WATCHMAN ICD

Watchman

Version 2024.0

Doc. Name: Watchman-2024.0-ICD-003

Revision	Date	Changed by	Summary of Changes
003	11-07-2024	M.S.	Add example requests and responses from using methods.
002	14-06-2024	M.S.	Increase patch level. Add lens_calibration to Stream. Replace description with title in Scene. Update component_type in Fault.
001	21-05-2024	M.S.	Add show_image, error_image, and frame_timeout to Overlay.
000	28-02-2024	M.S.	Initial document

Contents

Glossary

canvas The display area that Watchman uses to arrange and position overlays.

JSON JavaScript Object Notation, used to send and get requests from Watchman.

JSON-RPC A remote procedure call protocol encoded in JSON.

Mosquitto Open source message broker by Eclipse that implements the MQTT protocol.

MQTT A lightweight machine-to-machine network protocol used to publish and subscribe to network requests.

overlay A configured video source that is positioned on the canvas.

stream A video source used by Watchman (Video card/UDP/RTSP/File).

Video Wall The multiple connected screens over which Watchman displays video.

1. System Description

This document provides an overview of the definitions used in the Watchman Version 2024.0 API. Watchman is a video compositor that can take feeds from multiple camera sources and widgets and display them across multiple connected screens known as the Video Wall. Watchman is configured using JSON-RPC over MQTT.

Watchman subscribes to the MQTT topic `watchman_rpc`, and publishes replies to `watchman_response`, and alerts to `watchman_alert`.

2. Examples

2.1 Method examples

The following examples show the JSON requests, Mosquitto requests, and JSON responses for a variety of methods. Note that the JSONs have been piped through jq and/or sed to improve readability.

2.1.1 `add_stream`

This method is used to add new video stream sources to Watchman. These sources can be from a video capture card, a UDP/RTSP network stream or from a file reachable by the hardware running Watchman. Multiple streams can be defined per Watchman instance. An input stream primary source must be defined, this can be from multiple different sources, and each source requires unique parameters. Watchman can process video using the *ffmpeg* library or the *gstreamer* library, or a *dummy* source can be added for testing purposes. When using a remote RTSP source, username and password credentials may be required to access the stream.

An example JSON request to use the `add_stream` method is as follows:

```
{
  "id": "STREAMS",
  "jsonrpc": "2.0",
  "method": "add_stream",
  "params": [
    {
      "source": {
        "uri": "rtsp://user:password@192.168.1.68",
        "name": "rtsp"
      },
      "stream_id": 1
    }
  ]
}
```

This can be sent using Mosquitto with the following command:

```
mosquitto_rr -t "watchman_rpc" -e "watchman_reply" -m '{ "id": "STREAMS", "jsonrpc":
    "2.0", "method": "add_stream", "params": [ { "source": { "uri":
    "rtsp://user:password@192.168.1.68", "name": "rtsp" }, "stream_id": 1 } ] }'
```

The resulting response of this `add_stream` request is:

```
{
  "jsonrpc": "2.0",
  "result": {
    "stream_id": 1,
    "source": {
      "name": "rtsp",
      "uri": "rtsp://user:password@192.168.1.100",
      "resolution": [
        0,
        0
      ],
      "flags": 0,
      "extra": []
    },
    "frame_timeout": 0
  },
  "id": "STREAMS"
}
```

2.1.2 `add_render`

This method defines the render environment for the combined overlays. Multiple render surfaces can be defined and can correspond to a UDP video stream or a monitor connected to the device running Watchman. An output stream can be defined for each render. Parameters such as the bitrate, destination address, port, and resolution can be configured. The panel view (position on the canvas) is also defined here and is used to tell Watchman where the render is going to be displayed.

An example JSON request to use the `add_render` method is as follows:

```
{
  "jsonrpc": "2.0",
  "id": 1,
  "method": "add_render",
  "params": [
    {
      "render_id": 0,
      "panel_view": [
        0,
```

```
        0,
        1,
        1
      ]
    }
  ]
}
```

This can be sent using Mosquitto with the following command:

```
mosquitto_rr -t "watchman_rpc" -e "watchman_reply" -m '{ "jsonrpc": "2.0", "id": 1,
    "method": "add_render", "params": [ { "render_id": 0, "panel_view": [ 0, 0, 1, 1 ]
    } ] }'
```

The resulting response of this `add_render` request is:

```
{
  "jsonrpc": "2.0",
  "result": {
    "render_id": 0,
    "panel_view": [
      0.0,
      0.0,
      1.0,
      1.0
    ],
    "resolution": [
      1024,
      768
    ],
    "clear_color": [
      0.0,
      0.0,
      0.0
    ]
  },
  "id": 1
}
```

2.1.3 `add_overlay`

This method takes a previously defined video stream and uses it to create an overlay. This overlay is visible on the virtual screen. This function can be used to perform multiple actions on the supplied video stream:

- The original video can be cropped down to a smaller size.
- A scene can be assigned to each overlay, meaning that it can be hidden from display until required.
- The original stream can be flipped along the X axis.
- The position of the overlay on the canvas can be defined, allowing users to place the overlay anywhere on screen.

An example JSON request to use the `add_overlay` method is as follows:

```
{
  "id": "OVERLAYS",
  "jsonrpc": "2.0",
  "method": "add_overlay",
  "params": [
    {
      "scenes": [ "main" ],
      "crop": [
        0.0,
        0.0,
        1.0,
        1.0
      ],
      "flip_x": false,
      "overlay_id": 0,
      "stream_id": 0,
      "position": [
        0.0,
        0.1,
        0.5,
        0.7
      ],
      "z_index": 1
    }
  ]
```

```
}
```

This can be sent using Mosquitto with the following command:

```
mosquitto_rr -t "watchman_rpc" -e "watchman_reply" -m '{ "id": "OVERLAYS", "jsonrpc":
    "2.0", "method": "add_overlay", "params": [ { "crop": [ 0.0, 0.0, 1.0, 1.0 ],
    "flip_x": false, "overlay_id": 0, "stream_id": 0, "position": [ 0.0, 0.1, 0.5, 0.7
    ], "z_index": 1 } ] }'
```

The resulting response of this `add_overlay` request is:


```
{
  "jsonrpc": "2.0",
  "result": {
    "overlay_id": 0,
    "stream_id": 0,
    "show_image": "",
    "error_image": "",
    "frame_timeout": 0,
    "scenes": [],
    "z_index": 1,
    "position": [
      0.0,
      0.10000000149011612,
      0.5,
      0.699999988079071
    ],
    "crop": [
      0.0,
      0.0,
      1.0,
      1.0
    ],
    "crop_indicator_position": [
      0.44999998807907107,
      0.02500000037252903,
      0.10000000149011612,
      0.10000000149011612
    ],
```

```
      "border": false,
      "no_scaling": false,
      "selectable": true,
      "enable_zoom": false,
      "flip_x": false,
      "flip_y": false,
      "swap_x_y": false
    },
    "id": "OVERLAYS"
}
```

2.1.4 `status`

The JSON request to use the `status` method is as follows:

```
{
  "jsonrpc": "2.0",
  "id": 1,
  "method": "status"
}
```

This can be sent using Mosquitto with the following command:

```
mosquitto_rr -t "watchman_rpc" -e "watchman_reply" -m '{ "jsonrpc": "2.0", "id": 1,
    "method": "status" }'
```

The resulting response of this `status` request is:

```
{
  "jsonrpc": "2.0",
  "result": {
    "software_version": "2024.0 (2024-06-13, commit abc1234, branch 'develop')",
    "api_version": {
      "version": 1,
      "patch_level": 0,
      "extra_version": "beta"
    },
    "fault_conditions": [],
    "time": 423308275891
  },
```


```
  "id": 1
}
```

2.1.5 clear_current_settings

The JSON request to use the clear_current_settings method is as follows:

```
{
  "jsonrpc": "2.0",
  "id": 1,
  "method": "clear_current_settings"
}
```

This can be sent using Mosquitto with the following command:

```
mosquitto_rr -t "watchman_rpc" -e "watchman_reply" -m '{ "jsonrpc": "2.0", "id": 1,
    "method": "clear_current_settings" }'
```

The resulting response of this `clear_current_settings` request is:


```
{
  "jsonrpc": "2.0",
  "result": {
    "profile": {
      "profile_id": 0,
      "name": "Unnamed"
    },
    "screen_text": {
      "log": {
        "spacing": 0.03999999910593033,
        "x": -0.8999999761581421,
        "y": 0.8999999761581421,
        "count": 12,
        "expires": true,
        "lifetime_ns": 10000000000,
        "scene": "",
        "viewport": [
          0.0,
          0.0,
          1.0,
          1.0
        ],
        "color": [
          1.0,
          1.0,
          1.0,
          1.0
        ],
        "size": 1.0,
        "alignment": 1,
        "font": 1
      },
      "status": {
        "x": 0.0,
        "y": 0.0,
        "scene": "",
```

```
          "viewport": [
            0.0,
            0.0,
            1.0,
            1.0
          ],
          "color": [
            0.20000000298023225,
            1.0,
            0.20000000298023225,
            1.0
          ],
          "size": 1.5,
          "alignment": 2,
          "v_alignment": 1,
          "font": 2
        }
      },
      "streams": [],
      "overlays": [],
      "renders": [],
      "scenes": [],
      "last_updated": "Never"
    },
    "id": 1
}
```


2.2 FaultAlert

A FaultAlert is published to the MQTT topic "watchman_alert" periodically until they are resolved and then until they expire. While watchman is running, execute the following to subscribe to the fault alerts:

```
mosquitto_pub -t "watchman_alert"
```

This will usually produce a response similar to the following:


```
{
  "faults": [],
```

```
  "time": 16230902017620
}
```

where, the existence of no faults is shown as an empty array of type `ult]Fault`.

An example fault can be induced by an invalid camera source due to an incorrect camera IP address. The resulting fault alert during this fault is shown below:

```
{
  "faults": [
    {
      "component_type": 1,
      "item_id": 0,
      "fault_code": 2,
      "info": "Source process 0 (rtsp) did not provide a frame in time.",
      "began": 16261731260622,
      "last_seen": 16261731260622
    }
  ],
  "time": 16261739757696
}
```

3. API

3.1 Methods

Method:	**status**
Result Type:	`api::watchman::Status`
Parameters:	*None*

Method:	**load_settings**			
Result Type:	`settings::Compositor`			
Parameters:	**Name:**	`profile_id`	**Type:**	ProfileId (int)

Method:	**delete_settings**			
Result Type:	ProfileId (int)			
Parameters:	**Name:**	`profile_id`	**Type:**	ProfileId (int)

Method:	**save_current_settings**
Result Type:	`settings::Compositor`
Parameters:	*None*

Method:	**clear_current_settings**
Result Type:	`settings::Compositor`
Parameters:	*None*

Method:	**get_profiles**
Result Type:	Array of `settings::Profile`
Parameters:	*None*

Method:	**get_current_settings**
Result Type:	`settings::Compositor`
Parameters:	*None*

Method:	**get_settings**			
Result Type:	`settings::Compositor`			
Parameters:	**Name:**	`profile_id`	**Type:**	ProfileId (int)

Method:	**get_scenes**
Result Type:	Array of `settings::Scene`
Parameters:	*None*

Method:	**set_scenes**			
Result Type:	Array of `settings::Scene`			
Parameters:	**Name:**	`scenes`	**Type:**	Array of `settings::Scene`

Method:	**get_current_scene**
Result Type:	`settings::Scene`
Parameters:	*None*

Method:	**set_current_scene**			
Result Type:	`settings::Scene`			
Parameters:	**Name:**	`scene_id`	**Type:**	String
		`indicator`		Bool

Method:	**set_current_scene_by_index**			
Result Type:	`settings::Scene`			
Parameters:	**Name:**	`index`	**Type:**	Unsigned int
		`indicator`		Bool

Method:	**change_selected_overlay**			
Result Type:	ItemId (int) or null			
Parameters:	**Name:**	`increment`	**Type:**	Int

Method:	**joystick_position**			
Result Type:	Array of 3 float			
Parameters:	**Name:**	`position`	**Type:**	Array of 3 floats

Method:	**set_profile**			
Result Type:	`settings::Profile`			
Parameters:	**Name:**	`profile`	**Type:**	`settings::Profile`

Method:	**add_stream**			
Result Type:	`settings::Stream`			
Parameters:	**Name:**	`stream`	**Type:**	`settings::Stream`

Method:	**remove_stream**			
Result Type:	ItemId (int)			
Parameters:	**Name:**	`stream_id`	**Type:**	ItemId (int)

Method:	**add_overlay**			
Result Type:	`settings::Overlay`			
Parameters:	**Name:**	`overlay`	**Type:**	`settings::Overlay`

Method:	**move_overlay**			
Result Type:	`settings::Overlay`			
Parameters:	**Name:**	`overlay_id`	**Type:**	ItemId (int)
		`position`		Array of 4 floats

Method:	**remove_overlay**			
Result Type:	ItemId (int)			
Parameters:	**Name:**	`overlay_id`	**Type:**	ItemId (int)

Method:	**add_render**			
Result Type:	`settings::Render`			
Parameters:	**Name:**	`render`	**Type:**	`settings::Render`

Method:	**remove_render**			
Result Type:	ItemId (int)			
Parameters:	**Name:**	`render_id`	**Type:**	ItemId (int)

Method:	**get_video_interface**
Result Type:	Array of `settings::VideoInterface`
Parameters:	*None*

3.2 Definitions of Types

Fault			
Name	**Type**	**Required**	**Description**
`component_type`	ComponentType (Unsigned int)	Yes	Possible values: 0: None, 1: Heartbeat, 2: Source, 3: Sink, 4: Render, 5: Overlay, 6: Keyboard.
`item_id`	ItemId (int)	Yes	Individual element that the fault corresponds to.
`fault_code`	FaultCode (Unsigned int)	Yes	Possible values: 0: General fault, 1: Screen not found or disconnected, 2: Frame out of date, 3: Sink restarted
`info`	String	Yes	Description of the fault.
`began`	Int	Yes	Fault start time (monotonic clock).
`last_seen`	Int	Yes	Fault was last reported at this time (monotonic clock).

FaultAlert			
Name	**Type**	**Required**	**Description**
`faults`	Array of `Fault`	Yes	Array of current reported faults. May be empty.
`time`	Int	Yes	Monotonic clock's time of this message, for comparison with fault times.

settings::Profile			
Name	**Type**	**Required**	**Description**
`profile_id`	ProfileId (int)	Yes	Profile identifier. Default value: 0.
`name`	String	Yes	Profile name. Default: "Unnamed".

settings::ScreenMode			
Name	**Type**	**Required**	**Description**
`description`	String	Yes	Description of the screen mode.
`mode`	String	Yes	A mode string that should be sent verbatim to select this screen mode.
`preferred`	Bool	Yes	True if this is a native/preferred mode of the monitor.

settings::MonitorInfo			
Name	**Type**	**Required**	**Description**
`description`	String	Yes	Description of the connected monitor.
`modes`	Array of `settings::ScreenMode`	Yes	A list of modes supported by the monitor.

settings::VideoInterface			
Name	**Type**	**Required**	**Description**
`connector_name`	String	Yes	Description of the connector, eg "DP-1".
`connected_device`	`settings::MonitorInfo`	No	The monitor information, if one is connected.

settings::Display			
Name	**Type**	**Required**	**Description**
`connector_name`	String	No	Name of connector type, e.g., HDMI.
`mode`	String	No	Monitor mode.
`vsync`	Bool	No	Default: *true*.

settings::Sink			
Name	**Type**	**Required**	**Description**
`name`	String	No	sink name. Default: 0.
`resolution`	Array of 2 unsigned ints	No	Sink resolution width and height. Default: (1024, 768).
`udp_destination`	String	No	UDP sink address. Default: 0
`udp_port`	Int	No	UDP port. Default: 5000.
`rtsp_port`	Int	No	RTSP port. Default: 8554.
`bitrate`	Int	No	Bitrate in `kb/sec`. Default: 2000.
`quality`	Int	No	Encoder `speed-preset`. Default: 1 `ultrafast`.
`key_int`	Int	No	Maximal distance between two key-frames. Default: 100.
`use_gpu`	Bool	No	Whether or not to use the GPU. Default: *false*.
`intra_refresh`	Bool	No	When enabled, Periodic Intra Refresh is used instead of IDR frames. Default: *false*.
`tag_family`	String	No	*AprilTag* family. Default: `tag36h11`, possible values: `tag36h11`, `tag36h10`, `tag36artoolkit`, `tag25h9`, `tag25h7`, `tag16h5`.
`tag_code`	Int	No	*AprilTag* code. Default: 0.
`tag_size`	Double	No	Default: 0.3.

settings::Render			
Name	**Type**	**Required**	**Description**
`render_id`	ItemId (int)	Yes	Render identifier. Default: 0.
`panel_view`	Array of 4 floats	No	Panel view x position, y position, width and height. Default: (0.0, 0.0, 1.0, 1.0).
`resolution`	Array of 2 unsigned ints	No	Resolution width and height. Default: (1024, 768).
`clear_color`	Array of 3 floats	No	Clear colour RGB value (default is black). Default: (0.0, 0.0, 0.0), value Range: (0 to 1, 0 to 1, 0 to 1).
`display`	`settings::Display`	No	Render window.
`sink`	`settings::Sink`	No	Render sink.

settings::FrameSource			
Name	**Type**	**Required**	**Description**
`name`	String	Yes	Possible values: *dummy*: for testing, *ffmpeg*: ffmpeg, *rtsp*: gstreamer rtsp/rtp client, *web*: Web plugins.
`uri`	String	Yes	The meaning is different depending upon the source, but typically a URL.
`resolution`	Array of 2 ints	No	Resolution, for sources that the user can control the resolution, e.g. web.
`flags`	Unsigned int	No	A sum of the chosen flag's values: 0 : nothing, 1 : use GPU (used by "rtsp" source (gstreamer)), 2 : use TCP (used by "ffmpeg" rtsp source).

settings::Stream			
Name	**Type**	**Required**	**Description**
`stream_id`	ItemId (int)	Yes	Unique number representing this source. Default: -1.
`source`	`settings::FrameSource`	Yes	Settings for the frame source.
`lens_calibration`	LensCalibrationParams (array of 7 floats)	No	Calibration parameters are *fx, fy, k1, k2, k3, k4, k5*. See: Adobe Camera Model; Simon Chen, Hailin Jin, Jeff Chien, Eric Chan, Dan Goldman; Adobe Systems Inc. Technical Report, Version 1.0, January, 2010
`frame_timeout`	Int	No	Timeout in milliseconds, after which the panel will display an error. Default: 1,000.

settings::Overlay			
Name	**Type**	**Required**	**Description**
`overlay_id`	ItemId (int)	Yes	overlay identifier. Default: -1.
`stream_id`	ItemId (int)	No	Identifier for the stream that the overlay is for. Default: -1.
`show_image`	String	No	Always shows selected image.
`error_image`	String	No	Select test card to be shown in event of error: 'DTUP_camera_failed_2', 'DTUP_camera_failed_3', 'DTUP_camera_failed_4', 'DTUP_camera_not_in_use_2', 'DTUP_camera_not_in_use_3', 'DTUP_camera_not_in_use_4'
`frame_timeout`	Int	No	Frame timeout in milliseconds. Default: 0.
`scenes`	Array of strings	No	The `settings::Compositor` scenes that will display the overlay. If `settings::Compositor` has no scenes, the overlay will always be displayed.
`z_index`	Int	No	overlay z index. Larger values appear on top. Default: 0.
`position`	Array of 4 float	No	overlay x position, y position, width and height. Default: (0.0, 0.0, 1.0, 1.0).
`crop`	Array of 4 float	No	overlay crop x position, y position, width and height. Default: (0.0, 0.0, 1.0, 1.0).
`crop_indicator_position`	Array of 4 float	No	Crop indicator x position, y position, width and height. Default: (0.45, 0.025, 0.1, 0.1).

`border`	Bool	No	Whether or not to draw a border around the overlay. Default: *false*.
`no_scaling`	Bool	No	Whether or not to scale the overlay *(unused)*. Default: *false*.
`selectable`	Bool	No	Whether or not the overlay is selectable. Default: *true*.
`enable_zoom`	Bool	No	Whether or not the overlay is zoomed. The `crop` needs to be set for `enable_zoom` to be applied. Default: *false*.
`flip_x`	Bool	No	Whether or not to flip the overlay on the x axis. Default: *false*.
`flip_y`	Bool	No	Whether or not to flip the overlay on the y axis. Default: *false*.
`swap_x_y`	Bool	No	Whether or not to swap x and y of the overlay. Default: *false*.

settings::Scene			
Name	**Type**	**Required**	**Description**
`id`	String	Yes	Unique string identifying the scene.
`title`	String	Yes	Title of the scene.

settings::TextLog			
Name	**Type**	**Required**	**Description**
`spacing`	Float	No	Vertical text spacing. Default: 0.04.
`x`	Float	No	x position. Default: -0.9.
`y`	Float	No	y position *(unused)*. Default: 0.9.
`count`	Unsigned int	No	Count of text items to log. Default: 12.
`expires`	Bool	No	Whether or not to remove text after its lifetime expires. Default: *true*.
`lifetime_ns`	Int	No	Lifetime of text in nanoseconds. Default: 10,000,000,000.
`scene`	String	No	Scene to display text on. Will also display if no current scene.
`viewport`	Array of 4 float	No	The text's viewport location. Default: (0.0, 0.0, 1.0, 1.0).
`color`	Array of 4 float	No	The text's colour. Default: (1.0, 1.0, 1.0, 1.0).
`size`	Float	No	Text horizontal and vertical size. Default: 1.0.
`alignment`	TextAlignmentEnum (Int)	No	Text alignment. Default: 1 *align left*. Possible values: 0 *align prefix left (unused)*, 1 *align left*, 2 *align center*, 3 *align right*, 4 *align postfix right*.
`font`	TextFontEnum (Int)	No	Font style. Default: 1 *bold*. Possible values: 0 *regular*, 1 *bold*, 2 *italic*.

settings::StatusText			
Name	**Type**	**Required**	**Description**
`x`	Float	No	x position. Default: 0.0.
`y`	Float	No	y position. Default: 0.0.
`scene`	String	No	Scene to display text on. will also display if no current scene.
`viewport_location`	Array of 4 float	No	The text's viewport location. Default: (0.0, 0.0, 1.0, 1.0).
`color`	Array of 4 float	No	The text's colour. Default: (0.2, 1.0, 0.2, 1.0).
`size`	Float	No	Text horizontal and vertical size. Default: 1.5.
`alignment`	TextAlignmentEnum (Int)	No	Text alignment. Default: 2 *align center*. Possible values: 0 *align prefix left (unused)*, 1 *align left*, 2 *align center*, 3 *align right*, 4 *align postfix right*.
`v_alignment`	TextVAlignmentEnum (Int)	No	Text vertical alignment. Default: 1 *center*. Possible values: 0 *top*, 1 *center*, 2 *bottom*.
`font`	TextFontEnum (Int)	No	Font style. Default: 2 *italic*. Possible values: 0 *regular*, 1 *bold*, 2 *italic*.

settings::ScreenText			
Name	**Type**	**Required**	**Description**
`log`	`settings::TextLog`	No	Text log settings.
`status`	`settings::StatusText`	No	Status text settings.

settings::Compositor			
Name	**Type**	**Required**	**Description**
`profile`	`settings::Profile`	No	The compositor profile.
`screen_text`	`settings::ScreenText`	No	The compositor screen text.
`streams`	Array of `settings::Stream`	No	The compositor streams.
`overlays`	Array of `settings::Overlay`	No	The compositor overlays.
`renders`	Array of `settings::Render`	No	The compositor renders.
`scenes`	Array of `settings::Scene`	No	The compositor scenes.
`last_updated`	String	No	Timestamp of last compositor settings update, or *Never*. Default: *Never*. Timestamp format: *YYYY-MM-DD HH:MM:SS.uuuuuu*

ApiVersion			
Name	**Type**	**Required**	**Description**
`version`	Unsigned int	Yes	Major version of this API.
`patch_level`	Unsigned int	Yes	Patch level of this API.
`extra_version`	String	Yes	Extra information about the version, e.g. beta. May be empty.

api::watchman::Status			
Name	**Type**	**Required**	**Description**
`software_version`	String	Yes	*major.minor.patch* version of Watchman.
`api_version`	`ApiVersion`	Yes	Version of this API.
`faults`	Array of `Fault`	Yes	Array of current reported faults. May be empty.
`time`	Int	Yes	Monotonic clock's time of this message, for comparison with fault times.