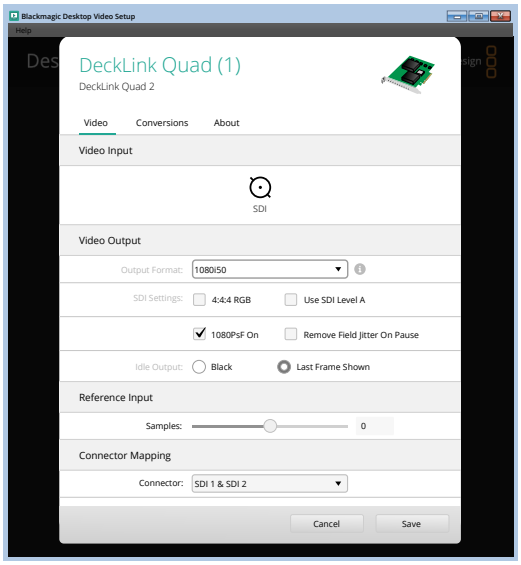


The picture of the card shown at the upper centre of the application window changes according to the type of card being configured.

The Device ID (shown in brackets, see yellow highlighter text) is significant for CasparCG Operations. It is used as part of the mapping of a CasparCG logical channel to a physical output.

The series of eight white dots at the bottom of the software window correspond to the Device ID. The pictograms above the dots show the video settings for the selected channel. In the example the output is an SDI signal, the video output format is set to 576i25 (“PAL”), and the REF Input shows the selected reference. This display element is only available when a reference is connected to the Quad 2 card.

Click the Edit Device Properties button to display the properties form. This will be similar to the picture shown in the following page.



For this illustrative example assume the first output channel has both a fill and key signal, requiring two SDI connectors.

Set the Connector Mapping (Drop down selector near bottom of the edit screen) to "SDI 1 & SDI 2".

Set the Output Format to required value - eg 1080i50.

Do ***not*** activate the Remove Field Jitter on Pause

Set Idle Output to "Last Frame Shown".

Set the reference to 0 samples offset.

Save the config. This may require confirming to windows that the software may change the currently saved settings.

Repeat the config exercise for the other seven Decklink IDs. The major differences are in the Connector Mapping. Devices 2, 3, 4, 6, 7, 8 should be set to select just a single SDI connector.

Device 5 should be set as Connector NONE because the nominal connector was grabbed for use by channel 1 key output.

After a device edit, review the settings for all 8 devices to check no accidentally changes to a connector mapping that has affected a partner device.

Once all Decklink configuration is complete, reboot the PC. Whilst this may not always be required, experience shows it reduces problems with subsequent CasparCG Server configuration.

Installing CasparCG Server Software

If the CasparCG is an update to an existing installation take copies of all **.config** files in the existing CasparCG folder. These will be restored after the updated server software is installed.

The software for both the NRK and SVT LTS branches is available from Github. Most search engines return a valid link using a search string of either **github casparcg server nrk** or **github casparcg server lts**. When this document was created the NRK release was available here, and the SVT LTS version was available here. Click on the **Releases** button in the browser page to ensure the latest release is at the top of the page.

Create or locate a suitable folder on the target host computer to hold the unzipped file content, for example C:\casparcg\unzips. Copy the zip archive to a temporary folder on the target computer, then select the zip file in the Windows file browser. There may be packaging structure differences between the various server distributions. If there are folders in the archive, search for the one called **server** and open that folder.

Click on the **Extract all** icon of the file browser. A pop-up window appears with a target folder for the extract. Browse to the chosen extraction folder and click OK to extract the archive.

Create the target folder that will hold the running copy of the files - for example a folder that contains some identification of the server version such as C:\CasparCG\Server_2d3d3_LTS. Select all files in the unzip folder and copy to the target folder.

If this installation is replacing an existing server version copy the .config files saved earlier in the install process into the new folder.

Delete all desktop shortcuts to CasparCG server and create a new shortcut to the batch file called **casparcg_auto_restart.bat**. Whilst the NRK server can be run directly using the exe file, the LTS branch has

a separate media folder scanner task that must be run before the server code starts. The batch file starts the scanner and server as required by the release version.

Editing the Server Configuration File

The configuration file defines the storage paths to the media and template files, the spatial resolution properties of output channels and the control access ports. The default configuration file is called casparcg.config and it is available in the same folder as the CasparCG executable task. The file uses XML tags to define the properties, and it can be edited using any text editor.

The bare minimum configuration would contain:

```
<?xml version="1.0" encoding="utf-8"?>
<configuration>
     <paths></paths>
     <channels></channels>
     <controllers></controllers>
</configuration>
```

<paths>

The paths section of the configuration defines the paths to the resources the server needs to access. The paths may be set relative to the folder where the server executable is stored, or absolute including the drive letter and the paths. Take care entering paths, as the common Windows path separator, \, is an escape character in XML encoded files. If the media is stored on a drive known to Windows as **M:** the two methods to enter the content are:

```
     M:\\CasparCG\\ccg\\media\\

     M:/CasparCG/ccg/media/
```

NOTE: All path entries *MUST* include the slash at the end of the path name. Omitting this will cause errors when the server runs. The two main code branches have a different number of tags.

The path configuration for the SVT 2.3.x_LTS branch takes the form:

```
<paths>
     <media-path>m:/CasparCG/ccg/media/</media-path>
     <log-path>c:/casparcg/server/log/</log-path>
     <data-path>m:/CasparCG/ccg/data/</data-path>
     <template-path>m:/CasparCG/ccg/template/</template-path>
</paths>
```

The <media-path> entry is the root folder for the video and stills media. Sub-folders are allowed in the media folder, and these can simplify media management and client library searches.

The <log-path> often points to a different drive to the other entries. It defines the location where the log reports from the server are stored. The log detail is defined by another tag in the configuration file.

The <data-path> folder is where user-defined data sets are stored. These data sets are typically written by an external computer process, and a template is directed to read the content from the named data set.

The <template-path> is where the template files and their resources are stored. Templates may share the media drive or use a different drive, leaving maximum data bandwidth for the media transfers.

The path configuration in the NRK code branch takes the form:

```
<paths>
     <media-path>m:/CasparCG/ccg/media/</media-path>
     <log-path>c:/casparcg/server/log/</log-path>
     <data-path>m:/CasparCG/ccg/data/</data-path>
     <template-path>m:/CasparCG/ccg/template/</template-path>
     <thumbnail-path>m:/CasparCG/ccg/thumbnail/</thumbnail-path>
     <font-path>m:/CasparCG/ccg/font/</font-path>
</paths>
```

The <thumbnail-path> folder is where the media scanner process of the server stores the thumbnail pictures of the video media and still images. The thumbnail pictures are sent to the client when requested. In the SVT code branch the media scanner process is a separate process used by the server code but also providing an http interface for other software accesses.

The <font-path> is the folder where fonts required by the text producer are stored. Note these fonts are **not** used by templates which use fonts from the host computer font folder or an external URL. This tag is not required or used by the SVT LTS server code branch as it does not include a text producer. If the tag is present in the configuration file the information is ignored by the SVT server code.

<channels>

This section of the configuration defines the properties of all the server physical and virtual output channels. Each channel is identified by a <channel></channel> tag pair. The order of entries in the configuration defines the channel numbers. For example:

```
<channels>
     <channel>
          <!-- Properties of CasparCG channel 1 -->
     </channel>
     <channel>
          <!-- Properties of CasparCG channel 2 -->
     </channel>
     <channel>
          <!-- Properties of CasparCG channel 3 -->
     </channel>
</channels>
```

A channel that uses a Decklink card for output will have a channel properties section of the form:

```
<channel>
     <video-mode>1080i5000</video-mode>
     <consumers>
          <decklink>
               <device>1</device>
               <embedded-audio>true</embedded-audio>
               <latency>normal</latency>
               <keyer>external</keyer>
               <key-only>false</key-only>
               <buffer-depth>3</buffer-depth>
          </decklink>
     </consumers>
</channel>
```

The <device> tag value is the Decklink card number that will output the channel. The Decklink configuration software links the number used in this tag with the SDI connector on the card edge.

The <embedded-audio> tag is set true when the channel is to output audio. Enabling embedded audio also enables other audio outputs such as analogue or AES on Decklink cards that support separate audio outputs. Pure graphics channels can set this tag to false, reducing the latency through the output by 1 frame.

The <latency> tag controls the latency through the Decklink card processing. The settings of normal or default add 1 extra frame of latency relative to low, but this default setting tends to be more stable (lower chance of a dropped frame).

The <keyer> tag tells the server how to manage the keyer that is onboard some Decklink models. If the card, such as an Extreme model, supports internal keying set this tag to external. If the configuration is to use a separate output channel for the key use a value of external_separate. To use the card to key content onto the signal applied to the SDI input connector set the value to internal. The <keyer> tag can be omitted if the channel is used for fill video only.

The <key-only> tag is set true to output only the key signal. This mode is activated when a channel is set to output the key generated by another channel.

The <buffer-depth> sets the buffer size in frames used in the path to the card. Modern 4k cards may be stable with a depth of 2, but most cards require a buffer depth of 3.

<controllers>

The entries in the controllers tag set define ports and protocols are used to control CasparCG. There are usually entries for AMCP control and OSC status and control.

```
<tcp>
     <port>5250</port>
     <protocol>AMCP</protocol>
</tcp>
```

The above example listens on tcp/ip port 5250 for AMCP commands sent from any source tcp/ip address.

Miscellaneous tags

There are several tags that control aspects of the server operation.

Channel Locking

A CasparCG client can set exclusive control of a selected channel of a server using the AMCP **LOCK** command. When locking the server the client must provide a user selected password for both lock and unlock operations. For example locking channel 1 can be achieved by:

```
lock 1 acquire brutus
```

where the password is brutus. Unlocking the channel uses the command:

```
lock 1 release brutus
```

How can the channel be unlocked if the user password is not known? One solution is restarting the server - best described as a rather drastic mechanism! The second method is to use the password defined in the <lock-clear-phrase> tag. For example:

```
<lock-clear-phrase>verysecret</lock-clear-phrase>
```

The server channel is then unlocked by a message:

```
lock 1 release verysecret
```

Log Control

The amount of information stored in the server log can be adjusted by the AMCP **LOG LEVEL** and **LOG CATEGORY** commands. The initial values for the logging process are set by <log-level> and <log-categories> tags:

```
<log-level>info</log-level>

<log-category>communication</log-category>
```

The log-level values are shown in the following table. Data is logged from the selected level plus those shown lower in the table. Thus a log-level of warning includes warnings, errors and fatal events.

Log-level	Logged Information
trace	Very detailed reports showing route taken through the software. Intended for deep debug operations during code development. Results in very large log files.
debug	Reports various diagnostics and debug properties, including dropped frames and other timing matters such as buffer allocation time usage.
info	Lists all commands and responses with clients, and shows the server startup process including the content of the selected configuration file. This label is the default and is set if there is not a valid <log-level> tag in the config.
warning	Actions or requests that result in a warning being issued.
error	Actions or commands that report errors are logged.
fatal	Only catastrophic errors are logged.

Each log entry includes a thread id and the date and time at which the log event occurred.

The log category has three options:

```
communication
calltrace
calltrace,communication
```

The default is to log the communication messages. If the communication category is set off, the log shows the messages that arrive from a client, but does not show the server response.

When calltrace is enabled it writes to file calltrace.log in the log folder enabling trace log capture even though the main log has disabled the trace level.

OSC control
CasparCG server can emit status data for the active outputs and layers using UDP packets carrying OSC data messages. This information is used in the SVT client to show the video playout progress bar. The `<osc>` tag defines the ports and addresses to be used.

```
<osc>
     <default-port>6250</default-port>
     <disable-send-to-amcp-clients>false</disable-send-to-amcp-clients>
     <predefined-clients>
          <predefined-client>
               <address>127.0.0.1</address>
               <port>5253</port>
          </predefined-client>
     </predefined-clients>
</osc>
```

The `<default-port>` tag defines the udp port number on which the status data is emitted. Each AMCP client that connects to the server receives the osc data unless the message send is disabled by the `<disable-send-to-amcp-clients>` tag.

Some CasparCG installations, for example automation playout systems such as Sofie, require knowledge of the progress of clip outputs. Such installations can receive broadcast OSC messages by defining the addresses and ports in the `<predefined-clients>` entries.

Media-Scanner
The SVT V2.3_LTS code branch has a media scanner that runs as a stand-alone task. That task is used by the main server but it also allows other access processes to request information about the available media via http get requests. The ip address and port used for access is defined in the `<amcp>` tag set.

```
<amcp>
     <media-server>
          <host>localhost</host>
          <port>8000</port>
     </media-server>
</amcp>
```

Thumbnail Images

The media scanning process creates thumbnail pictures for video clips and still images. The thumbnail generation process differs significantly between server version 2.3.X_LTS and server version 2.1.X_NRK. The 2.3.x server branch splits the server operation from the media scanning and thumbnail generation. The thumbnail images are created on request from the client. The 2.1 NRK brach media scanner and thumbnail generator are part of the main server code. Thumbnail images are stored in a user designated folder, and sent to the client on request from the client.

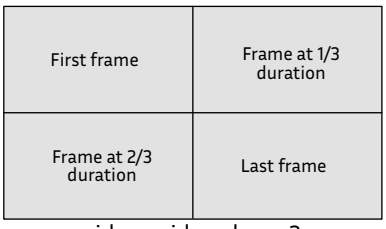
Server 2.3.X_LTS

The entries in the casparcg.config file for the media scanner access are shown in the previous paragraph in the `<amcp>..</amcp>` tags. The thumbnail image video is taken from the first frame of the video.

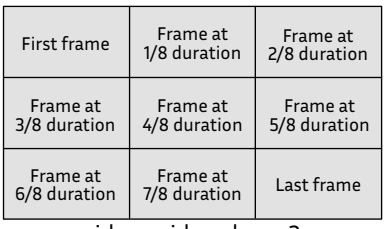
Server 2.1.X_NRK

The thumbnail images are stored in a user defined folder whose address is included in the `<paths>` tag block. The thumbnail image configuration defines several properties for the thumbnail images. A still image thumbnail is a reduced size version of the still, but the thumbnail for a video clip can be created from multiple time points through the video.

```
<thumbnails>
     <generate-thumbnails>true</generate-thumbnails>
     <width>256</width>
     <height>144</height>
     <video-grid>2</video-grid>
     <scan-interval-millis>5000</scan-interval-millis>
     <generate-delay-millis>2000</generate-delay-millis>
     <video-mode>1080i5000</video-mode>
     <mipmap>true</mipmap>
</thumbnails>
```



<video-grid> value = 2



<video-grid> value = 3

The width and height of the thumbnail image are defined in the `<width>` and `<height>` tags. The above example tag set has the width and height set for 16:9 media. The `<video-grid>` tag defines the sub-division of the thumbnail area for video clip operations in the NRK branch. The structure of the resulting thumbnail for values 2 and 3 is illustrated at the left. A value of 2 generally gives a good balance between an ability to resolve the picture information and understanding the video content progress through the clip.

If the video-grid tag value is changed, existing thumbnails are not regenerated. Manually delete the content of the thumbnail folder to force regeneration. The media store is scanned every few seconds, the interval between scans is set by the `<scan-interval-millis>` tag. Each scan checks for new, modified or removed media and updates the thumbnails as appropriate.

When the <mipmap> tag is true the thumbnail generation process may have reduced aliasing effects.

HTML Templates

HTML templates are rendered by the embedded Chromium engine. This engine supports remote debug access from the tools built into the Chrome browser. The `<html>` group tag has a `<remote-debugging-`

port> tag that sets the tcp/ip port number used. If this is set to 0 (the default value) the debug action is disabled.

```
<html>
     <remote-debugging-port>9001</remote-debugging-port>
     <enable-gpu>false</enable-gpu>
</html>
```

To use the debug access start Chrome browser on the server computer setting the URL as **chrome://inspect/#devices**

On the displayed page, tick the ***Discover network targets***, and click the ***Configure*** button. Add the localhost address with the port number defined in the <html> tag - **127.0.0.1:9001** and click the Done button. After a few seconds the should be a list of templates running in CasparCG server.

It is also possible to enable the use of the graphics processing unit, gpu, to speed rendering. When gpu support is enabled there may be some issues caused by the dual use of the gpu which is needed for the mixer effects. As at April 2021 there is also a bug report for a continued growth of memory use (ie a memory leak) when gpu acceleration is enabled. This bug is proving elusive to find!

Flash Support
CasparCG initially used Adobe Flash as the technology for templated graphics rendering. During 2017 Adobe announced that it was stopping support for Flash at the end of 2020. In January 2021 Windows rolled out an update that permanently removes Flash from a standard Windows install. There is a mechanism to continue to use Flash for CasparCG, useful for some organisations that have hundreds of templates that use Flash and ActionScript 3. The default for new CasparCG server deploys is to disable Flash support in the configuration file. If Flash is needed consult the CasparCGForum for instructions of how to install the modules and set registry keys. The edit the server configuration file to set the `<flash>` `<enabled>` tag to true.

```
<flash>
     <enabled>false</enabled>
     <buffer-depth>auto</buffer-depth>
</flash>
```

Audio Configuration
The SVT 2.3.X_LTS branch defaults to 8-channel audio passthrough, whereas the version 2.1.X_NRK server branch defaults to stereo audio operations. When more audio channels than the default are required the audio configuration must be set to enable the extra channels and to offer conversion between various modes. The mode to be used is set by a tag in the `<decklink>` section of the `<consumers>` segment of the configuration file.

The available channel layout options and the process of converting from one audio config to another are defined in the `<audio>` tag group. A sub-set of options is shown below:

```
<audio>
  <channel-layouts>
    <channel-layout name="mono" type="mono" num-channels="1" channel-order="FC" />
    <channel-layout name="stereo" type="stereo" num-channels="2" channel-order="FL FR" />
    <channel-layout name="matrix" type="matrix" num-channels="2" channel-order="ML MR" />
    <channel-layout name="8ch" type="8ch" num-channels="8" />
    <channel-layout name="16ch" type="16ch" num-channels="16" />
  </channel-layouts>

  <mix-configs>
    <mix-config from-type="mono" to-types="stereo, 5.1" mix="FL = FC | FR = FC" />
    <mix-config from-type="mono" to-types="5.1+downmix" mix="FL = FC | FR = FC  DL = FC |
                                        DR = FC" />
    <mix-config from-type="mono" to-types="matrix" mix="ML = FC | MR = FC" />
    <mix-config from-type="stereo" to-types="mono" mix="FC &amp;lt; FL + FR" />
```

```
  </mix-configs>
</audio>
```

Setting a Decklink channel output is implemented by adding two entries to the channel definition, one at the channel global level and one at the decklink consumer level. The configuration segment below shows how a channel and decklink card are set to operate in 8-channel passthrough mode.

```
<channel>
    <video-mode>1080i5000</video-mode>
    <channel-layout>8ch<channel-layout>
    <consumers>
      <decklink>
      ...
         <embedded-audio>true</embedded-audio>
         <channel-layout>8ch<channel-layout>
      ...
    </consumers>
</channel>
```

Supported tag values

When a new CasparCG server is installed the default configuration file has two blocks, the actual configuration data, and an XML comment block that shows the recognised tags and supported values. An XML comment block is present between **<!--** and **-->** markers.

Each tag set shows the tag name, followed by the default value then the list of supported values. An example from the Decklink consumer is the keyer mode:

```
<keyer>external [external|external_separate_device|internal|default]</keyer>
```

This states that the default keyer mode is external, and this value is set if the tag is not found in the configuration data entries. The recognised values are listed between the square brackets, with each option separated from a neighbour by the vertical line (pipe) character.

Do **not** just copy the options line and paste into the configuration area without editing the copy. Remove unwanted values and remove all whitespace. For example `<keyer>internal </keyer>` would ***not*** be recognised as the tag value is “internal “ and the recognised string value is “internal”

Configuration Validator Tool

Sometimes a new or updated configuration does not operate as expected. Diagnosing the error can take time and lots of testing. Standard fault-finding mechanisms can be used - a software version of the “rule of halving” to locate the problem zone can speed debug operations. Make extensive use of the XML comment markers to disable blocks if the configuration. Comments can not be nested so if the configuration includes lots explanatory comments, make a new version of the configuration without the comments in place and use that comment free version for the debug operation.

There is an online configuration checker tool available at https://casparcg.net/validator/ which may also provide useful pointers to issues. The checker tool can be downloaded and run locally.

Config Backup - Important!

It is very important to keep a well-named safety copy of the edited and proven configuration. This can be used either to restore operations after a major failure, or as the copy source when multiple active configurations are required.

CasparCG SVT Client Configuration

The SVT client has many properties controlled by configuration. The client uses an SQLite database to store client configuration data, some operational option lists such as easing modes, and copies of media thumbnail pictures. On a Windows host operating system the default folder for the database is stored in the active users workspace. With a user login name of ***myName*** the client logs and database are saved in folder C:\Users\myName\.CasparCG\Client

The default database file name is ***Database.s3db***. If required, this database can be opened and the content inspected or modified using the free ***DB Browser for SQLite*** application.

At launch, the client tries to open the database. If the file does not exist a new database is created containing default settings.

The type and content of fields in the database changes with client version. This is known to cause issues when a version 2.0.9 and a version 2.2.0 are used on the same host computer if they use the *same database file*. Client version 2.0.9 is used with server version 2.1.X_NRK, and client 2.2.0 is used with server version 2.3.X_LTS. Different clients are required because:

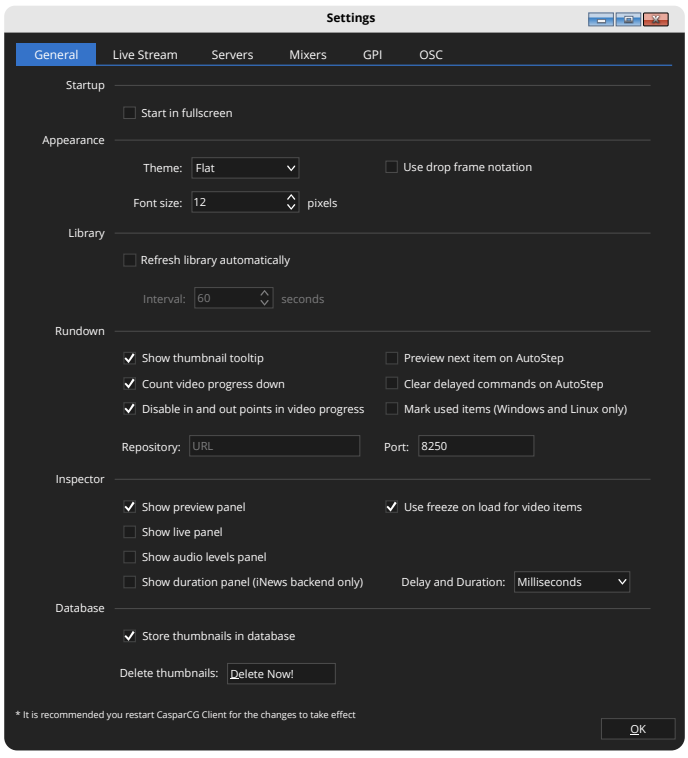
- OSC status data emitted by the servers differs in format,
- ffmpeg parameters used for live streaming of a preview are different because the ffmpeg code base has been updated for server version 2.3.

If the playout progress meter is missing on the rundown displays this may be because a client has modified the database. This error can only be resolved by deleting the database file, launching the client, and doing a full configuration.

The database file name to use can be provided as a command line parameter. The identifying argument is **-t** and is followed by the full path and name for the database file. An example command line is:

```
“C:/CasparCG/ClientV2d0d9/casparCG Client.exe” -t c:/casparclient/usedb209.s3db
```

The client configuration screens are accessed from the client application Edit menu, Settings… item. The General configuration window panel is shown below.



The Startup option allows the client to open in full-screen mode, probably the most desirable display mode in operational use.

The Appearance options define the look and feel of the client interface. The available Theme options are Flat and Curve. The latter option gives a rounded appearance to buttons and menu bars. Drop frame animation is relevant to non-integer frame rates such as 29.97 Hz.

The Library option allows the client to regularly request the state of the media libraries from connected servers. The time intervals between the requests are user adjustable when the auto-refresh mode is enabled.

Auto refresh operation is very helpful when a CasparCG server is operated in a news application where media is updating both ahead of and during the programme transmission.

The Rundown section of the General settings panel has several options that affect the rundown display. It is generally easier for users to see how much media is left in a clip, and this display is enabled by the Count video progress down option is active. The Mark used items option dims the rundown data for items that have been played. This can assist users to find the current playout position in a long rundown display. The rundown display brightness is reset when needed via a Rundown menu item Mark Item sub-menu.

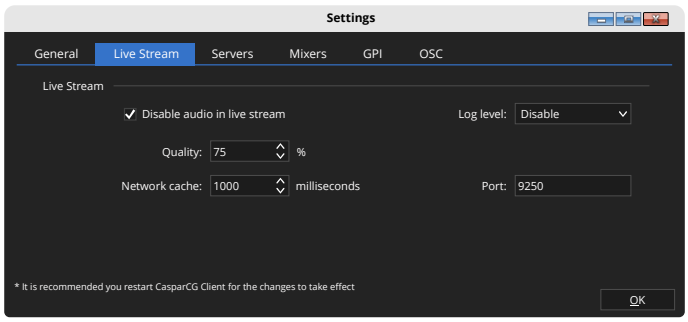
The repository URL and port are used for integration with a news production system. The endpoint is an html server system. When the client opens the repository a communication chain is created that supports adding and removing stories via a unique story id.

The Inspector section of the window controls the client Inspector panel appearance. The Preview panel is shown at the top left of the client library display section. When this panel is active, user library selections display the thumbnail still in the preview box.

The live video and audio level sub-panels are displayed at the top right of the client window area when enabled. The panels allow video fill or alpha from any channel to be seen in the client along with audio level bar graphs. Creating this preview requires spare processor capacity in the server which has to H264 encode the video content. Despite significant effort by the development team there is an as yet unresolved (April 2021) issue in the Windows server version 2.1.X NRK branch. The server crashes when the live panel values are changed.

The Use freeze on load setting for video items is a very useful operational feature, enabling a director to see the first frame of an upcoming video.

The Database section of the general settings enables thumbnail pictures to be stored in the local database, and provides a mechanism to flush the thumbnails from the database.

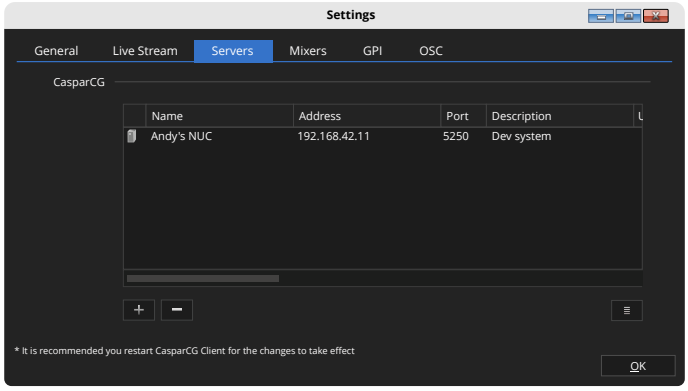


The Live Stream settings panel is illustrated at the left. Unused blank space has been removed to reduce the height of this image.

The settings in the segment are only significant when the live panel and audio panel are enabled in the general settings.

The Log Level selector controls logging for issues in the received live stream.

The Servers configuration section of the panel is illustrated below. The height of the server list section has been minimised to reduce the overall height of this graphic.



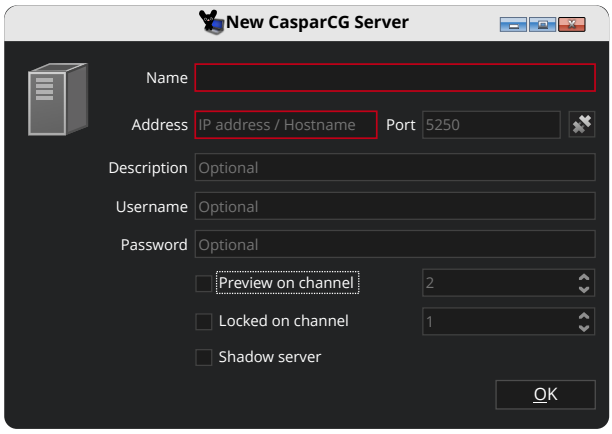
The function of this entry area is informing the client how to access the servers that it will control.

Each entry in the list includes a logical name, the TCP/IP address and port number for the server. There is also a text field that describes the server. This allows a short cryptic entry in the name field and a more reader friendly ident in the description field. For example a server name of ***StC-GFX*** could have a description ***Studio C Graphics Outputs***.

When media items are added to the rundown, or commands inserted into the rundown it is the logical server name that is shown in the drop-down server list in the item inspector panel. The logical name is also included in saved rundowns. If a saved rundown is opened in a different client that uses a different logical name for the same TCP/IP address all the entries in the rundown have to be edited to enable

control of the intended server. It is better to use a common set of names for every server available at the premises, as this ensures ease of copying rundowns between servers.

To edit an existing server entry double-click the line in the list of servers. To add or remove a sever use the **+** and **-** buttons below box with the current list of servers.



When the add server button is pressed a pop-up form requesting server details is displayed.

Enter the server logical name into the Name field, and the server IP address or Hostname into the Address field. Only use a hostname if the network includes a DHCP server that can resolve the name to a dotted IP address. The name and address must be supplied, the other fields are optional.

The Username and Password fields enable the client to use Windows verification when accessing the server. If the fields are empty the server must have an open share available to the client.

The Icon to the right of the Port number entry field runs a connection test. Pressing the icon starts an attempt to connect to the server, reporting a successful connection if it can establish the route. If no connection establishes, the client continues a background poll for the connection, reporting success if the route subsequently connects.

The F8 function key allows a user to show a preview of an item on a CasparCG channel if an output is available. The Preview on channel tick box enables the preview facility for the selected server. The server channel that provides the preview still is specified in the box at the right of the preview label.

The Locked on channel tick box constrains the client to only control of the specified channel number. When an item is added to the rundown it auto-selects the locked channel number.

The shadow server tick box stops media lists from the shadow server showing in client library lists. CasparCG assumes the installation team provided an automated media copy facility to maintain the same content on the primary and shadow servers. A github project available at https://github.com/hreinnbeck/casparcg-multicast-amcp uses node.js instances to support copying AMCP client messages directed to the primary server to the shadow servers using a multicast.

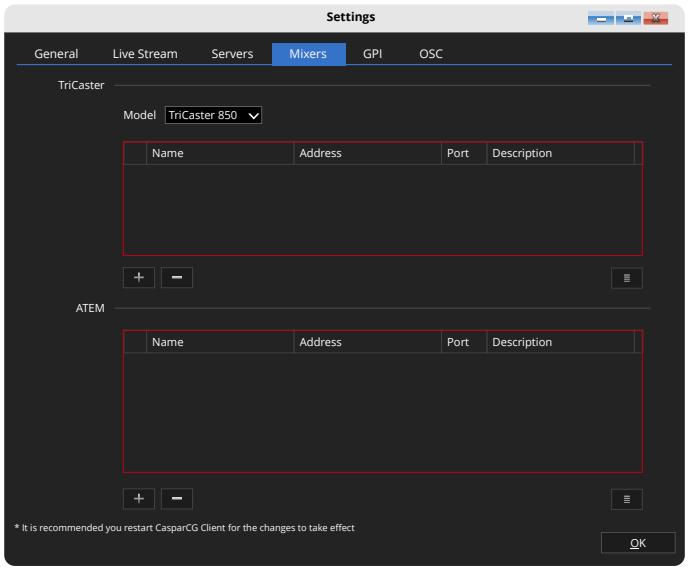
The CasparCG client can import server properties from a file containing details of many servers. A short distance below the right hand end of the bottom of the server list is a small icon with four horizontal bars. When this is clicked it allows the user to navigate to a file that holds the server list, then select servers from that list that are added to the list of controllable units. Using such a server list helps ensure the same names are used throughout a multi-server multi-client installation.

The server list is an XML document using the structure illustrated below.

```
<?xml version="1.0" encoding="UTF-8"?>
<servers>
   <server>
      <name>demoimport</name>
      <address>192.168.43.21</address>
      <port>5250</port>
      <username>myUserName</username>
      <password>myPassword</password>
      <description>Studio A Stills</description>
      <shadow>no</shadow>
      <previewchannel>0</previewchannel>
      <lockedchannel>0</lockedchannel>
   </server>
```

```
   <server>
      ...
   </server>
</servers>
```

Each XML server tag entry has nine sub-tags that correspond to fields in the manual New Server form.



The Mixers settings panel enables entry of the connection properties for Tricaster and ATEM mixers.

Existing entries are opened for edit using a double-click on the entry. New devices are added by clicking on the + button. The client displays an entry form for the properties.

The TriCaster form has fields for the name, the address, the TCP/IP port number and the description.

The ATEM form has fields for the name, address and description fields. The description is shown in the list starting in the Port field.

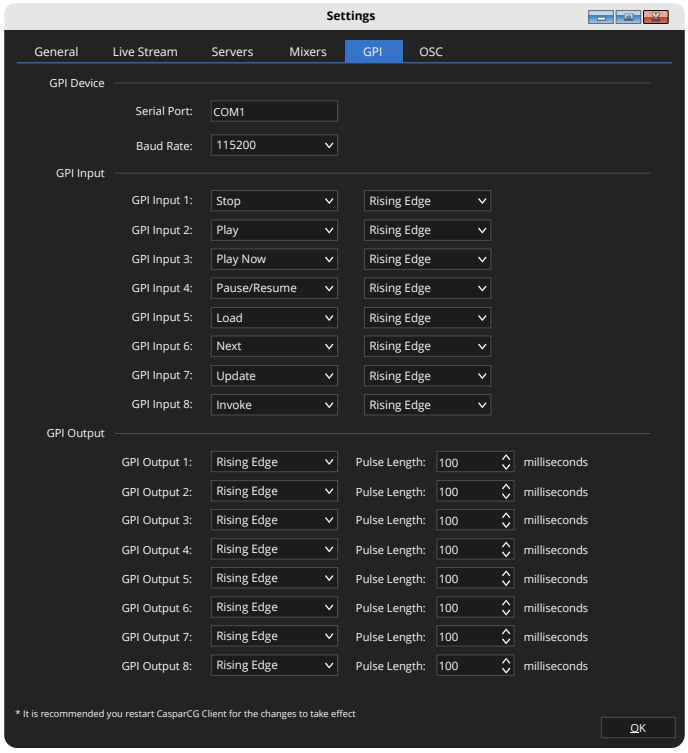
Both of the entry forms have communications test buttons.

Both types of mixer can import configuration settings from an XML file. The structure of the TriCaster XML configurations file is:

```
<?xml version="1.0" encoding="UTF-8"?>
<mixers>
   <mixer>
      <name>Tricaster V16</name>
      <address>192.168.52.11</address>
      <port>5950</port>
      <description>V16 Experimental system</description>
   </mixer>
   <mixer>
      ...
   </mixer>
</mixers>
```

The structure of the ATEM mixer configurations XML file is:

```
<?xml version="1.0" encoding="UTF-8"?>
<mixers>
   <mixer>
      <name>AtemMini 1</name>
      <address>192.168.50.11</address>
      <description>1U Studio unit</description>
   </mixer>
   <mixer>
      ...
   </mixer>
</mixers>
```



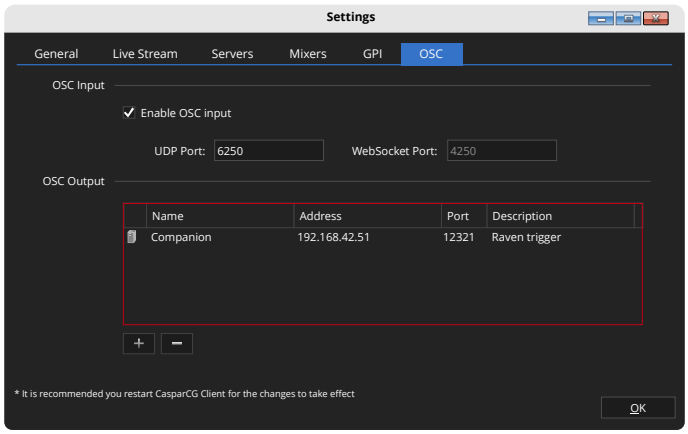
The GPI tab provides the configuration settings for the GPI input and output connections.

The GPI Device is the serial communications com port that is used for the GPI service. Two Baud rates are supported - 9600 and 115200.

GPI input properties set the active edge for each input, and the allocation of the action when the trigger condition occurs.

GPI output properties set the output state for each bit through the transition. A rising edge will be normally low, pulsing high for the duration defined in the associated pulse duration entry.

The OSC tab enables or disables OSC control of CasparCG, and defines targets for OSC messages that can be emitted by rundown events. The settings details differ slightly between client version 2.0.9 and 2.2.0

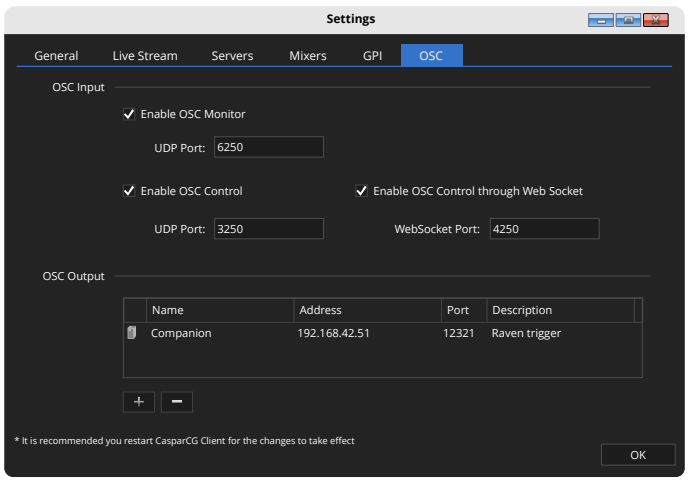


The example at the left is from the client 2.0.9 configuration.

When the Enable OSC input box is ticked the SVT client listens for OSC messages on it's own IP address on both the UDP and WebSocket ports selected by the numeric entry boxes.

Enabling OSC control enables both item-trigger and rundown-item select and trigger control.

The OSC output properties define one or more named OSC targets. The entries in the Name column are available in the drop down destination selector of the OSC Output tool in a rundown. The message sent to the destination is entered into data boxes in the OSC output element of the rundown via the element inspector.



The Osc properties form for client version 2.2.0 is shown to the left. The major change is that there are separate enable and port number boxes for the OSC monitoring and control connections.

When OSC monitor is enabled the client listens for the progress information reported by clip playout.

When OSC control is enabled the client accepts both item and rundown control triggers.

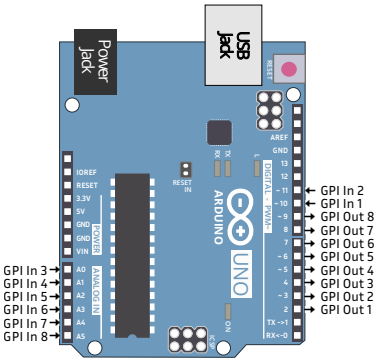
Appendix A - GPI Interfacing

There are some programme events where a user needs to trigger multiple pieces of equipment, for example to change some graphics and to change some lighting settings.

The general purpose interface (GPI) provides a simple mechanism to effect external triggers without the need to use complex communications or control protocols. The controlled equipment is user-configured to run a macro or trigger an internal event when the trigger state is received.

General-purpose computers require specialist interface cards to directly support GPI interfacing. Adding a card is possible in some desktop or server style computers, but not with a laptop. Serial RS232 interfaces are available as low-cost cards for desktop computers, and as USB adaptors for laptops or desktops. SVT adopted a serial data link using a simple message protocol to implement the CasparCG client GPI functionality. They also created an Arduino UNO based microcontroller to create the serial to GPI physical interface. The Arduino sketch code is available from a read-only Github archive at:

https://github.com/CasparCG/Tools/tree/master/cpp/gpio/trunk/arduino_gpio_controller



The allocation of the Arduino interface pins to the GPI functions is illustrated by the adjacent drawing.

Any engineer building the Arduino interface is expected to use relevant protection circuitry between the controller and the "outside world". This includes using opto-isolation on the inputs and miniature relays for the output connections.

The USB connection between the Arduino and the host computer implements the bit-serial connection used by the message protocol. Use the Windows Device Manager to find the com port number allocated to the connection, and enter the port number in the CasparCG client GPI configuration.

Modern equipment often provides an open API allowing external control via a TCP/IP connection. It is relatively simple to create a translation gateway that communicates with the CasparCG client via serial protocol, translating the assigned GPI bits into API messages sent via the TCP/IP connection. The translation gateway programming can use any language that supports both the serial messages and the TCP or UDP connection to the equipment.

GPI Serial Protocol

The message exchange uses 4-byte blocks with ASCII characters that encode the message information:

```
<byte 1> <byte 2> <cr> <lf>
```

Where `<cr>` is the carriage return character and `<lf>` is the line feed character. In software coding the carriage return is commonly written as `'\r'` and the line feed is written as `'\n'`.

Client to GPI Interface

There are four messages sent from the CasparCG client to the GPI interface:

Byte 1	Byte 2	Byte 3	Byte 4	Message content
'i'	'?'	<cr>	<lf>	Return the number of GPI inputs
'o'	'?'	<cr>	<lf>	Return the number of GPI outputs
'a'	'?'	<cr>	<lf>	Are you alive?
'0'...'7'	'0' | '1'	<cr>	<lf>	Set output port number in byte 1 to the state in byte 2

GPI Interface to Client

There are four messages sent from the GPI interface to the client:

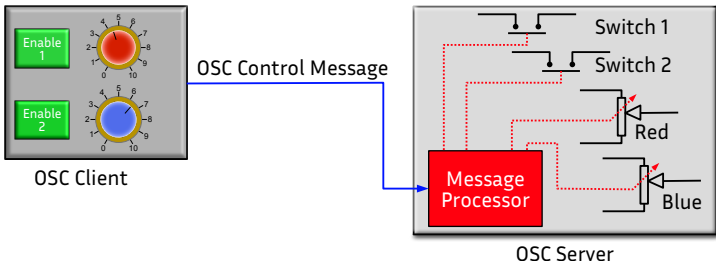
Byte 1	Byte 2	Byte 3	Byte 4	Message Content
'i'	'0' + Num_GPI_In	<cr>	<lf>	Response to Query from client. Returns the number of GPI inputs.
'o'	'0' + Num_GPI_Out	<cr>	<lf>	Response to Query from client. Return the number of GPI outputs
'a'	'!'	<cr>	<lf>	Response to "Are you alive?" query.
'0' _ port_no	'0' | '1'	<cr>	<lf>	Report a GPI input value.

Appendix B - Open Sound Control

Open Sound Control (OSC) was created to pass control messages between units of audio equipment. Version 1.0 was released in March 2002, with a subsequent minor update to version 1.1 in 2009. Version 1.1 supports more data tag types, thereby supporting a wider set of information formats. The flexibility of the OSC message package enables the transport of control data for non-audio applications, meaning OSC has developed more into Open System Control.

Full detail of version 1.0 is available via this link, and details of the 2009 version 1.1 changes are available via this link.

Control messages are sent from an **OSC client** to an **OSC server** that actions the control request.



In this example system, the content of the OSC control message identifies the property to adjust and the new value for the property.

The OSC server implementer decides if the switch element control is encoded as a boolean value, or as an integer value. Similarly, the implementor can choose if gain settings controlled by the rotary client controls are encoded as an integer or as a floating point value. The implementor can also choose to support multiple representations of each controlled parameter. The OSC client implementer or system installer require documentation of the address and data structure the server supports.

OSC messages are transport system agnostic. Commonly used connection methods include bit-serial (EIA RS232, EIA RS422 etc), USB, UDP/IP and TCP/IP. No pre-defined IP ports are allocated for OSC control functions, end users select port numbers that are available in their control network.

OSC use in CasparCG

CasparCG uses OSC over UDP transport. OSC is used in Caspar CG for 5 functions:

1. Sending status information from a CasparCG server to registered clients. This data is used to display progress bars on media clips.

2. Sending status information from a CasparCG server to designated OSC servers. This mechanism is used to create status displays showing media names and playout progress etc.

3. Sending event triggers into the CasparCG client from an external OSC client.

4. Sending event triggers from a web page using Web Sockets to trigger events in a CasparCG client.

5. Remote control of the rundown operations in a CasparCG client. This includes selecting the active item, and issuing commands such as load, play and stop.

OSC Message Structure

OSC messages are wrapped into a ***packet***. Each OSC packet contains an **address** that identifies the controlled element, a **type tag** that describes the encoding mechanisms of the control value or values, and an **information** field containing the updated property values.

Sequences of ASCII characters in an OSC message are known as an **OSC string**. The end of the character sequence is identified by a null character ('\0'). The OSC string length must be a multiple of 32-bits, implemented by appending extra null characters when required.

The control address is a variable-length URL-style name, encoded in an OSC string. An example address is **/control/titles/play**. There is a small list of reserved characters that may not be used as part of an address:

(Space character)' ' **# * , / ? [ ] { }**

The **type tag** is an OSC string that identifies the data type or types transported in the packet. This OSC string starts with a comma and is followed by a series of letters identifying the encoding of the parameter values that follow. Commonly used parameters are 32-bit integer values, 32-bit floating point values, OSC strings, OSC blobs (binary data), midi data (4-bytes), logical True and logical False.

The OSC information field has the same number of values delivered in the same order as the type tag identifier bytes in the Type Tag. The table below shows some of common Tag Ids and their encoding.

Tag ID	Data Type	Encoding
i	int32	32-bit big endian two's complement integer
f	float32	32-bit big-endian IEEE 754 floating point number
s	OSC-string	A sequence of non-null ASCII characters followed by a null, followed by 0-3 additional null characters to make the total bit length a multiple of 32.
b	OSC-blob	An int32 size count followed by that many 8-bit bytes of arbitrary binary data, followed by 0-3 additional zero bytes to make the total number of bits a multiple of 32.
m	midi	4 byte MIDI message. Bytes from MSB to LSB are: port id, status byte, data1, data2
T	boolean	No data encoded for a boolean item
F	boolean	No data encoded for a boolean item.

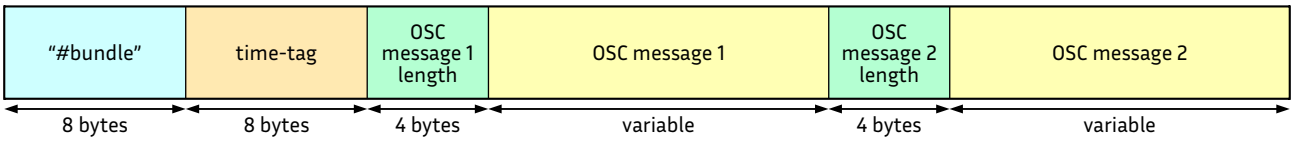
Example Encoded Message

The example encoding has an address of **/myaddr/level** with one **integer** parameter with a value of **8**.

Byte No	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
ASCII	/	m	y	a	d	d	r	/	l	e	v	e	l	'\0'	'\0'	'\0'	,	i	'\0'	'\0'				
Hex	2f	6d	79	61	64	64	72	2f	6c	65	76	45	6c	00	00	00	2c	69	00	00	00	00	00	08

OSC Bundles

Some control events require several commands sent in the minimum possible time. The OSC bundle mechanism is an outer wrapper that combines multiple OSC packets into a single transmission block. The structure of a bundle is shown below:



A bundle is identified by an initial OSC string of **#bundle**. The bundle ident is followed by an eight byte time tag. The upper 32-bits are the number of seconds since midnight on January 1st 1900. The lower 32-bits are the fractional parts of a second to a precision of about 200 picoseconds. This is the same time format used in NTP time messages. The encoded value is the instant the commands in the bundle should be actioned by the server. There is a special time value comprising 63 zero-bits followed by a 1 which has the meaning ***immediately***.

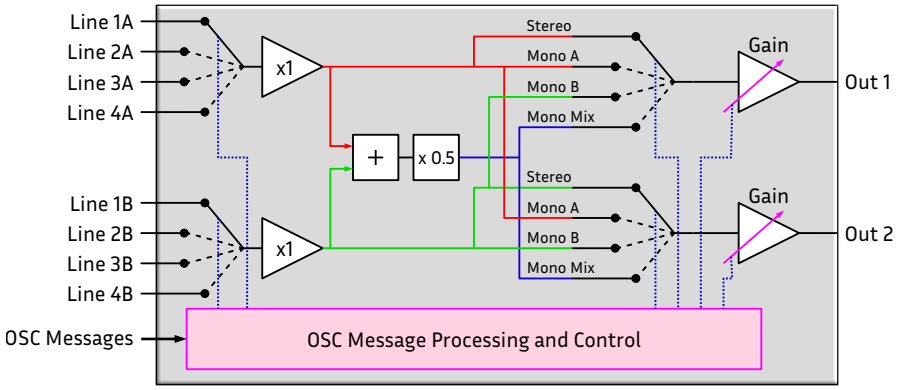
The ident and time tag fields are followed by any number of OSC messages, each message preceded by the length of the message in bytes. The OSC message may be a simple message or another bundle.

The CasparCG server contains an OSC client that uses OSC bundles to output status reports. The SVT CasparCG client can send OSC messages to other equipment. The default wrapping uses a simple message structure, but a tick box enables the encode to be changed to a bundle.

OSC Address Schemes

The OSC address scheme is highly adaptable. The address scheme is very similar to the one used for web browsing, starting with just a single **/** rather than **http://**. The address scheme uses a **/root/branch/twig/leaf** structure. The final element in the address is the controlled target, known as the **OSC Method**.

The diagram below shows a hypothetical audio amplifier with OSC remote control of some facilities.



The amplifier block has two input chains, notionally for left and right channels.

There are two switches in each chain, one selects the input source and one the input mode. The available modes are stereo, mono from left, mono from right or a 50-50 mix of left and right. The mode switches of each chain are independently controlled, enabling swapping of left and right sources if required. The switch positions and output gain of each chain is controllable via OSC messages.

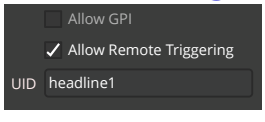
A possible address scheme starts with a root of **/amp** that divides into two sub-addresses of **/amp/left** and **/amp/right**. The controllable hardware, the OSC methods, are named as **input**, **mode** and **gain**, giving an address set for the left channel of:

/amp/left/input	use an integer parameter for the switch position.
/amp/left/mode	use an integer parameter to select the mode.
/amp/left/gain	use a floating point value.

OSC clients can send address strings that include wild characters, such as **?** and *****, used in address comparisons. A question mark matches a single character, and an asterisk matches a string of zero or more characters. An OSC server with two addresses /myunit/output/1/gain and /myunit/output/2/gain can be sent a message directed at /myunit/output/?/gain to set the same gain for both nodes.

There is also a special character code of **//** which acts a shortcut to a common element of a control address, enabling the shortened address of the form **//gain** or **//left/mode**.

OSC message Triggers in CasparCG Client



The SVT CasparCG client includes an OSC server that enables an external OSC client to trigger items in the active rundown. The screen grab at the left illustrates the entries in a rundown item that enable OSC triggering. The string entered in the UID box forms part of the trigger address.

The triggered OSC address is of the form **/control/<UID_name>/<action>** where <UID_name> is the target name entered in the UID box and <action> is the control process required. For example:

/control/headline1/play

The supported OSC <action> names are:

stop	play	playnow	load
pause	next	update	invoke
preview	clear	clearvideolayer	clearchannel

The user selected <UID_name> must follow general OSC rule (no spaces, none of the reserved characters). The OSC message must include a single integer parameter with value 1 to be recognised as a valid control string.

The OSC commands can be sent direct via UDP to the client or they can be sent as WebSocket JSON traffic.

CasparCG Client OSC Rundown Controls

The SVT client rundown can also be controlled by OSC messages. This control mechanism provides remote operation of the rundown as if the user were directly controlling the rundown playout from a keyboard. All the commands use an OSC address of the form **/control/<action>** where <action> is one of the entries in the table below.

stop	play	playnow	load
pause	next	update	invoke
preview	clear	clearvideolayer	clearchannel
down	up	playnowifchannel	playandautostep
playnowandautostep			

The majority of the control values are self-documenting. One, **playnowifchannel**, is less obvious. This value causes the SVT client to look at the item properties and to only issue the play action if the rundown item is a CasparCG control. If the playnowifchannel is actioned by a group, any command such as ATEM, TriCaster or GPI output are not triggered.

Using Web Sockets for OSC Control from a Browser

It is possible to implement simple control operations from a web page. The html code below is a basic implementation. It has one button that plays a remote control enabled clip with a control UID of "1". It has four buttons that can move the cursor up and down the active rundown, play the selected item, or play the selected item and move the cursor down to highlight the following item. These four buttons operate if the rundown is remote control enabled (via Remote control enable item in the rundown menu).

```
<!DOCTYPE html>
<html lang="en">
<head>
      <meta charset="UTF-8">
      <meta name="viewport" content="width=device-width, initial-scale=1.0">
      <title>Simple CasparCG Websocket control client</title>
</head>

<script>
// CCG control from a web page. Remember to enable the rundown Allow Remote
// Trigger to allow message cursor move processing by the rundown.

var ws
function onReady(){
      // Open the websocket connection to the CasparCG server port 4250 once
      // the browser has loaded all page elements. The tcp/ip address
      // is the client host pc address.
      ws = new WebSocket("ws://192.168.42.12:4250/");
}
```

```
function playClip1(){
      // This is the trigger for an item with UID of 1
      ws.send('{"path" : "/control/1/play", "args" : [1]}');
}

function doup(){
      ws.send('{"path" : "/control/up", "args" : [1]}');
}

function dodown(){
      ws.send('{"path" : "/control/down", "args" : [1]}');
}

function doplay(){
      ws.send('{"path" : "/control/play", "args" : [1]}');
}

function doplayandadvance(){
      ws.send('{"path" : "/control/playandautostep", "args" : [1]}');
}

</script>
<body onload="onReady()">
      <input id="Titles" type="button" value="slide 1" onclick="playClip1();"/><br>
      <input id="UP" type="button" value="Cursor Up" onclick="doup()"/>
      <input id="DOWN" type="button" value="Cursor Down" onclick="dodown();" />
      <input id="TAKE" type="button" value="Play at Cursor" onclick="doplay();" />
      <input id="TAKEAUTO" type="button" value="Play and Advance" onclick="doplayandadvance();"/>
</body>
</html>
```

CasparCG Server OSC Data Output

CasparCG server uses OSC to emit the current server status to connected clients and addresses defined in the server configuration file. Because this status data includes details of all active channels, the audio levels etc the OSC bundle is used to combine several status items into a single network transmission. The data block below was captured using Wireshark exporting the UDP packet as both a hex dump and ASCII..

```
0000   00 e0 4c 36 57 79 1c 69 7a 02 fd 0a 08 00 45 00   ..L6Wy.iz.....E.
0010   00 e4 e0 7b 00 00 80 11 84 25 c0 a8 2a 0b c0 a8   ...{.....%..*...
0020   2a 0c e8 f0 18 6a 00 d0 7a 1e 23 62 75 6e 64 6c   *....j..z.#bundl
0030   65 00 00 00 00 00 00 00 00 01 00 00 00 44 2f 64   e............D/d
0040   69 61 67 2f 38 2f 74 65 78 74 00 00 00 00 2c 73   iag/8/text....,s
0050   00 00 66 66 6d 70 65 67 5b 48 61 72 62 6f 75 72   ..ffmpeg[Harbour
0060   5f 43 6c 69 70 2e 6d 6f 76 7c 37 32 30 78 35 37   _Clip.mov|720x57
0070   36 69 35 30 2e 30 30 7c 36 39 36 2f 31 35 31 32   6i50.00|696/1512
0080   5d 00 00 00 00 34 2f 63 68 61 6e 6e 65 6c 2f 31   ]....4/channel/1
0090   2f 73 74 61 67 65 2f 6c 61 79 65 72 2f 31 30 2f   /stage/layer/10/
00a0   70 72 6f 64 75 63 65 72 2f 74 79 70 65 00 2c 73   producer/type.,s
00b0   00 00 66 66 6d 70 65 67 00 00 00 00 00 34 2f 63   ..ffmpeg.....4/c
00c0   68 61 6e 6e 65 6c 2f 31 2f 73 74 61 67 65 2f 6c   hannel/1/stage/l
00d0   61 79 65 72 2f 31 30 2f 70 72 6f 66 69 6c 65 72   ayer/10/profiler
00e0   2f 74 69 6d 65 00 2c 66 66 00 00 00 00 00 3c a3   /time.,ff.....<.
00f0   d7 0a                                             ..
```

The start of the bundle is shown by the **blue** text. The bundle header is followed by the **time** at which the command should be actioned, in this instance immediately. The first OSC message length is 44 (hex) bytes long. The address field of the first message is **/diag/8/text** which has one null character string terminate bye and 3 padding bytes to make the string a multiple of 4-bytes long.

The value carried in the message is a string, with the string text being:

```
ffmpeg[Harbour_Clip.mov|720x576i50.00|696/1512]
```

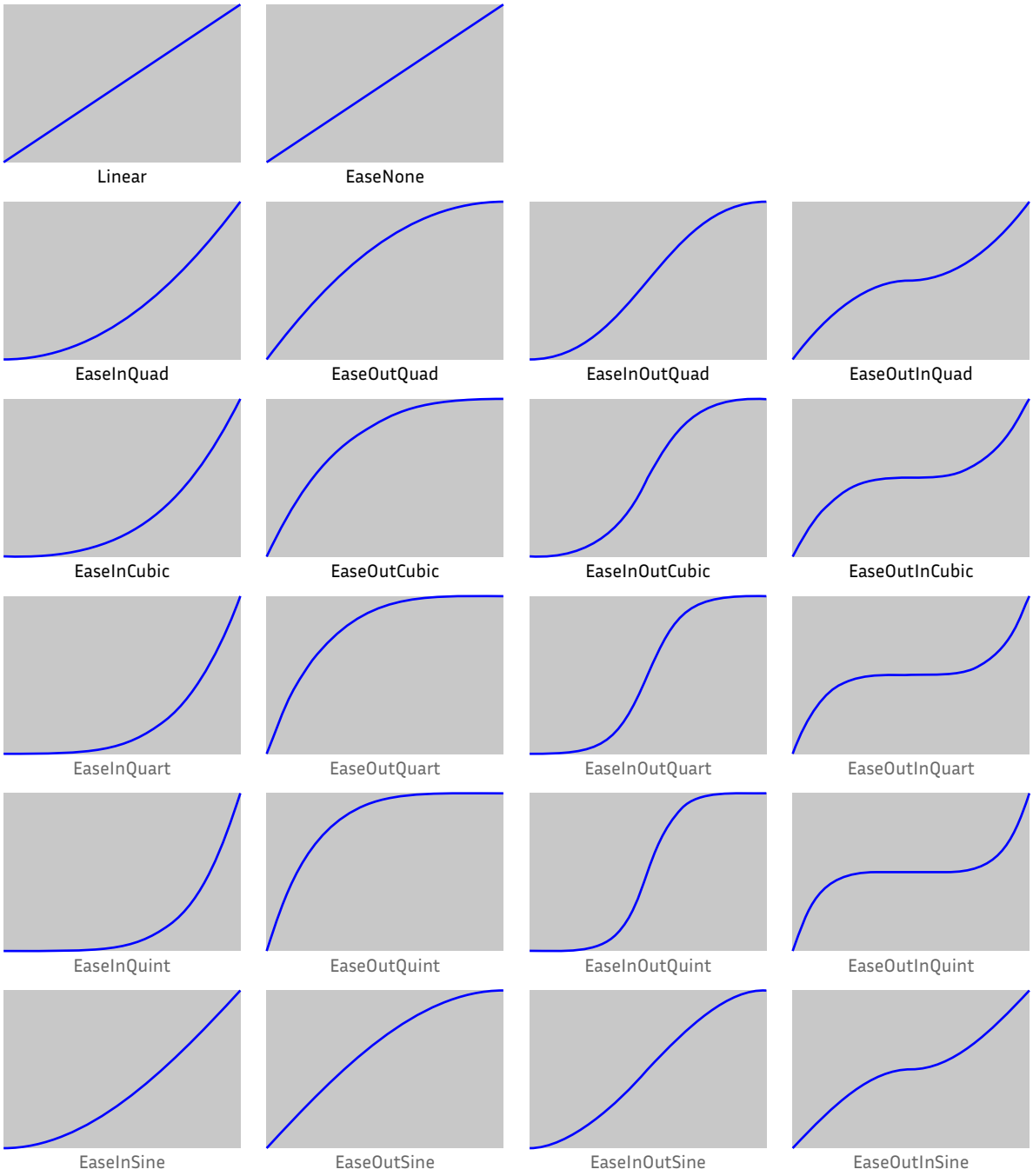
A client can extract the clip name and the playout progress which is frame 696 of 1512 frames total. The timebase is present in the message allowing a computation of the time remaining in the playout.

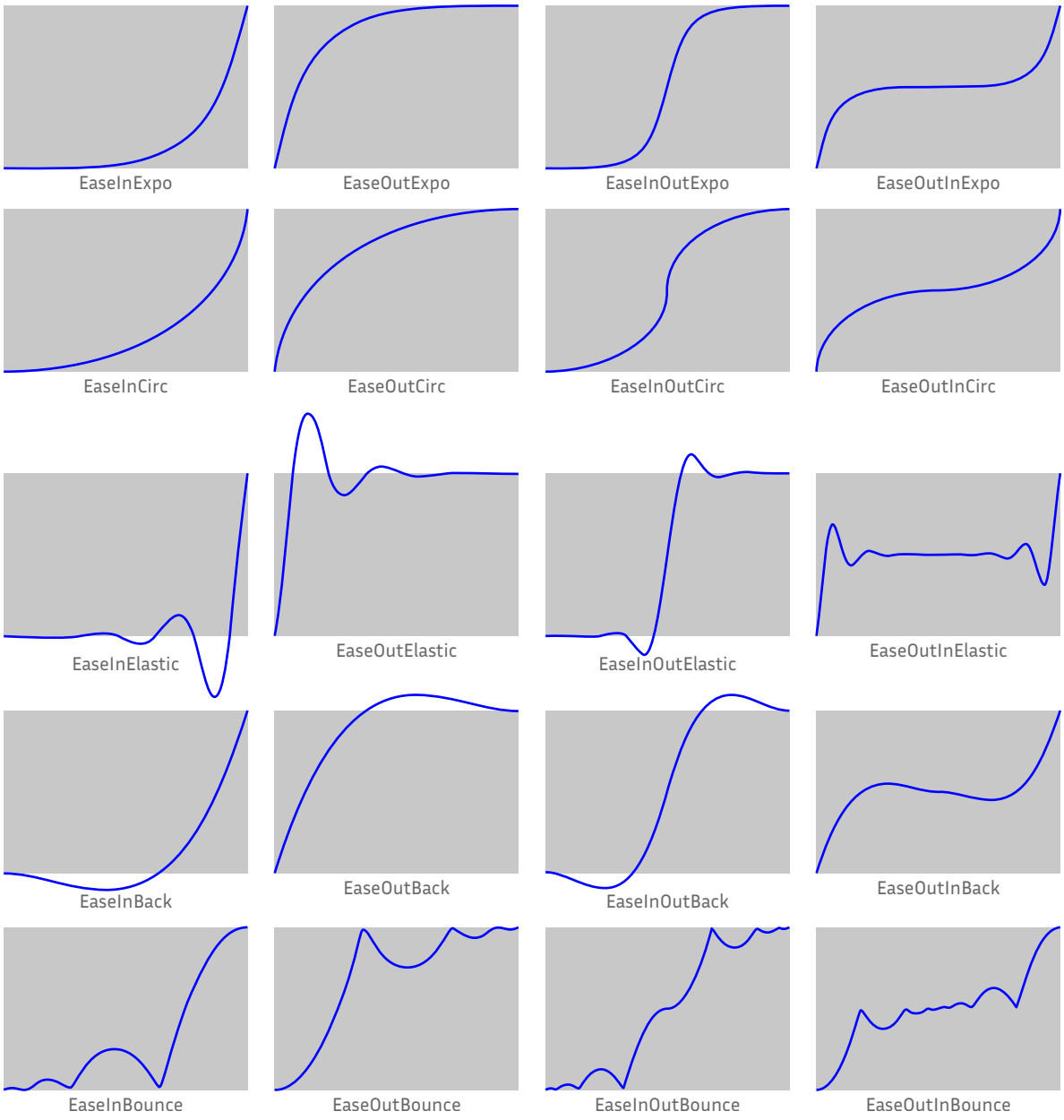
Appendix C - Transition Easing

CasparCG implements a standard set of transition easing functions published by Robert Penner, originally for use in flash animations. For more detail, including the equations, see http://robertpenner.com/easing/.

In the following graphs the vertical axis is the value change for the modified parameter (mix percentage, position etc). The horizontal axis is effect time expressed as a percentage (0% to 100%) of the user-set physical transition time.

The Penner equation implementations generally have an EaseNone function rather than the more descriptive Linear function. Linear combination is widely used in TV equipment for mixes etc. CasparCG server has added code that maps a request for a Linear transition to the EaseNone transition (actually called Linear by Penner!).




EaseInExpo
EaseOutExpo
EaseInOutExpo
EaseOutInExpo
EaseInCirc
EaseOutCirc
EaseInOutCirc
EaseOutInCirc
EaseInElastic
EaseOutElastic
EaseInOutElastic
EaseOutInElastic
EaseInBack
EaseOutBack
EaseInOutBack
EaseOutInBack
EaseInBounce
EaseOutBounce
EaseInOutBounce
EaseOutInBounce

Appendix D - Software Versions

There are several branches and versions of CasparCG server. At April 2021 the two most relevant versions are the outlined below.

Server Version 2.3.3_LTS (Long Term Support)

This branch is managed by the development team at SVT. This server computer for this version requires a graphics card that supports OpenGL version 4.5 or higher. The ffmpeg library and the OSC status outputs require a client version 2.2 to show clip playout progress.

The design targets in this branch are operational stability with a support life of more than two years from the initial release in mid-2020.

This version provides native support for Newtek NDI input and output streams via the NDI producer and NDI consumer.

There have been three incremental releases to the initial version 2.3.0 release:

Version 2.3.1 - January 2021

Enables Flash graphic templates to operate after Windows removed Flash support.

Version 2.3.2 - March 2021

Bug fixes -Intel Threading Library and ffmpeg producers.

Version 2.3.3 - March 2021

Added Image Scroll producer that was removed during early development of the LTS code base. This allows simple stills images to create roller and crawler captions.

Software Repository:

https://github.com/CasparCG/Server/releases

Server Version 2.1.12_NRK

This NRK supported fork of the server requires a graphics card that supports OpenGL version 3.0 or higher. The matching clients are versions 2.0.8 and 2.0.9 (which shows V2.0.8 on the About... display, but the changelog file shows the 2.0.9 developments).

A key target of this branch is stability of operation as it is deployed in live news playout operations. It has the ability to trigger server playout events based on EBU/SMPTE timecode connected to the physical input/output cards, or via the use of the computer's internal clock.

This code branch can create NDI output streams, labelled iVGA, an earlier name for NDI. This code branch also supports some templated operations using Adobe Photoshop files with the Adobe timeline.

Recording and Streaming Issue

There is a known bug in the **Windows** version this branch that causes the server to crash when the file producer or ffmpeg producer is removed from an output. The development team have used significant effort to locate the cause, but issue remains unsolved. There are two consequences:

- It is not possible to create recordings of a channel
- It is not possible to stream a live client preview

The development team report that switching the Microsoft VS2015 linker to /Incremental mode removes the issue, but there is a consequential performance loss. The SVT LTS version seems to have identical code base, but is built using a newer Microsoft compiler/linker. There is significant risk that porting the code to a newer Microsoft version of VS will be take a long time and suffer many issues as various libraries will also need updating. The bug report for the issue is available here.

The Linux version of the server records as expected.

Software Repository:
https://github.com/nrkno/tv-automation-casparcg-server/releases

Appendix E - Swapping between Software Versions

Server Version

The elements in the casparcg.config file is described earlier in this document. The major differences between the configuration files for the NRK and SVT branches are:

- There are less folder definitions in the <paths> (<font> path not required)
- Definitions are required for the media scanner
- The default audio mode is stereo in the NRK version and 8-channel passthrough in the SVT version.

The other key difference between the servers is the client requirement - version 2.0.8/2.0.9 for the NRK branch, and version 2.2.0 for the SVT version. Where time entries in frames are used, such as fade durations, the values are 25Hz timebase for version 2.0.9 and 50Hz timebase for 2.2.0.

A key change in the SVT 2.3 branch is splitting the media scanner and thumbnail generation out from the server code and using a new support tool to provide the services. Clients still request the information via AMCP commands directed to the server, but the server requests the data from the media scanner tool. The SVT distribution includes a batch file that runs the scanner, then runs the server. Whilst the two elements can be manually started in sequence the batch file is the better process to use. The NRK branch also has a batch file, but the only thing it runs is the server.

The server command line can include the name of the configuration file to use, allowing easy support where different channel configurations are required - for example one for SD-SDI and one for HD-SDI operations. The provided batch files do not provide support for this, although it could be added to the NRK version, but the media scanner used in the SVT server does not support a command line change of configuration file name to use.

The simplest process to manage both code branch configurations is keeping the required configuration files in their server folder, but with an added name element. For example, for HD and SD operations have two files called casparcg_SD.config and casparcg_HD.config.

Create one or more batch files whose function is copying the desired configuration file into the standard casparcg.config file name.

Keep shortcuts to the server startup batch files on the computer desktop as well as in the main folder of the server code. Ensure all users know that the system must ***only*** be started via the batch files.

The system installer chooses the folder that holds the server LOG files by an entry in the <paths></paths> tag pair in the server configuration. The log files for the different server versions can use a common folder. Using a single folder simplifies the log management process (see *here* for more detail).

Client Version

The default client configuration uses a single folder that has the client configuration database file, with a sub-folder for client log files. A Windows client host uses a folder called .CasparCG that is stored in the logged on users home folder on the primary disk drive. If the user is called casparcg the full path to the configuration and log store is:

C:\Users\casparcg\.CasparCG

There is a sub-folder called Client which has the database called **Database.s3db** and a folder called **Logs**.

The operational issue is that the fields in the database file has different content for client version 2.0.9 and client version 2.2. Operating more than one client therefore needs tight operational discipline.

The client program checks it's startup command line looking for any options that may be present. One option is to choose a different database file name or path. The simplest mechanism to implement this change of file name is to use a shortcut with the command line of the format:

```
"C:/CasparCG/ClientV2d0d9/casparCG Client.exe" -t c:/casparclient/usedb209.s3db
```

where -**t** is the option tag for client database name.

If someone now directly runs a client EXE application the database uses the standard location and name, and does not corrupt the normal working file for a specific client.

Good operational practice requires that after a client is fully configured a copy of the configuration is taken and stored in a "safe" location. If there is a subsequent problem with the working copy the safety copy is used to restore basic configuration. It may still be necessary to use the client configuration tool to delete thumbnail pictures from the database, then allow the thumbnails for the current media set to be re-loaded.

An alternate method of managing the use of two clients is making a manual copy of each client configuration database after every configuration change, then copying that file to the default name and location before starting the client.

Typeset in Inria Serif and Inria Sans using Affinity Publisher with drawings created in Affinity Designer.
Document created and typeset by Andy Woodhouse (andy@amwtech.co.uk)
Version 1.0
Print Date and Time: 2021-07-20 12:51