

GPI SCRIPTING FOR KEYERS

1. GPI SCRIPTING

GPI script files are text-based files that can be programmed and sent to the script-enabled unit via Overture. The syntax is important as the script represents programming code that will be executed when the allocated GPI trigger event occurs. Each keyer, media inserter, and switcher has a number of GPI inputs. Each input has 2 events: *close* and *open*. The user can program scripts for each of these 16 events. If a script file is present on the flash file system the unit will process the script when the event is triggered. If the script file is not present when a GPI event is triggered, then the internal GPI menu settings will be used.



Note: The script files are stored on the compact flash of the Logo and Media Inserters and Keyers. Thus, the scripts are retained over any power cycling of the units.

The GPI script files are text files, therefore the user can easily edit them using existing tools such as Windows Notepad.

The title of a script is used by the system to determine which GPI event the file is applicable to.

For example: **gpi-h-close** (script is called when GPI H is closed) and **gpi-a-open** (script is called when GPI H is open)

In this case, *gpi* states that the script will be running off the state of a GPI trigger. The *h* represents which specific GPI trigger the script will react to. GPI triggers range from *A* to *H* (for 9625 and 7725 series) or *0* to *31* (for 9725 series) or *0* to *7* (for QMG series).

1.1. GENERAL SCRIPT SYNTAX

In the programming language the script utilizes, there are three types of syntax used: comments, commands, and objects.

A *comment* is represented by the symbol **#**. Any text that follows this symbol on a line of code will not execute a command or conduct any sort of action. This is simply used to add user comments within the script. For example:

```
# this file describes the complete state of the keyer
```

A *command* (**cmd**) will perform an appropriate function of the media/logo inserter or switcher product. For example:

```
cmd media_in("logo1")
```

In the above example, the `media_in` command will put `logo1` on-air.

An *object* is used to set up various configuration items on the media/logo inserter and switchers. For example:

```
object transition()  
{  
    type = "transition type"  
    rate = "integer number of fields"  
    swap = "yes" (or "true"), or "no" (or "false")  
}
```

The above example shows how the transition object will set up a transition on a 9625SW or HD9625SW. Please refer to section 1.4 for more details on object definition and supported objects.

1.2. SUPPORTED COMMANDS FOR QMG, 9625, AND 9725 SERIES LOGO AND MEDIA INSERTERS

There are many different kinds of commands that can be run. Each command will perform a specific action when the script is run. The following is a list of commands:

- cmd media_all_out():** This command fades out all logos and audio clips.
- cmd media_cue("02"):** This command will cue up logo "02".
- cmd media_in("03") :** This command will fade in logo "03".
- cmd media_out("04"):** This command will fade out logo "04".
- cmd media_toggle("05"):** This command will toggle the state of logo "05." If the logo is faded out it will be faded in, if it is faded in then it will be faded out.
- cmd udt_start(1) :** This command will start the up/down timer #1.
- cmd udt_stop(2):** This command will stop the up/down timer #2.
- cmd udt_toggle(1):** This command will toggle the start/stop state of the up/down timer #1. If the timer is started it will stop, if stopped it will start.
- cmd udt_reload(2):** This command will reset up/down timer #2 to the timer start time.
- cmd voiceover_enable():** This command will enable the voiceover function.
- cmd voiceover_disable():** This command will disable the voiceover function.
- cmd voiceover_toggle():** This command will toggle the state of the voiceover function. If the state is enabled it will be disabled, if disabled it will be enabled.

1.3. SUPPORTED COMMANDS FOR HD9625SW, 9625SW, 7725DSK-LG AND 7725DSK-LG-HD

There are many different kinds of commands that can be run. Each command will perform a specific action when the script is run. The following is a list of commands:

cmd load_preset(1):	This command will load preset 1 (not implemented).
cmd udt_start(1) :	This command will start up/down timer #1.
cmd media_all_out() :	This command fades out all logos and audio clips.
cmd media_cue("02"):	This command will cue up logo "02".
cmd media_in("03") :	This command will fade in logo "03".
cmd media_out("04"):	This command will fade out logo "04".
cmd media_toggle("05"):	This command will toggle the state of logo "05." If the logo is faded out it will be faded in, if it is faded in then it will be faded out.
cmd udt_stop(2):	This command will stop the up/down timer #2.
cmd udt_toggle(1):	This command will toggle the start/stop state of the up/down timer #1. If the timer is started it will stop, if stopped it will start.
cmd udt_reload(2):	This command will reset up/down timer #2 to the timer start time.
cmd voiceover_enable():	This command will enable the voiceover function.
cmd voiceover_disable():	This command will disable the voiceover function.
cmd voiceover_toggle():	This command will toggle the state of the voiceover function. If the state is enabled it will be disabled, if disabled it will be enabled.
cmd transition("video"):	This command will enable a video transition setting the preview bus to the program bus using the transition settings specified either in the script or from the panel.
cmd transition("audio"):	This command will enable an audio transition from the device.
cmd transition ("key,audio"):	This command will enable a key and audio transition from the device.
cmd transition ("bg,audio"):	This command will enable a background and audio transition from the device.

1.4. DESCRIPTION OF OBJECTS

An *object* is used to set up various configuration parameters on the media/logo inserters and switchers. These parameters can be set up before a **cmd** is sent.

1.4.1. OBJECT FORMAT

An *object* has a particular set up. The format of the object is:

```
object TYPE (name) {  
    PROPERTY = "value"  
    PROPERTY = "value"  
    . . .  
}
```

The TYPE is the parameter of interest (such as transition). The PROPERTY is considered sub-parameters of the TYPE. Some TYPEs may have multiple properties.

The use of **obj** can be used instead of **object** when configuring the object.



The opening brace "{" and closing brace "}" for an object definition MUST always be present. An absence of a closing brace "}" will cause issues with the script and its execution.

1.5. AVAILABLE OBJECT TYPE

The current defined TYPEs are:

- **bus_setup**
- **channel**
- **router**
- **transition**
- **matte**
- **key**
- **audio_over**
- **misc**
- **key_enable**
- **voiceover_trans**

1.5.1. *bus_setup* Object

The **bus_setup** object will select the input bus for audio configuration of the media inserter, keyer or switcher.

```
object bus_setup ("bus names") {  
    object channel ("channel names") {  
    }  
}
```

The "bus names" have different values for different products. Please see below for the appropriate values for "bus names".

bus names for 9625SW or HD9625SW

"black", "white", "1", "2", "3", "4", "5", "6", "7", "8", "9", "10", "11", "12"

bus names for 7725DSK-LG or 7725DSK-LG-HD

"1", "2"

bus names for 9625DSK-LGA

"black", "bkg" or "a", "fill" or "b"

bus names for QMG, 9625LGA, HD9625LGA, 9725LGA and HD9725LGA

"black", "a"

1.5.1.1. *channel* Sub-Object

The **channel** sub-object for the *bus_setup* object will set up the input audio channels (for a selected bus) of the media inserter, keyer or switcher.

```
object channel ("channel names") {  
}
```

The appropriate values for "channel names".

channel names

"1L", "1R", "2L", "2R", "3L", "3R", "4L", "4R"

The channel object has a number of properties. These properties are:

- source = "audio_source_name"
- gain = "gain_db"
- router = "1", ..., "12", "none"
- nonpcm = "1" or "0"

For “audio_source_name” there are different choices based on platforms.

audio_source_names for 9625SW or HD9625SW

AES1L, AES1R, AES2L, AES2R, AES3L, AES3R, AES4L, AES4R, AES1M, AES2M, AES3M, AES4M, EMB1L, EMB1R, EMB2L, EMB2R, EMB3L, EMB3R, EMB4L, EMB4R, EMB1M, EMB2M, EMB3M, EMB4M, Silence

audio_source_names for 7725DSK-LG or 7725DSK-LG-HD

AEMB1L, AEMB1R, AEMB2L, AEMB2R, AEMB3L, AEMB3R, AEMB4L, AEMB4R, AEMB1M, AEMB2M, AEMB3M, AEMB4M, BEMB1L, BEMB1R, BEMB2L, BEMB2R, BEMB3L, BEMB3R, BEMB4L, BEMB4R, BEMB1M, BEMB2M, BEMB3M, BEMB4M, Silence

audio_source_names for 9625DSK-LGA

AES1L, AES1R, AES2L, AES2R, AES3L, AES3R, AES4L, AES4R, AES5L, AES5R, AES6L, AES6R, AES7L, AES7R, AES8L, AES8R, AES1M, AES2M, AES3M, AES4M, AES5M, AES6M, AES7M, AES8M, AEMB1L, AEMB1R, AEMB2L, AEMB2R, AEMB3L, AEMB3R, AEMB4L, AEMB4R, AEMB1M, AEMB2M, AEMB3M, AEMB4M, BEMB1L, BEMB1R, BEMB2L, BEMB2R, BEMB3L, BEMB3R, BEMB4L, BEMB4R, BEMB1M, BEMB2M, BEMB3M, BEMB4M, Silence

audio_source_names for QMG, 9625LGA, HD9625LGA, 9725LGA and HD9725LGA

AES1L, AES1R, AES2L, AES2R, AES3L, AES3R, AES4L, AES4R, AES5L, AES5R, AES6L, AES6R, AES7L, AES7R, AES8L, AES8R, AES1M, AES2M, AES3M, AES4M, AES5M, AES6M, AES7M, AES8M, AEMB1L, AEMB1R, AEMB2L, AEMB2R, AEMB3L, AEMB3R, AEMB4L, AEMB4R, AEMB1M, AEMB2M, AEMB3M, AEMB4M, Silence

1.5.2. *router* Object

The **router** object will select the sources for the internal buses of the media inserter, keyer or switcher.

```
object router() {  
}
```

The **router** object has a number of properties. These properties are:

- pgm_source = “bus_name”
- pvw_source = “bus_name”
- key_source = “bus_name”
- fill_source = “bus_name”

The “bus names” have different values for different products. Please see below for the appropriate values for “bus names”.

bus names for 9625SW, HD9625SW, 7725DSK-LG, and 7725DSK-LG-HD
“black”, “white”, “1”, “2”, “3”, “4”, “5”, “6”, “7”, “8”, “9”, “10”, “11”, “12”

bus names for 9625DSK-LGA
“black”, “bkg” or “a”, “fill” or “b”

bus names for QMG,. 9625LGA, HD9625LGA, 9725LGA and HD9725LGA
“black”, “a”

1.5.3. *transition* Object

The **transition** object will configure the transitions of the media inserter, keyer or switcher.

```
object transition() {  
}
```

The **transition** object has a number of properties. These properties are:

- type = “transition type”
- rate = “integer number of fields”
- swap = “1” or “0” , where “1” will enable the swap

The “transition type” has different values for different products. Please see below for the appropriate values for “transition type”.

transition type for 9625SW, HD9625SW, 7725DSK-LG, and 7725DSK-LG-HD
Cut, Fade, BarWipeTopToBottom, DiagonalWipeTopLeft, BarWipeLeftToRight,
DiagonalWipeBottomLeft, BarWipeBottomToTop, DiagonalWipeBottomRight,
BarWipeRightToLeft, DiagonalWipeTopRight, BoxWipeBottomLeft,
BoxWipeBottomRight, BarnDoorWipeVerticalClose, BarnDoorWipeHorizontalClose,
BarnDoorWipeVerticalOpen, BarnDoorWipeHorizontalOpen, IrisWipeRectangleClose,
IrisWipeRectangleOpen, IrisWipeCircleClose, IrisWipeCircleOpen,
IrisWipeDiamondClose, IrisWipeDiamondOpen, CutFade, FadeFade, FadeCut

transition type for 9625DSK-LGA
Cut, Fade, BarWipeTopToBottom, DiagonalWipeTopLeft, BarWipeLeftToRight,
DiagonalWipeBottomLeft, BarWipeBottomToTop, DiagonalWipeBottomRight,
BarWipeRightToLeft, DiagonalWipeTopRight, BoxWipeBottomLeft,
BoxWipeBottomRight,

1.5.4. *matte* Object

The **matte** object will configure the matte feature of the media inserter, keyer or switcher.

```
object matte() {  
}
```

The **matte** object has a number of properties. These properties are:

- enable = “1” or “0”, where 1 enables the Matte function
- top = “integer number of lines”
- bottom = “integer number of lines”

1.5.5. *key* Object

The **key** object will configure the keyer feature of the media inserter, keyer or switcher.

```
object key() {  
}
```

The **key** object has a number of properties. These properties are:

- mode = “input” or “self”
- offset = “integer offset”
- threshold = “integer threshold”

1.5.6. *audio_over* Object

The **audio_over** object will configure audio over behaviour for the media inserters, keyer or switcher.

```
object audio_over (“layer names”) {  
    object channel (“channel names”) {  
    }  
}
```

The “layer names” have different values for different products. Please see below for the appropriate values for “layer names”.

layer names for 9625DSK-LGA, 9625SW, HD9625SW, 7725DSK-LG, and 7725DSK-LG-HD
“audio_keyer”, “audio_clip”, “voiceover”

layer names for QMG, 9625LGA, HD9625LGA, 9725LGA and HD9725LGA
“audio_clip”, “voiceover”

The *channel* sub-object for the *audio_over* has a number of properties for each channel name. For channel names:

“1L”, “1R”, “2L”, “2R”, “3L”, “3R”, “4L”, “4R”

The following properties can be set:

- *source* = “*audio_source_name*” or “*audioclip names*”. For “*audio_source_name*” see section 1.5.1.1. For “*audioclip names*”, the choices are “CLIP1L”, “CLIP1R”, “CLIP2L”, and “CLIP2R”
- *gain* = “*gain_db*”, set gain (in dB)
- *duck* = “*duck_db*” set duck value of program audio (in dB)
- *router* = “1”, ..., “12”, “none” (used for 9625SW or HD9625SW only)
- *nonpcm* = “1” or “0”

1.5.7. *misc* Object

The **misc** object will configure the miscellaneous features of the media inserter, keyer or switcher.

```
object misc() {  
}
```

The **misc** object has a number of properties. These properties are:

- *line21_protect* = “1” or “0”, where 1 enables the Line 21 protect function
- *b_blanking* = “1” or “0”, where 1 enables the blanking function

1.5.8. *key_enable* Object

The **key_enable** object will enable the DSK and Logo layer to be On or Off for the switcher. Used for 9625DSK-LGA, 9625SW, HD9625SW, 7725DSK-LG, and 7725DSK-LG-HD.

```
object key_enable(“bus name”) {  
}
```

The “bus name” can be “program” or “preview.”

The **key_enable** object has a number of properties. These properties are:

- *keyer* = “yes” or “no”
- *media* = “yes” or “no”

1.5.9. *voiceover_trans* Object

The **voiceover_trans** object will voiceover parameters of the media inserter, keyer or switcher. This object is typically used by the 9625SW, HD9625SW, 9625DSK-LGA, QMG, 9625LGA, HD9625LGA, 9725LGA and HD9725LGA.

```
object voiceover_trans() {  
}
```

The **voiceover_trans** object has a number of properties. These properties are:

- **vo_in_rate** = “nn number of fields”, where “nn” represents the rate (in fields) that the voiceover transition takes to bring the voiceover to air.
- **vo_out_rate** = “nn number of fields”, where “nn” represents the rate (in fields) that the voiceover transition takes to bring the voiceover off air.

2. SCRIPTING EXAMPLES

2.1. LOGO COMMANDS

To cue a logo on the 7725DSK-LG using a closure of GPI A, the user will create a file called ***gpi-a-closed***. In this file, the commands are:

```
# Cue a logo named "test"  
cmd media_cue("test")
```

To fade in a logo on the 7725DSK-LG using a closure of GPI B, the user will create a file called ***gpi-b-closed***. In this file, the commands are:

```
# Fade in a logo named "rabbit"  
cmd media_in("rabbit")
```

Similarly for a HD9725LGA, to fade in a logo using a closure of GPI 10, the user will create a file called ***gpi-10-closed***. In this file, the commands are:

```
# Fade in a logo named "fox"  
cmd media_in("fox")
```

2.2. BRING UP A TROUBLE SLIDE CALLED "TROUBLE"

These 2 GPI scripts will fade out all displayed logos and stop all audio clips as well as invoke a full screen logo called "trouble" or a trouble slide. The slide must first be prepared in Overture and uploaded to the unit.

For GPI H, under a file named ***gpi-h-close***:

```
# Logo commands  
cmd media_all_out()  
cmd media_in("trouble")
```

For GPI H, under a file named ***gpi-h-open***:

```
# Logo commands  
cmd media_all_out()
```



Note for the 9625DSK-LGA, 9625SW, HD9625SW, 7725DSK-LG and the 7725DSK-LG-HD, the Logo Layer must be enabled for logos and audio clips to play out on the desired output bus. If the user wants to preview an audio clip, ensure that the logo layer for the preview bus is enabled, and disable the logo layer on the program bus of the units.

2.3. INITIATING A TRANSITION

To initiate a transition, on the 7725DSK-LG (or 9625DSK-LGA, 9625SW and HD9625SW) by using a closure of GPI B, the user will create a file called ***gpi-b-closed***. In this file, the commands will be:

```
# Perform a transition named "video"  
cmd transition("video")
```

This script will cause a video transition setting the Preview bus to the Program bus using configured transitions (via control panel or *VistaLINK*®). If the user wants to set up the transition as part of the script, then the following procedure should be followed. For a Diamond open transition of 50 frames on the open of GPI C, in a file called ***gpi-c-open*** the following will be entered:

```
# Transition Type configuration  
  
object transition()  
{  
    type = "IrisWipeDiamondOpen"  
    rate = "50"  
    swap = "no"  
}  
  
cmd transition("video")
```

2.4. SAP PROGRAMMING

SAP Programming means that audio must always be present on audio channel 3 (group 1 Pair 2 channel 1). It requires 2 GPIs for the devices.

In this example for the 9625LGA, the default operation is to have a Mono Mix of channels 1 and 2 and then map it to channel 3.

In a file called ***gpi-a-close***, a closure of GPI A set to simply pass audio on input channel 3 to output channel 3. The file will have the following:

```
# Source Bus A Config  
object bus_setup ("a") {  
# For output channel 3  
object channel ("2L") {  
# Set input to be discrete channel 3  
    source = "AES2L"  
    gain = "0" }  
}
```

In a file called ***gpi-a-open***, when GPI A is open send a Mono-mix of channels 1 and 2, to output channel 3. The file will have the following:

```
# Source Bus A Config
object bus_setup ("a") {
# For output channel 3
object channel ("2L") {
# Set input to be Mono-mix of channels 1 and 2
    source = "AES1M"
    gain = "0" }
}
```

In a file called ***gpi-b-close***, a closure of GPI B set to simply map audio on input channel 4 to output channel 3. The file will have the following:

```
# Source Bus A Config
object bus_setup ("a") {
# For output channel 3
object channel ("2L") {
# Set input to be discrete channel 4
    source = "AES2R"
    gain = "0" }
}
```

In a file called ***gpi-b-open***, when GPI B is open the 9625LGA will revert to a Mono-mix of channels 1 and 2, to output channel 3. The file will have the following:

```
# Source Bus A Config
object bus_setup ("a") {
# For output channel 3
object channel ("2L") {
# Set input to be Mono-mix of channels 1 and 2
    source = "AES1M"
    gain = "0" }
}
```

2.5. EAS FOR AUDIO VOICE OVER

For an EAS voiceover on the HD9725LGA, two GPI scripts will enable and disable the voiceover function mixing the output audio with the configurations set on the HTML voiceover settings. In this example, GPI 12 will be used for EAS voiceovers.

In a file called ***gpi-12-close***, a closure of GPI 12 will enable the voiceover. The file will have the following:

```
# Voiceover Commands
cmd voiceover_enable()
```

In a file called *gpi-12-open*, when GPI 12 is open the EAS voiceover will be disabled. The file will have the following:

```
# Voiceover Commands
cmd voiceover_disable()
```

When the voiceover is disengaged, the audio that was active will still be active as the background source selections are not affected.

2.6. EAS FOR AUDIO VOICEOVER USING 1 OTHER VOICEOVER

Another application is to have a voiceover for EAS and a voiceover for another item. On a HD9725LGA, the user will have four GPI scripts that will enable and disable the voiceover function mixing the output audio with the configurations set from the source selections in the file and then reset the voiceover to the standard inputs. In this example, GPI 9 remaps and invokes the voiceover, and will be connected to the tally out from the EAS decoder, while GPI 10 will be used for normal voiceovers.



Warning, GPI G Open can disable the voiceover function while the EAS voiceover is active

In a file called *gpi-9-close*, a closure of GPI 9 will enable the voiceover using ALT AES IN 4 as source (EAS audio is connected to this AES input). The file will have the following:

```
# Voiceover Commands
cmd voiceover_disable()

# Setup voiceover sources for EAS
object audio_over ("voiceover") {
  # For output channel 1
  object channel ("1L") {
    # Channel 1 from ALT AES 4IN (EAS audio)
    source = "AES8L"
    gain = "0"
    duck = "-6"}

  # For output channel 1
  object channel ("1R") {
    # Channel 2 from ALT AES 4IN (EAS audio)
    source = "AES8R"
    gain = "0"
    duck = "-6"}
}

cmd voiceover_enable()
```

In a file called ***gpi-9-open***, when GPI 9 is open the EAS voiceover will be disabled. The file will have the following:

```
# Voiceover Commands
cmd voiceover_disable()

cmd voiceover_disable()
```

In a file called ***gpi-10-close***, a closure of GPI 10 will enable a normal voiceover using ALT AES IN 1 as source. The file will have the following:

```
# Voiceover Commands
cmd voiceover_disable()

# Setup voiceover sources for EAS
object audio_over ("voiceover") {
  # For output channel 1
  object channel ("1L") {
    # Channel 1 from ALT AES 1IN
    source = "AES5L"
    gain = "0"
    duck = "-10"}

  # For output channel 1
  object channel ("1R") {
    # Channel 2 from ALT AES 1IN
    source = "AES5R"
    gain = "0"
    duck = "-10"}
}

cmd voiceover_enable()
```

In a file called ***gpi-10-open***, when GPI 10 is open the normal voiceover will be disabled. The file will have the following:

```
# Voiceover Commands
cmd voiceover_disable()
```

When the voiceover is disengaged, the audio that was active will still be active as the background source selections are not affected, however the voiceovers will be off.

2.7. AUDIO CONFIGURATIONS FOR QMG, 9625 AND 9725 SERIES MEDIA INSERTERS

GPI scripts can be used to setup audio channels on the QMG, 9625LGA, HD9625LGA, 9725LGA, and HD9725LGA. Users will typically use the HTML page on the devices to configure audio; however, there may be situations where an audio mapping needs to change.

For example, at a facility, the user will typically have all embedded audio pass through the HD9725LGA. In one special case, the user will use the second channel from ALT AES 2 IN on output channel 3. This event will be handled by GPI 1.

In a file called ***gpi-1-close***, a closure of GPI 1 will enable the HD9725LGA to use channel 2 from ALT AES 1 IN as source. The file will have the following:

```
object bus_setup ("a") {
    # For output channel 3
    object channel ("2R") {
        # Channel 2 from ALT AES 2 IN
        source = "AES6R"
        gain = "0"}
}
```

In a file called ***gpi-1-open***, when GPI 1 is open the HD9725LGA will operate in a normal manner where all the embedded audio channels are passed through. The file will have the following:

```
object bus_setup ("a") {
    # For output channel 3
    object channel ("2R") {
        # Channel 3 from embedded input
        source = "AEMB2L"
        gain = "0"}
}
```

2.8. CONFIGURING 7725DSK-LG FOR A FULL TRANSITION WITH KEYER AND LOGO

For the 7725DSK-LG, this example will use a closure on GPI A to perform a *cut* video transition, and will enable the internal keyer and internal logo store onto air. The open state of GPI will disable the internal keyer and logo from program path but will keep it on preview path.

In a file called ***gpi-a-open***, when GPI A is open the 7725DSK-LG will have the internal keyer and logo store on preview path. The file will have the following:

```
object key_enable ("program") {
    keyer = "no"
    media = "no"}

object key_enable ("preview") {
    keyer = "yes"
    media = "yes"}
```

In a file called ***gpi-a-close***, when GPI A is closed the 7725DSK-LG will turn the internal keyer and logo store on air and perform a *cut* transition of the video sources. The file will have the following:

```
object key_enable ("program") {
    keyer = "yes"
    media = "yes"}

object key_enable ("preview") {
    keyer = "no"
    media = "no"}

object router() {
    pgm_source = "1"
    pvw_source = "2"
}

# Transition Type configuration
object transition() {
    type = "Cut" }

cmd transition("video")
```

This page left intentionally blank