

Graphics Actions - управление графическими композициями

Starting from software version 1.6.0 the SL NEO Media Platform supports dedicated graphics layers for Program Channels. The number of graphics layers is configured during Program Channel module configuration in the server's web console. the parameter is called «Graphics composition» as opposed to «Graphics playlists» which are supposed to be controlled by by playlists composed of graphics compositions and/or semitransparent video clips. Note, that the sum of «Graphics compositions» and «Graphics playlists» must be equal or less than the total number of graphics layers that is specified in the «Configure server components» dialog.

Each graphics layer can host one graphics composition at any particular time which, in turn, can contain an arbitrary number of graphics object. Graphics object can be directly controlled by external business logic through a set of commands. The Graphics Composition Editor (AirMgr or NewsCut) assigns a name for every graphics object it creates. These names are used to refer to graphics object when controlling commands are executed. There is only one namespace that is common for all graphics compositions running at the same time in a Program Channel. This means that controlling logic can be decoupled from actual graphics composition implementation and allows to change graphics design and move parts of graphics composition to a different layer with no changes required to the controlling logic.

Graphics compositions are controlled through an XML-RPC/HTTP gateway which sits on the player's port. For each layer the port number is calculated as follows: $port = 4742 + PGM * 16 + PLN + N$ where PGM - is the Program Channel number (zero based), PLN - the number of graphics playlists, N - number of the graphics layer (zero based). The XML-RPC interface is also registered as a set of actions in the system action router, so all the commands can be triggered by play-list, capture or GPI events.

Some commands (such as «play_set_clip») are executed asynchronously and can take some time to complete. If you need to wait for completion of such a command before sending new commands (for example, you may need to wait for a graphics composition to fully load before sending commands to it's object) you can use this method to set a flag into asynchronous execution queue and test for it's completion with «play_test_fence»

Implementation of some logic often requires reaction to some play-list event or actions generated by other services. In the SL NEO Media Platform architecture actions are routed through the action router service. To the receive actions a service needs to register with the action router service.

Поддерживаемые XML-RPC методы

play_set_clip

Method name: "play_set_clip"

Parameter 0 - string - url of the clip containing a graphics composition to be set.

This method can be used to set current graphics composition that is hosted in a graphics layer.

play_get_clip

Method name: "play_get_clip"
Parameters - none
Return value - string - url of the currently loaded clip
This method return url or the currently loaded clip.

play_set_fence

Method name: "play_set_fence"
Parameters - none
Return value - int - flag identifier
This method inserts a completion flag into asynchronous execution queue.

play_test_fence

Method name: "play_test_fence"
Parameter 0 - int - identifier of a completion flag returned by "play_set_fence"
Return value - int - 0 if the flag hasn't been processed,
1 if the flag has been processed and all the previous commands are competed
This methods test a completion flag obtained by "play_set_fence".

do_effect

Method name: "do_effect"
Parameter 0 - int - effect number to perform on the layer. Currently the following effect codes are supported: 0 - toggle layer visibility, 1 - show layer, 2 - hide layer.

graphics_action

Method name: "graphics_action"
Parameter 0 - string - command name.
Parameter 1 - string - object name. object name can be specified by a wildcard where "?" matches any character and "*" matches any number of any characters.
Parameter 2 - string - first parameter to the command.
Parameter 3 - string - second parameter to the command.
.....
Parameter N - string - (N-1)th parameter to the command.
This method can be used to execute commands on the graphics objects. The

namespace for the graphics object is global for the whole Program Channel, so it doesn't matter which port this command is sent to. All graphics object whose name matches the pattern specified will react to the command. Method parameters are passed as strings even when the command specifies a different type. A lexical conversion is expected in such a case.

graphics_multi_action

Method name: "graphics_multi_action"
 Parameter 0 - array - array of arrays of parameters for the actions. Each subarray has the same structure as in "graphics_action"
 This method is similar to "graphics_action" and can be used to execute a batch of commands in one go.

graphics_get_state

Method name: "graphics_get_state"
 Parameters - none
 Return value - array<AVGraphicsStateInfo> - array of states for all currently loaded graphics compositions
 This method allows to query current list of loaded graphics compositions as well as object lists for these compositions.
 Structure AVGraphicsStateInfo has the following fields:
 "name" - string - name of a graphics composition (title field of the clip through which this composition was loaded)
 "elems" - array<AVGraphicsElemStateInfo> - element list for the composition
 Structure AVGraphicsElemStateInfo has the following fields:
 "name" - string - name of the element
 "type" - string - type of the element. The following types are defined: "PIP Frame", "Text Area", "Text Counter", "Text Feed", "Rect" and "UVMeter"
 "visible" - int - non-zero value indicated that the element currently shown
 "empty" - int - non-zero value indicates that the element is "empty". "Emptiness" is specially defined for each type of object.
 "cued" - int - amount left to play in milliseconds of the current media
 "appended" - int - duration in milliseconds of the queued media

"modif_counter" - int - amount of modifications made to elem, only changes for "PIP Frame", all other elem types default to 0.

"value" - int - counter value (ms) for "Text Counter" object, all other elem types default to 0.

"Emptiness" is defined for the following types:

"PIP Frame" - an object of this type is empty when no video is playing.

"Text Feed" - an object of this type is empty when there is no more text to display.

For "PIP Frame" objects the "cued" duration is the duration of the media that has been "taken" minus what has already been played. "appended" duration is duration of the media set with "pip_next_clip" but has not been "taken" yet.

For "Text Feed" objects the "cued" duration is the amount of time that is left for the current text object and the "appended" duration for the queued text objects. Unlike the "cued" duration "appended" doesn't include time needed to travel across the display area for scrolling or crawling feeds.

graphics_get_text_metrics

Method name: "graphics_get_text_metrics"

Parameters:

Parameter 0 - string - name of the text feed object

Parameter 1 - string - text to be measured

Return value - array<AVGraphicsTextMetrics> - array of text metrics for all currently loaded graphics compositions containing a text feed with the name specified.

Structure AVGraphicsTextMetrics has the following fields:

"name" - string - name of a graphics composition (title field of the clip through which this composition was loaded)

"width" - int - width of the text in pixels.

"height" - int - height of the text in pixels.

"duration" - int - Duration in millisecond that will take this text to be played. For scrolling and crawling compositions this duration is the time needed for the text to appear and doesn't include the time needed to travel across the screen.

lookup_media

```

Method name: "lookup_media"
Parameter 0 - string - media object uri
Return value: - structure with the following fields:
  "exists" - int - 1 if a media object corresponding to uri is found and 0
  otherwise
  "width" - int - width of the media in pixels
  "height" - int - height of the media in pixels
  "frame_time" - double - frame duration of the media in seconds
                  (-1 for still images, 0.01 for audio only)
  "in_point" - int - in point of the media in frames
  "duration" - int - out point minus in point of the media in frames
                  (-1 for infinite streams/clips or still images)
  "length" - int - total duration of the media in frames regardless of
  in/out points
                  (-1 for infinite streams/clips or still images)
  "sample_rate" - int - audio sample rate in samples per second
  "channels" - int - number of audio channels
  "lead_out" - int - lead out duration in frames.

  "title" - string - Title field of the media asset
  "type" - string - Type field of the media asset
  "comment" - string - Comment field of the media asset
  "src_name" - string - Source name field of the media asset
  "layer" - string - Layer field of the media asset
  "user_0" to "user_7" - string - User fields of the media asset

```

purge_media_at

```

Method name "purge_media_at"
Parameter 0 - string - media url or media id
Parameter 1 - int - julian day at the end of which the media will be purged
Return value: none

```

purge_media

```

Method name "purge_media"
Parameter 0 - string - media url or media id
Return value: none

```

Graphics object commands and their parameters.

object_set_transition

Sets effect type and duration when the object is hidden/shown.

Command format: `object_set_transition <name> [effect] [dur]`

Parameters:

[effect] - string - transition effect name. Possible values:
"fade" - object's opacity gradually increases/decreases.
"cut" - object's shows up/hides instantly.
[dur] - integer - duration of the transition in milliseconds.

object_show

Makes the object to show up after [delay] milliseconds. The command clears "hide_empty" flag.

Command format: object_show <name> [delay] [effect] [dur]

Parameters:

[delay] - integer - time in millisecond after which the show up effect is started.
[effect] - string - transition effect name. Possible values:
"fade" - object's opacity gradually increases.
"cut" - object's shows up instantly.
[dur] - integer - duration of the transition in milliseconds.

object_hide

Makes the object to hide itself after [delay] milliseconds. The command clears "hide_empty" flag.

Command format: object_hide <name> [delay] [effect] [dur]

Parameters:

[delay] - integer - time in millisecond after which the hide effect is started.
[effect] - string - transition effect name. Possible values:
"fade" - object's opacity gradually decreases.
"cut" - object's hides itself instantly.
[dur] - integer - duration of the transition in milliseconds.

object_toggle

Makes the object to show up if it's hidden or hide itself if it's shown. The command clears "hide_empty" flag.

Command format: object_toggle <name> [delay] [effect] [dur]

Parameters:

[delay] - integer - time in millisecond after which the show up/hide

effect is started.

[effect] - string - transition effect name. Possible values:

"fade" - object's opacity gradually increases/decreases.

"cut" - object's shows up/hides itself instantly.

[dur] - integer - duration of the transition in milliseconds.

object_hide_empty

Sets the object's "hide_empty" state. When "hide_empty" state is set the graphics engine monitors the "empty_state" of the object. When the object is found to be in the "empty_state"

for the <delay> specified all the object specified on the hide list are automatically hidden.

Different graphics object implement their "empty_state" according to the nature of the object.

As of now the following object implement "empty_state" functions:

"Picture in picture" - the empty state is signaled when no video clip is currently taken

for playback.

"Text feed" - the empty state is signaled when there is no more text to be displayed.

Command format: object_hide_empty <name> <delay> <name1[,name2[,...]]> [effect] [dur]

Parameters:

<delay> - integer - time in millisecond that the object has to be found in the "empty_state".

<hide_list> - string - coma-separated list of object name masks to be hided.

[effect] - string - transition effect name. Possible values:

"fade" - object's opacity gradually increases/decreases.

"cut" - object's shows up/hides itself instantly.

[dur] - integer - duration of the transition in milliseconds.

object_move

Moves the object by <dx>/<dy> pixel in [dur] milliseconds.

Command format: object_move <name> <dx> <dy> [dur]

Parameters:

<dx> - integer - horizontal offset in pixels.

<dy> - integer - vertical offset in pixels.

[dur] - integer - time in milliseconds required to move the object.

object_scale

Changes the object's size by <dw>/<dh> pixels in [dur] milliseconds.

Command format: object_scale <name> <dw> <dh> [dur]

Parameters:

<dw> - integer - horizontal size delta.

<dh> - integer - vertical size delta.

[dur] - integer - time in milliseconds required to scale the object.

object_set_geometry

Changes the object's position and size in [dur] milliseconds.

Command format: object_set_geometry <name> <x> <y> <w> <h> [dur]

Parameters:

<x> - integer - horizontal position of the top left corner. (uses current X if at empty string is given)

<y> - integer - vertical position of the top left corner. (uses current Y if at empty string is given)

<w> - integer - width of the object. (uses current W if at empty string is given)

<h> - integer - height of the object. (uses current H if at empty string is given)

[dur] - integer - time in milliseconds required to transform the object.

object_set_opacity

Changes the object's opacity in [dur] milliseconds.

Command format: object_set_opacity <name> <opct> [dur]

Parameters:

<opct> - integer - opacity of the object where 0 means fully transparent object and

100 means fully opaque object.

[dur] - integer - time in milliseconds required to change the object's opacity.

==== object_raise ====

<code>

Brings the object to the front in the graphics composition.

Command format: object_raise <name>

Parameters: none

pip_set_clip

Sets a new clip for a Picture in picture insertion object. All previously appended clips are purged from the queue.

Command format: pip_set_clip <name> <uri|EMPTY> [loop] [in_point] [out_point]

Parameters:

<uri|EMPTY> - string - url of a clips to be set for this PIP object.
Special value "EMPTY"

will set a fully transparent clip as the current clip.

[loop] - boolean - if "1"/"true"/"yes" then the duration of the clip will be infinite and

the contents will be looped if needed. The default is true".

[in_point] - integer - The in point in frames. If not specified the in point is taken

from the media.

[out_point] - integer - The out point in frames. If not specified the out point is taken

from the media.

Note that special optimization takes place if the same uri is specified in consecutive calls and the media will not be reused. To display a single frame from a clip the out_point should be one frame greater than the in point.

pip_next_clip

Appends a new clip to the playback queue of a Picture in picture insertion object. If autostart flag is not set all previously append clips that have not been "taken" will be purged from the queue. The clip will be inactive and will not appear on playout until it is "taken" by "pip_take" command.

Command format: pip_next_clip <name> <uri|EMPTY> [loop] [autostart][in_point][out_point]

Parameters:

<uri|EMPTY> - string - url of a clips to be appended for this PIP object.

Special value "EMPTY"

will append a fully transparent clip.

[loop] - boolean - if "1"/"true"/"yes" then the duration of the clip will be infinite and

the contents will be looped if needed. The default value is "true".

[autostart] - boolean - if "1"/"true"/"yes" then this clip will automatically start when the previous clip finishes. The default value is "false". Transition

for such clips is always "cut".

[in_point] - integer - The in point in frames. If not specified the in point is taken from the media.

[out_point] - integer - The out point in frames. If not specified the out point is taken from the media.

pip_set_transition

Sets effect for transition between clips in PIP objects.

Command format: pip_set_transition <name> [<cut|vmix|xmix|over>] [dur] [params]

Parameters:

[<cut|vmix|xmix|over>] - string - transition type

[dur] - int - transition duration in milliseconds

[params] - string - parameters for the transition. The following parameters are implemented:

"vmix":

"wipe=URL" - the url should refer to a still image resource.

"fade_alpha" - makes vmix to fade to zero alpha.

"#RRGGBB" - make vmix to fade to specific color.

"xmix":

"wipe=URL" - the url should refer to a still image resource.

"over":

"wipe=URL" - the url should refer to a semi-transparent video.

pip_take

Starts transition to the next clip in the playback queue that has been added with "pip_next_clip".

Command format: pip_take <name> [<cut|vmix|xmix|over>] [dur] [params]

Parameters:

[<cut|vmix|xmix|over>] - string - transition type

```
[dur] - int - transition duration in milliseconds
[params] - string - parameters for the transition. The following
parameters are implemented:
  "vmix":
    "wipe=URL" - the url should refer to a still image resource.
    "fade_alpha" - makes vmix to fade to zero alpha.
    "#RRGGBB" - make vmix to fade to specific color.
  "xmix":
    "wipe=URL" - the url should refer to a still image resource.
  "over":
    "wipe=URL" - the url should refer to a semi-transparent video.
```

pip_unloop

Breaks the loop of the first infinite duration clip in the playout queue. The new duration will be calculated in such a way that no more than one duration of out_point-in_point of the clip will be played depending on the current playout position.

Command format: pip_unloop <name> [on_empty]

Parameters:

[on_empty] - string - the name of an object with will trigger un-loop when it becomes empty.

If this parameter is not specified the loop will break immediately.

pip_set_audio_volume

Sets audio volume for the PIP element

Command format: pip_set_audio_volume <name> <vol_db>

Parameters:

<vol_db> - new audio volume in DB.

txt_append

Appends a text string to the text queue of a Text Feed object. Appends a text string to the currently displayed text when applied to a simple Text Area object.

Command format: txt_append <name> <text> [display_once] [break]

Parameters:

<text> - string - a UTF-8 text string to be added.

[display_once] - boolean - when "true" the text will be displayed only once.
Otherwise, it will be placed to the rotation queue.
(Ignored for simple Text Area objects)

[break] - boolean - when "true" clears the rotation queue before adding the text.
If applied to a simple Text Area clear current text before appending when "true".

txt_break

Clears the rotation queue of a Text Feed object.
Clears current text when applied to a simple Text Area object.

Command format: txt_break <name> [remove_current]

Parameters:

[remove_current] - boolean - when "true" the currently displayed item will be removed immediately as well as the queue.
(Ignored for simple Text Area)

txt_time_slot

Inform a text feed of how much time is available to display the text. Only those text strings will be displayed that will fit into current time slot. If the time slot is too short the remaining strings will be in waiting state and will be processed as soon as the time slot is big enough for the next string.

Command format: txt_time_slot <name> [time_available]

Parameters:

[time_available] - int - duration of the current time slot in milliseconds. A negative duration means an infinite amount of time. The default value is -1.

counter_start

Instructs a text counter object of type "External" to start counting time.

Command format: counter_start <name>

Parameters: none

counter_stop

Instructs a text counter object of type "External" to stop counting time.

Command format: counter_stop <name>

Parameters: none

counter_set_val

Sets a text counter object of type "External" to a new value.

Command format: counter_set_val <name> <value_ms> [start|stop]

Parameters:

- <value_ms> - int - new value of the text counter in milliseconds
- [start|stop] - string - "start" makes the counter to start counting, "stop" maked the counter to stop counting.

Text Objects Tags

Text object such as "Text Feed" support control tags embedded in the text body. This tags can be used to control how the text will be displayed.

Control tags have the following structure: "<?TAG BODY?>". Every tag is opened by the following sequence of symbols - "{?" followed by the tag's body and "?}" closes the tag.

For the time being, the following tags are supported:

- {?fg:COLOR?} - sets foreground color to COLOR
- {?fg:?} - sets foreground color to default value (from the graphics composition definition)
- {?bg:COLOR?} - sets background color to COLOR
- {?bg:?} - sets background color to default value (from the graphics composition definition)
- {?NAME?} - inserts a static image using "NAME" as the image url (id-only urls can be used here)

Color values may be specified by 6 or 8 hexadecimal symbols prepended with "#"
("#RRGGBB" or "#RRGGBBAA"). HTML color names can also be used.

Example1: {?LOGO?} - this will insert a still picture with title "LOGO" from

the media database

Example2: `{?fg:ff0000?}`This will be red`{?fg:?}`

Example3: `{?bg:white?}`This will be on white background`{?bg:?}`

Action Router service

The action router service sits at port number 5454 and it accessible through XML-RPC protocol. Every service which needs to react to actions has to register with the action router by calling «add_action» method. Once registered the new handler can be selected in the web console configurator of an action source service. Action parameters are always passed as strings and converted to numerical types if the actual handler method specifies a numeric type. Actual handler method parameter are mapped to actual action parameter according ActionParamInfo specification.

execute_action

Method name: "execute_action"

Call an action handler for the specified service.

This method has the following parameters:

Parameter 0 - string - service name

Parameter 1 - string - action name

Parameter 2 - array - array of action parameters. Each action parameter is a string and is

interpreted internally by an action handler.

add_action

Method name: "add_action"

Register a new action handler with the Action Router service

This method has the following parameters:

Parameter 0 - struct ActionInfo - a structure describing a new action handler

Return value - None.

The ActionInfo structure contains the following members:

"name" - string - name of the action used to trigger the action by other services.

"server_type" - string - server type name. It's used for user's reference only.

"server_name" - string - server name used to trigger the action by other services.

"descr" - string - user visible description of the action.

"arg_descr" - array<string> - user visible description of the action parameters.

"reaction_delay" - double - time in seconds needed to execute the action.

For example,

the play-list actions are usually bound to

some particular event in the play-list at some point in time.
 The play-list action generator will take this reaction delay
 into account and emit the action "reaction_delay" seconds
 earlier so that the actual handler completes at the proper
 time.

"handlers" - array<ActionHandlerInfo> - array of the action handler method descriptions.

The ActionHandlerInfo struct contains the following members:

"host" - string - ip address of the service that handles the action.

"port" - int - port number of the service that handles the action.

"method" - string - XML-RPC method name that handles the action.

"parameters" - array<ActionParamInfo> - array of actual action parameters mapping to the

XML-RPC method's parameters.

The ActionParamInfo struct contains the following members:

"param_type" - string - type name of a parameter. Currently, the following type names

are supported:

"int" - integer number

"string" - utf8 string

"param_val" - string - value specifier. When a '%X' character sequence is encountered

(where X can be in range '1'-'9' or 'A'-'Z') it is

substituted

by corresponding actual action parameter. For

example, an

action may specify just one parameter but the

handler method

may take two parameters. In this case, one of the

parameters

can be specified and a fixed value and the other

one as "%1"

which will use the action's actual parameter.

From:
<http://wiki.skylark.ru/> - **wiki.skylark.ru**

Permanent link:
http://wiki.skylark.ru/ru/api/graphics_actions

Last update: **2020/02/07 15:58**

