SURF Communication Solutions

SURF HMP API reference guide

Document version: 1.09
Date: June 2015

SURF Communication Solutions, Ltd.

Tavor Building, P.O. Box 343

Yokne'am 20692 Israel

Tel: +972 (0)73 714-0700

Fax: +972 (0)4 959-4055

Web site	http://www.surfsolutions.com/
Email	info@surfsolutions.com

Table of Contents

1. Introduction

1.1 Abstract

This document describes API specification between Media Processor and upper level application (the controller), for example Motion or Python script that was written for testing purposes.

MediaProcessor is the process name that handles the HMP functionalities, for example voice trans-coding or video mixing. It can be described as an engine for doing something

The Controller is the application that exploits the HMP abilities to implement some functionality, for example RTP voice conferencing. The application can be developed using any platform/programming language/operating system/etc. The only common thing between all those applications is that every application must comply to this API description in order to use HMP correctly.

1.2 General

- The API is based on TCP/IP protocol
- API is message based
- Each message consists of 4 byte length in little endian format + byte array of length that is contained in 4 first bytes.
- A byte array contains text message in UTF-8 encoding
- A text message is in JSON format
- Each text message is represented by { <message type> : { <message description> }} format
- Messages are asynchronous
- Most of parameters in messages have default values, if not specified – default value will be used. This allows to use this API both as a high level API when not specifying most of the fields, as well as very low level API when using all possible configurations and features

1.3 JSON standard

JavaScript Object Notation (JSON, RFC 4627) is a lightweight, text-based, language-independent data interchange format. JSON is a text format for the serialization of structured data. It defines a small set of formatting rules for the portable representation of structured data.

JSON can represent four primitive types (strings, numbers, booleans, and null) and two structured types (objects and arrays).

JSON is widely used today in mobile applications and the main purpose of this language is to replace heavy XML messages in various configuration protocols that do not use wide range of XML features. It is natively supported in internet browsers like mozilla firefox and google chrome.

2. Connection and handshake

2.1 Handshake

Media Processor listens on a predefined TCP port.

When a connection is accepted both sides send a predefined message "surfapi", this message is sent without 4 bytes length at the beginning, only 7 character bytes.

This is done in order to ensure than the other side supports the same protocol and avoid situation that the connection was made by mistake.

Immediately after "surfapi" message both sides sends a special "connect" message. This message is in the same format as other messages in the protocol, i.e. 4 bytes length and then text buffer in JSON syntax.

Both sides send these 2 messages immediately after TCP connection establishment without waiting to other side's response. If one of the sides does not support any of the parameters, it closes the connection, but by default the connection is considered as working.

If one receives first 7 bytes other than "surfapi" or timeout (TBD) expires, the connection is closed.

2.1.1 Connect message

"connect" message that is sent by both sides will be as following:

```
{"connect":{
    "api_version":<api version>
}}
```

api_version is set to API version number that is supported by the sender. For current version it is [1,2] (is represented as array of integers with major and minor versions)

The controller side may also include optional parameter in this message:

```
"keep_alive_timeout":20 // set keep alive timeout to 20 seconds
```

This parameter sets keep-alive timeout for the protocol, see next section for details. If set to 0, keep-alive mechanism is disabled. This is done mainly for easy debugging with Perl scripts.

If this parameter is not set, default value is used (20sec)

After "connect" message is received, each side validates the connection parameters sent by the other side and if it is not supported – sends "disconnect" message and closes TCP connection.

2.1.2 Disconnect message

"disconnect" message is sent by either controller side or Media Processor side before closing TCP connection. It is also send in a "regular" format – 4 bytes length with text buffer after it. The message is defined as following:

```
{"disconnect":{
    "error_code":<error_code>,
    "reason":<textual description of disconnect reason"
}}
```

"error_code" values corresponds to the following values:

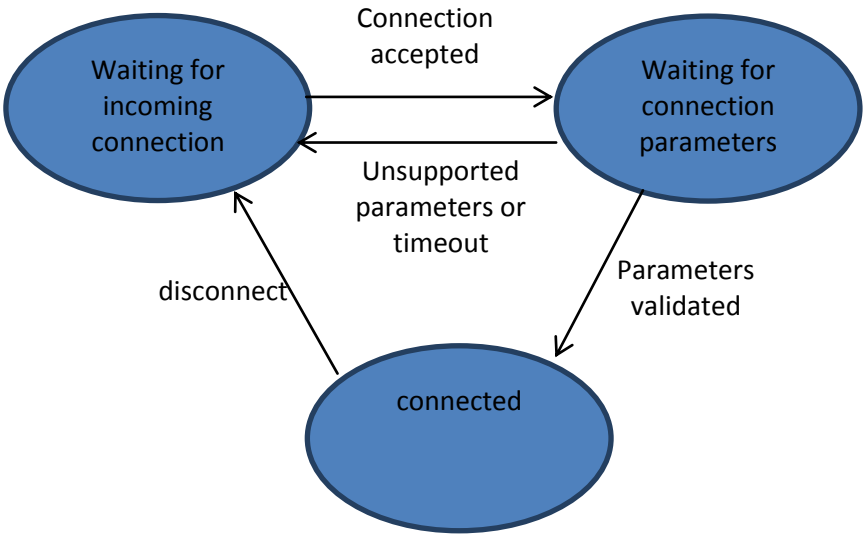
- 0 – "no error"
- 1 – "api version not supported"

"reason" can be set to any string value and is used mainly for debugging and logging purposes. In case of error it is set to textual error description. In case of other disconnect reason, it describes the cause of disconnect.

This message can also be used at any phase after "connect" message was received.

Any side that wants to close TCP connection must send "disconnect" message first

2.1.3 The Media Processor connection state machine



If The Media Processor enters "connected" state, it begins communicating using messages defined below.

If The Media Processor declines the connection because of unsupported parameters it closes this connection and resumes listening for new incoming connections.

2.2 "Keep alive"

Keep alive messages are sent by each side once in a defined period (TBD) only if other messages were not sent during this period.

If no messages were received from the remote side during timeout period (TBD), the connection is closed and the Media Processor will return to "waiting for incoming connection" state.

Keep alive message is represented by zero length message (4 zero bytes).

Keep alive messages can be disabled by setting "keep_alive_timeout" parameter of "connect" message to zero at the handshake phase. By default keep-alive messages is enabled. This option is added in order to use light weight scripts with Media Processor for testing purposes.

Default timeout is configured if this parameter is not specified in "connect" message. The default value of timeout is 20 seconds.

3. Message types

All messages in the API is divided into the following 3 groups:

- System wide messages – these messages names always start with "*sys*_" and be related to the whole system like connect, reset or get status
- Tool related messages – always start with "tool_" and make an impact only on a specific tool
- Tools group related messages – always start will "group_" and affect group of tools with a single message

From another perspective, these messages can be divided into 3 other groups:

- Request messages: these messages issue a request of a certain type; this type of message requires response; names of these messages always end with suffix "_req". Sent only by the controller side.
- Response messages: these messages are sent only as a response to a request message. Always end with suffix "_ans".
- Information messages: messages that are sent by the MediaProcessor to inform the controller on something (event, statistics, etc.). Does not require response.

Each message will contain its mandatory parameters. All request/response/information specific data is contained inside "data" object

3.1 "tool_req" messages

This message always contains some request to specific tool like configure/remove/command, etc. To allow the controller identify the response to this message (recall: the protocol is asynchronous) it contains req_id parameter that represents unique request id. The response message is named "tool_ans" and contains the same req_id parameter in its body. Another mandatory parameter is tool_id that identified the tool that handles this message.

Parameters:

- **req_id** – integer, request id, this number is sent back to the controller with response message so that the controller is able to identify the response
- **tool_id** – integer, unique tool identifier, valid values: 0 - 65535
- **req_type** – string, represents request type

The following subsections are structured according to req_type value

3.1.1 "set_config" request type

Request to create a new or configure an existing tool in the system. The rest parameters are assumed as default (at tool creation) or as unchanged (at tool configuration stage)

Parameters:

- **tool_type** – string, type of the tool (h264 encoder, resizer, voice FE IP, etc.)

- **group_id** – integer, optional, id of the group of tools which can be used later to manipulate all group's tools
- **app_info** – string, contains optional information that can be used by the application/for debugging purposes; this information is saved inside the tool and sent to the controller with every message related to this tool
- **dst_tool_ids** – array, (used only in specific tools) used by several tools. Consists of array of integers which represents unique tool IDs of the tools which will receive this tool's output. If not set, output list is considered empty. Range of values for each array member: 0..65535
- **events** – array of objects that contains configuration for event messages generated by this tool. All events are disabled by default. Each object contains the following fields:
  - **type** – string, event message name
  - **enable** – Boolean, true/false, enable/disable this type of event messages
- **status** – array of objects that contains configuration for status messages generated by this tool. All messages are disabled by default. Each object contains the following fields:
  - **type** – string, status message name
  - **period** – integer, period of automatic sending of this status in milliseconds. When set to 0, this status type is considered disabled
- **traces** – array of objects that contains configuration of debug traces. Every trace type represents a separate stream for each tool. So, if enabled for this tool's configuration, only information from this tool will be saved to the given file. Each object contains the following fields:
  - **id** – integer, identifier of the requested trace type
  - **file** – string, file name to write the trace information to
  - **action** – string, add/remove trace: "add" opens an empty file and starts recording this trace, "remove" closes the given file.
- **active** – Boolean, if set to false, the system scheduler will not run this tool until this flag is set to true. Default value: true
- Additional parameters (can be represented by a tree of parameters with any amount and types of parameters) – specific for each tool type

Example:

```
{"tool_req" : {
  "req_type":"set_config"
  "req_id" : 4002,
  "tool_id" : 2,
  "data":{"tool_type" : "h264_decoder",
     "app_info" : "any textual information for debug",
     "out_id" : 3,
     "max_width" : 1280,
     "max_height" : 720}
}}
```

3.1.2 "remove" request type

Request to remove specific tool from the system.

Parameters: none

Example:

```
{"tool_req":{
  "req_type":"remove",
  "req_id":4003,
  "tool_id":2}}
```

3.1.3 "get_config" request type

Request to get current configuration of a tool

Parameters:

- **send_non_default_only(not implemented)** – Boolean, if set to true, tool_getconfig_ans message will contain only parameters that differ from default values. Default value: false

Examples:

```
{"tool_req":{
  "req_type":"get_config",
  "req_id":4005,
  "tool_id":2,
  "data":{"send_non_default_only":true}}}
```

3.1.4 "command" request type

Request to some action to be performed by the tool, for example request for intra frame generation by encoder or request for DTMF generation in voice tool.

Parameters:

- **cmd_type** – string, command name, action that should be performed by the tool
- command specific parameters

Examples:

```
{"tool_req":{
  "req_type":"command",
  "req_id":4006,
  "tool_id":67,
  "data":{"cmd_type" : "generate_i_frame"}
}}
```

3.1.5 "get_status" request type

Request to send tool_inf message with information type of "status"

Parameters:

- **type** – string, status message name
- **reset_stat(not implemented)** – Boolean, if true, this status data is reset at the Media Processor. Default value:false

Examples:

```
{"tool_req":{
  "req_type":"get_status",
  "req_id":4007,
  "tool_id":45,
  "data":{"stat_type":"general",
    "reset_stat":true}
```

3.2 "tool_ans" messages

Message of this type is sent by the Media Processor as a response to the request message. It contains the same "req_id" that was set in the original request message. It also contains req_type field mostly for debugging and readability of the protocol.

Parameters:

- **req_id** – integer, request id, this number is copied from the original tool_req message
- **tool_id** – integer, unique tool identifier
- **req_type** – string, represents the original request type
- **app_info** – string, optional info that was set in tool_req::set_config message. If was not set, it will not be included in this message

The following subsections are structured according to req_type value.

3.2.1 "set_config", "get_config", "remove" and "command" request types

These 3 types of responses contain the same parameters:

Parameters:

- **error_code** – integer, 0 on success; otherwise contains a specific error number
- **text_description** – textual description of the result

Example:

```
{"tool_ans":{
    "req_type":"set_config",
    "req_id":5006,
    "tool_id":2,
```

```
    "app_info":"debug info",
    "data":{"error_code":0,
      "text_description":"OK"}}}
```

3.2.2 "get_config" request type

Contains current configuration of a tool

Parameters:

- Tool specific parameters – contain tool's current configuration

example:

```
{"tool_ans":{
    "req_type":"get_config",
    "req_id":4002,
    "tool_id":2,
    "data":{"tool_type":"h264_decoder",
      "app_info":"chain id = 4",
      "out_id":3,
      "max_width":1280,
      "max_height":720}}}
```

3.3 "sys_req" messages

Messages of this type contain a request for certain action that the MediaProcessor must perform. As well as tool_req messages, it contain req_id field. As an answer to this message, Media Processor must send sys_ans message with the results.

Parameters:

- **req_id** – integer, request id, this number will be sent back to the controller with response message so that the controller will be able to identify the response
- **req_type** – string, represents request type

The following subsections are structured according to req_type value.

3.3.1 "get_tool_defaults" request type

Request to get default configuration of a tool

Parameters:

- **tool_type** – string, tool type

Example:

```
{"sys_req":{
  "req_type":"get_tool_defaults"
  "req_id":4004
```

```
"data":{"tool_type":"rtp_session"}}}
```

3.3.2 "get_tools_list" request type

Request to obtain list of all currently configured tools in the system (Media Processor).

Parameters: none

Examples:

```
{"sys_req":{
    "req_type":"get_tools_list",
    "req_id":4006}}
```

3.3.3 "set_config" request type

System-wide configuration message

Parameters:

- **scheduler_mode(not implemented)** – scheduling model of Media Processor, supported values:
  - multimedia – supports full featured scheduler
  - high_density_voice – supports high density voice scheduling model
- **status** – array of objects that contains configuration for status messages generated by the system. All messages are disabled by default. Each object contains the following fields:
  - **type** – string, status message name
  - **period** – integer, period of automatic sending of this status in milliseconds. When set to 0, this status type is considered disabled
- **traces** – array of objects that contains configuration of system wide debug traces. So, if enabled for this tool's configuration, only information from this tool will be saved to the given file. Each object contains the following fields:
  - **id** – integer, identifier of the requested trace type
  - **file** – string, file name to write the trace information to
  - **action** – string, add/remove trace: "add" opens an empty file and starts recording this trace, "remove" closes the given file.
- other parameters can be optionally added in the future

Example:

```
{"sys_req":{
    "req_type":"set_config",
    "req_id":5006,
    "data":{"scheduler_mode":"multimedia",
      "status_msg_period":10}}}
```

3.3.4 "get_config" request type

A request to send current system configuration

Parameters: none

example:

```
{"sys_req":{
    "req_type":"get_config",
    "req_id":5006}}
```

3.3.5 "command" request type

System related command

Parameters:

- **cmd_type** – string, command type, the following command types are supported:
    - **reset(not implemented)** – request to release all Media Processor resources and re-initialize the system
- command specific parameters, depends on command type

3.3.5.1 reset command specific parameters

none

example:

```
{"sys_req":{
    "req_type":"command",
    "req_id":6004,
    "data":{"cmd_type":"reset"}}}
```

3.3.6 "get_version" request type

System related command, Command for sending information about the Media Processor like all supported tool types and software version

Parameters: none

Example:

```
{"sys_req":{
    "req_type":"get_version",
    "req_id":3241}}
```

3.3.7 "get_density" request type (not implemented)

System related command, Command for sending the Media Processor density information
Parameters: none

Example:

```
{"sys_req":{
    "req_type":"get_density",
    "req_id":3241}}
```

3.3.8 "get_status" request type

System related command, request to send sys_inf message with status information type of "status".

Parameters:

- **type** – string, status type, allowed values: "performance"
- **reset_stat(not implemented)** – Boolean, relevant only for "get_status" command, if true, this status data is reset at the Media Processor. Default value:false

Example:

```
{"sys_req":{
    "req_type":"get_status",
    "req_id":3241,
    "data":{"type":performance", "reset_stat":true}}}
```

3.4 "sys_ans" messages

Message of this type is sent by the Media Processor as a response to the request message. It contains the same "req_id" that was set in the original request message. It also contains req_type field mostly for debugging and readability of the protocol.

Parameters:

- **req_id** – integer, request id, this number is copied from the original tool_req message
- **req_type** – string, represents the original request type

The following subsections are structured according to req_type value.

3.4.1 "set_config" request type

Result of a configuration that was applied

Parameters:

- **error_code** – integer, result of the configuration
- **description** – string, textual description of error_code

Example:

```
{"sys_ans":{
    "req_type":"set_config"
    "req_id":4002,
    "data":{"error_code":0,
```

```
        "description":"ok"}}}
```

3.4.2 "get_config" request type

Contains current system configuration

Parameters: See set_config request message parameters

Example:

```
{"sys_ans":{
     "req_type":"get_config"
     "req_id":4002,
     "data":{"scheduler_mode":"multimedia",
       "status":[{"type":"performance","period":1000}] }}}
```

3.4.3 "get_tool_defaults" request type

Contains default configuration of a tool

Parameters:

- **tool_type** – string, tool's type
- Tool specific parameters – contain tool's default configuration

Example:

```
{"sys_ans":{
     "req_type":"get_tool_defaults",
     "req_id":4002,
     "data":{"tool_type":"h264_decoder",
       "max_width":1920,
       "max_height":1088}}}
```

3.4.4 "get_tools_list" request type

Contains all currently configured tools in the Media Processor

Parameters:

- **tool** – contains information about a single tool (can be more than 1 tool parameters)
    - **id** – integer, unique tool id
    - **type** – string, type of the tool
    - **app_info** – string, optional text information that was set by tool_req::set_config message

Example:

```
{"sys_ans":{
     "req_type":"get_tools_list",
```

```
    "req_id":3004,
    "data":{"tools":[
      {"type":"h264_decoder","id":0,"app_info":"participant 1
 decoder"},
      {"type":"h264_encoder","id":1,"app_info":"participant 1
encoder"},
      {"type":"video_resizer","id":2,"app_info":"participant 1 re-
sizer"}]}
}}
```

3.4.5 "command" request type

Contains a command result

Parameters:

- **cmd_type** – string, command type
- **error_code** – integer, result code of the command
- **description** – string, optional, textual description of the result
- other optional command specific parameters

3.4.5.1 reset command response specific parameters(not implemented)

error_code corresponds to the following descriptions

- 0 – "no error"
- 1 – "internal error"

example:

```
{"sys_ans":{
    "req_type":"command",
    "req_id":4008,
    "data":{"cmd_type":"reset",
      "error_code":0,
      "description":"no error"}}}
```

3.4.6 "get_version" request type

Contains system information like description of all supported tool typed in this Media Processor and density info

Parameters:

- **tools** – array of object, contains information about single tool type that is supported in this Media Processor. Each object includes the following fields:
  - **type** – string, tool type

- **version** - string, Media Processor version name

Example:

```
{"sys_ans":{
     "req_type":"get_version",
     "req_id":4009,
     "data":{
        "supported_tools":[{"type":"rtp_session"},
                {"type":"h264_decoder"},
                {"type":"voice"}]
     },
     "version":"1.1.35 debug3"}}}
```

3.4.7 "get_density" request type(not implemented)

Contains system density information

Parameters: TBD

Example:

```
{"sys_ans":{
     "req_type":"get_version",
     "req_id":4009}}
```

3.4.8 "get_status" request type

Contains current system configuration

Parameters:

- **type** – string, status type to request, currently supported types:
    - **performance** – contains resources utilization info like CPU usage and memory usage.

Example:

```
{"sys_ans":{
     "req_type":"get_status",
     "req_id":4003,
     "data":{"type":"performance"}}}
```

3.5 "group_req" messages(not implemented)

Messages of this type contain a request for certain action that is addressed to multiple tools: all tools with specified group_id. As well as tool_req and sys_req messages, it contain req_id field. As an answer to this message, Media Processor must send group_ans message with the results.

Parameters:

- **req_id** – integer, request id, this number will be sent back to the controller with response message so that the controller will be able to identify the response
- **req_type** – string, represents request type
- **group_id** – integer, specifies group id of tools which will be activated

The following subsections are structured according to req_type value.

3.5.1 "activate" request type

Sets active flag to true for all tools with specific group_id

Parameters: none

Example:

```
{"group_req":{
     "req_type":"activate",
     "req_id":3200,
     "group_id":356}}
```

3.5.2 "deactivate" request type

Sets active flag to false for all tools with specific group_id

Parameters: none

Example:

```
{"group_req":{
     "req_type":"deactivate",
     "req_id":3200,
     "group_id":356}}
```

3.5.3 "remove" request type

Removes all tools with specific group_id

Parameters:

- **notify_for_each_tool** – Boolean, if true, a notification message will be sent for each tool that is removed. Default value: false

Example:

```
{"group_req":{
     "req_type":"remove",
     "req_id":3200,
     "group_id":356,
     "data":{"notify_for_each_tool":true}}}
```

3.6 "tool_inf" messages

These messages are initiated and sent by the Media Processor side. It contains information related to specific tool like event or tools statistics.

Parameters:

- **tool_id** – integer, unique tool identifier
- **app_info** – string, optional info that was set in tool_req::set_config message. If was not set, it will not be included in this message
- **tool_type** – string, the type of the tool
- **inf_type** – string, the information type that is contained in this message

The following subsections are structured according to inf_type value.

3.6.1 "status" information type

Statistics periodically or on demand sent by a tool according to its configuration.

Parameters:

- **status_type** – string, status type
- Tool's specific data – contain tool's statistics

Example:

```
{"tool_inf":{
    "tool_id":0,
    "tool_type":"h264_decoder",
    "inf_type":"status",
    "data":{"type":"general",
      "app_info":"debug info",
      "nof_decoded_frames":457,
      "nof_errors":2,
      "nof_iframes_decoded":23}}}
```

3.6.2 "event" information type

Notification about an event occurred in the tool

Parameters:

- **event_type** – string, type of the event
- Event specific data – contain event information

Example:

```
{"tool_inf":{
    "tool_id":7,
    "tool_type":"rtp_session",
```

```
"inf_type":"event",
"data":{"event_type":"iframe_request",
  "app_info":"debug_info"}}}
```

3.6.3 "tool_removed" information type(not implemented)

Notification about tool removal will be sent by each tool after group_req of "remove" type was sent by the controller only if "notify_for_each_tool" parameter is set to true.

Parameters: none

Example:

```
{"tool_inf":{
    "tool_id":7,
    "tool_type":"rtp_session",
    "app_info":"debug_info",
    "inf_type":"tool_removed"}}
```

3.7 "sys_inf" messages

These messages are initiated and sent by the Media Processor side. It contains system-wide general information.

Parameters:

- **inf_type** – string, the information type that is contained in this message

The following subsections are structured according to inf_type value.

3.7.1 "performance" status message

Performance system statistics sent by the Media Processor periodically

Parameters:

- **"type":"performance"** – for this type of messages
- **CPU** – object, contains CPU usage information, includes the following fields: (These fields are reset after each status report)
  - **CPU_usage** – integer, cpu usage in percents
  - **nof_late_sched_iterations** – integer, number of scheduler iterations where all required tools were not run during the iteration period and therefore certain amount of tools were late.
- **memory_pools –** array of objects, contains memory utilization information, each object contains the following fields (memory pools are extendable and can allocate from the operating system additional blocks when required):
  - **block_size** – integer, size of a single allocation unit for this memory pool.
  - **total_nof_blocks** – integer, total number of memory blocks that are held by this memory pool.

- **nof_free_blocks** – integer, number of memory blocks that are available for allocation inside MediaProcessor.

- additional parameters will be added in future

Example:

```
{"sys_inf":{
     "inf_type":"status",
     "data":{
        "type":"performance",
        "CPU":{"CPU_usage":57, "nof_late_sched_iterations":0},
        "memory_pools":[
          {"block_size":48, "total_nof_blocks":2027,
"nof_free_blocks":45},
          {"block_size":128, "total_nof_blocks":251, "nof_free_blocks":6}
]
}}}
```

3.7.2 "network" status message

Performance system statistics sent by the Media Processor periodically

Parameters:

- **"type":"network"** – for this type of messages
- **packet_loss** – integer, global packet loss counter accumulated from all the tools that receive UDP data from the network, can be used to monitor system under heavy load

Example:

```
{"sys_inf":{
     "inf_type":"status",
     "data":{
        "type":"network",
        "packet_loss":0
 }}}
```

3.8 "group_ans" messages(not implemented)

Message of this type is sent by the Media Processor as a response to the request message. It contains the same "req_id" that was set in the original request message. It also contains req_type field mostly for debugging and readability of the protocol. This response is related to a group of tools.

Parameters:

- **req_id** – integer, request id, this number is copied from the original tool_req message
- **group_id** – integer, group identifier

- **req_type** – string, represents the original request type

The following subsections are structured according to req_type value.

3.8.1 "activate", "deactivate" and "remove" request types

Contains result to the above requests

Parameters:

- **error_code** – integer, result of the action
- **description** – string, textual description of error_code

Example:

```
{"group_ans":{
     "req_type":"activate",
     "req_id":4532,
     "group_id":356",
     "data":{"error_code":0,
        "description":"ok"}}}
```

4. Reserved words

- ***default*** – can be used to set any parameter to its default value
- ***null*** – can be used to connect output of a tool to nowhere, natively supported by JSON standard
- ***true***, ***false*** – for use with boolean fields, natively supported by JSON standard

5. Specific tools API:

Note: tool_getconfig_ans message created by Media Processor uses the same supported parameters from tool_set_req (sent by the controller) message for any tool. The difference is that tool_set_req may contain only part of the parameters when tool_getconfig_ans message contains full configuration of the tool and therefore uses all supported parameters.

5.1 voice_p2p and voice_fe_ip tools

Voice p2p implements half duplex packet to packet functionality. Voice Front-End IP (voice_fe_ip) tool implements full-duplex Voice channel that is connected to IP using RTP from one side and to other tool from the other side. Used to connect to the voice_mixer tool from internal side.

5.1.1 Tool configuration

Parameters:

- **dst_tool_ids** – as described in section 3.1.1, relevant only for "voice_fe_ip" tool
- **input_from_RTP** – boolean, if set to true, the voice source is taken from RTP (UDP socket) otherwise it is taken from tool's input (and therefore is received from other tools). Should be set to false when receiving voice from file. Default value: true.
- **encoder** – object, represents encoder configuration, contains the following fields:
    - **type** – string, encoder coding standard, supported values: "linear"/"G.711alaw"/G.711ulaw"/"G.729"/"AMR_NB"/"AMR_WB". Default value:"G.711alaw"
    - **packet_duration** – integer, duration of encoded voice in each packet in milliseconds. Range: 0 – 60. Default value: 10
    - **wait_for_decoder** – boolean, if set to true, encoder does not send any data before a valid frame was received by the decoder.
    - **rate** – string, applicable only for AMR Narrow Band/Wide Band codecs, specified encoder AMR rate. Default value:"12.2". NB AMR rates: "4.75", "5.15", "5.90", "6.70", "7.40", "7.95", "10.2", "12.2". WB AMR rates: "6.60", "8.85", "12.65", "14.25", "15.85", "18.25", "19.85", "23.05", "23.85".
    - **packing** – string, applicable only for AMR NarrowBand/WideBand codecs, specifies frame packing mode (octed aligned/bandwidth efficient), supported values: "OA"/"BE". Default value:"OA"
    - **VAD** – object, contains Voice Activity Detector configuration with the following fields:
        - **enabled** - Boolean, enabled VAD functionality in the encoder. Default value: false

- **type** – string, supported values: "none"/"light"/"G.729B". Default value: "none". This field is relevant only for codecs that does not support VAD natively like G.711
- **enable_SID** – boolean, enables SID silence packet sending. Default value: true. This field is relevant only for codecs that does not support VAD natively like G.711

- **decoder** – object, represents decoder configuration, contains the following fields:
  - **decoder_type** – string, decoder coding standard, supported values: "linear"/"G.711alaw"/G.711ulaw"/"G.729"/"AMR_NB"/"AMR_WB". Default value:"G.711alaw"
  - **packing** – string, applicable only for AMR NarrowBand/WideBand codecs, specifies frame packing mode (octed aligned/bandwidth efficient), supported values: "OA"/"BE". Default value:"OA"
  - **PLC** – Boolean, enables Packet Loss Concealment mechanism. Default value: false. Relevant only for codecs that does not support PLC natively like G.711
  - **CNG** – object, contains Comfort Noise Generation configuration, contains the following parameters:
    - **enabled** – boolean, enable/disable CNG, Default value:false
    - **level_shift** – integer, Shift CNG output level in dB. Range of values: -72.. 72. Default value:0
    - **CNI_enable** – boolean, enable/disable Comfort Noise Injection (injection of comfort noise regardless of SID packet receive, depending on input signal level). Default value: false
    - **CNI_min_level** – integer, input signal level threshold for injection of comfort noise. Range of values: 0 - 32767. Default value: 0
    - **CNI_level** – integer, comfort noise to be injected if the input signal level is below CNI_min_level. Range of values: 0 – 255. Default value:0
- **RTP** – object, contains various rtp configuration parameters:
  - **local_udp_port** – integer, RTP will listen on this port to incoming stream
  - **remote_udp_port** – integer, RTP will send output stream to this destination port
  - **remote_ip** – string, remote RTP side IP address in format “a.b.c.d”
  - **tx_ifc** – string, name of linux ethernet interface (the only interface) that will be used for RTP transmit. If set to empty string, the interface will be chosen automatically by the operating system. Default value: "".
  - **out_payload_type** – integer, configure dynamic RTP payload type for the encoder, Default value: taken from RTP payload static table.
  - **in_payload_type** – integer, configure dynamic RTP payload type for the decoder, Default value: taken from RTP payload static table.
  - **dtmf_in_payload_type** – integer, dynamic payload type for received RFC2833, Default value:96
  - **dtmf_out_payload_type** – integer, dynamic payload type for sent RFC2833, Default value:96

- **auto_rtp** – string, determines how to handle SSRC collision. Value:"manual_statistics_not_reset"/"auto_statistics_not_reset"/"manual_statistics_reset"/"auto_statistics_reset". Default value: "auto_statistics_reset"
- **mode** – string, configures RTP mode of operation, valid values: "end_system"/"translator". Default value:"end_system"
- **header** – object, defines configuration of rtp initial header fields
  - **ssrc** – integer, RTP SSRC value that will be used for outgoing RTP stream. If not specified, will be generated randomly.
  - **seq** – integer, initial RTP sequence number for the outgoing stream. If not specified, will be generated randomly.
  - **timestamp** – integer, initial RTP time stamp for the outgoing stream. If not specified, will be generated randomly.
- **duplicate_stream** – array of objects, each object defines one destination that will receive replication of voice data. Default value: empty. Each object includes the following fields:
  - **data_src** – string, “ip_in”/”ip_out”/”samples_in”/”samples_out”
  - **remote_ip** – string, IP of the destination in “a.b.c.d” format
  - **remote_udp_port** – integer, destination UDP port
  - **payload_type** – integer, payload type that will be used in the stream for this destination

- **RTCP** – object, contains RTCP configuration. Consists of the following fields:
  - **rx_enabled** – Boolean, enable/disable receiving of rtcp frames. Default value: false
  - **report_interval** – integer, period between RTCP reports in seconds. Default value: 5
  - **tx_enabled** – Boolean, enable/disable sending of RTCP frames. Default value: false
  - **local_udp_port** – local UDP port used for RTCP
  - **remote_udp_port** – remote UDP port used for RTCP
  - **coupled_tool_id** – integer, id of the tool which represents the other direction of this RTP session to obtain required statistics for RTCP reports
  - **cname** – string, RTCP cname. Default value: empty string
  - **name** – string, RTCP name. Default value: empty string
  - **email** – string, RTCP email. Default value: empty string
  - **phone** – string, RTCP phone. Default value: empty string
  - **loc** – string, RTCP loc. Default value: empty string
  - **tool** – string, RTCP tool. Default value: empty string
  - **note** – string, RTCP note. Default value: empty string
- **jitter_buffer** – object, contains jitter buffer configuration. Consists of the following fields:

- **init_delay** – integer, JB initial delay in milliseconds. Range of values: 0 – 300. Default value: 70
- **adaptation_type** – string, type of the adaptation algorithm: "none"/"full_adaptive"/"short_adaptive"/"non_managed_network". Default value: "short_adaptive"
- **depth** – integer, jitter buffer depth in milliseconds. Range of values: 0 – 300. Default value:200
- **min_delay** – integer, minimal delay in milliseconds, used by adaptation algorithm. Range of values: 0 – 300. Default value: 40
- **max_delay** – integer, maximal delay in milliseconds, used by adaptation algorithm. Range of values: 0 – 300. Default value: 100
- **short_adapt_fraction** – integer, defines trigger for short run adaptation – if, out of the last twenty packets received, the number of packets falling outside the min_delay and max_delay is equal or greater than the number specified in this field, then the short run adaptation will occur. Range of values: 0 -20. Default value:7

- **AGC** – object, contains Automatic Gain Control configuration. In "voice_fe_ip" tool this object includes 2 sub-objects:"encoder_side" and "decoder_side" where each sub-object includes the following fields. encoder_side applies to the samples that are coming from this tool input and are sent to the RTP side. decoder_side applied to the samples that are received from the IP (using RTP) and are sent to the tool defined by "dst_tool_id". In "voice_p2p" tool this object includes the following fields:
  - **enabled** – Boolean, enables/disables AGC. Default value: false
  - **energy_avg_window** – integer, the energy averaging window length in milliseconds. Range: 50-1000. Default value: 100
  - **min_signal_level** – integer, minimal signal level in dBm0 units, if the signal is lower, it will be amplified to this level. Range: -37.. -6. Default value: -14
  - **max_signal_level** – integer, maximal signal level in dBm0 units, if the signal level is higher – it will be attenuated to this level. Range -37.. -6. Default value: -10
  - **step_level** – integer, the gain step level in 0.1 dB/sec units. Range 0 – 127. Default value: 10
  - **silence_threshold** – integer, threshold for silence detection in dBm0 units, signal with energy lower than this level will not be part of energy calculation and will not be amplified. Range: -60.. -30. Default value: -30
  - **limit_gain** – Boolean, determines if max_gain value is applied and should be used. Default value: false
  - **max_gain** – integer, the maximum value the applied gain may take in 1db units. Range: -31.. 18. Default value: 18
- **EVG** – object, defines EVent Generator configuration. In "voice_fe_ip" tool this object includes 2 sub-objects:"encoder_side" and "decoder_side" where each sub-object includes the following fields. encoder_side applies to the samples that are coming from this tool input and are sent to the RTP side. decoder_side applied to the samples that are received from the IP (using RTP) and are sent to the tool defined by "dst_tool_id". In "voice_p2p" tool this object includes the following fields:

- **enabled** – Boolean, enables/disables event generator. Default value: false
- **host_override** – Boolean, determines if host event generation command can override automatic generation by EVG module. Default value: false.
- **delay_or_delete** – string, determines what EVG will do with the less preffered event that should be generated at the same time. Range of values: "delay", "delete". Default value: "delay".
- **convert_RTP_events_to_inband** – string, automatically generate inband event if corresponding event was received from RTP. Allowed values: "none", "CPS", "modem", "all". Default value: "none".
- **convert_inband_events_to_RTP** – Boolean, automatically generate RTP events when a corresponding event was detected in-band. Default value: false.
- **relay_RTP_events** – Boolean, relay input RTP events to RTP output. Default value: false.

• **EVD** – object, defined EVent Detector configuration. Contains the following fields:

- **enabled** – Boolean, enables/disables EVD. Default value: false.
- **events** – array of strings, defines event types that will be detected by EVD. Default value: empty. Supports the following values: DTMF_GROUP, MF_GROUP, MFC_FWD_GROUP, MFC_BACK_GROUP, ANS, ANS_BAR, ANSAM, ANSAM_BAR, CNG, CNG_UNDER_NOISE, CT, V21FLAG, SS7, SS7DUAL, V8BIS_INI, V8BIS_RES, CAS, V22, SILENCE, V32_AA_AC, GR30, V23, VOICE
- **tone_suppression** – string, defines type of tone suppression that will be applied to the voice samples. Allowed values: "none"/"delay_suspected"/"no _delay"/"delay_constant". Default value: "none".
- **detection_params** – object, contains advanced detection parameters, includes the following fields:
  - **mf_min_power** – integer, Default value: -31.
  - **mf_rigorous** – Boolean, Default value: false.
  - **dtmf_ttc** – Boolean, Default value: false.
  - **voice_zc_max_std** – integer, Default value: 6.
  - **voice_holdover_duration** – integer, Default value: 100.
  - **voice_min_duration** – integer, Default value: 170.
  - **voice_special_mode** – Boolean, Default value: false.
- **used_defined_events** - array of objects where each object defines one user defined event. Default value: empty. Each object contains the following fields:
  - **name** – string, event that will be defined. Allowed values: "USER_DEF1" .. "USER_DEF8".
  - **description** – string, event description. Allowed values: "undefined", "dial_tone", "ringback_tone", "busy_tone", congestion_tone", "information_tone".
  - **duration_tolerance** – integer
  - **min_power** – integer

- **type** – string, Allowed values: "single", "repeating", "continuous".
- **sequence** – array of objects, where each object defines one item out of the event sequence. Each object contains the following fields:
  - **frequency** – array of 2 integers representing 2 frequencies.
  - **duration** – integer.

5.1.2 Commands

5.1.2.1 Stop event generation

Stops event generation by the EVG module

Parameters:

- **"cmd_type":"stop_tone_generation"**

5.1.2.2 Generate voice events command

Generates either "in band" or "out of band" voice events

Parameters:

- **"cmd_type":"generate_tone"**
- **total_duration** – integer/string, total duration of the events to generate in millisecond. If this value exceeds the sum of all events durations, the generator will start generating event from the beginning of the sequence until required total duration is reached. Can be also set to string "infinite".
- **rtp_or_inband** – string, defines whether this event will be generated in-band or out-of-band. Allowed values:"RTP","inband"
- **tone_type** – string, relevant only if rtp_or_inband is set to "inband".
  allowed values: "predefined","user_defined"
- **ip_event** – object, relevant only if rtp_or_inband is set to "RTP". Contains the following fields:
  - **named_event** – string, allowed values: DTMF0..DTMF15, MF0..MF9,MF_CODE11, MF_KP, MF_KP2, MF_ST, MF_CODE12, MFC_FWD1..MFC_FWD15, MFC_BACK1..MFC_BACK15, ANS, ANS_BAR, ANSAM, ANSAM_BAR, CNG, CNG_UNDER_NOISE, CT, V21FLAG, SS7, SS7DUAL, V8BIS_INI, V8BIS_RES, CAS, V22, SILENCE, V32_AA_AC, GR30, V23, VOICE
- **predefined_event** – object, relevant only if rtp_or_inband is set to "inband" and tone_type is set to "predefined". Contains the following fields:
  - **amplitude1** - integer
  - **amplitude2** – integer
  - **on_duration** – integer/string. Can be also set to string "infinite".
  - **off_duration** – integer/string. Can be also set to string "infinite".

- **digits** – array of strings, each item represents a single item from the event sequence. Allowed values: see "named_event" values. Max size of the array is 32.

- **user_defined_event** – array of objects, relevant only if rtp_or_inband is set to "inband" and tone_type is set to "user_defined". Each array item represents a single item from the event sequence. Max array length is 4. Contains the following fields:
  - **freq1** – integer
  - **freq1** – integer
  - **level1** – integer
  - **level2** – integer
  - **duration** – integer/string, length of the item in milliseconds. Can be also set to string "infinite".
  - **cadence_or_am** – string, defined how the item will be described. Allowed values: "cadence", "am".
  - **on_duration** – integer/string, relevant only when cadence_or_am is set to "cadence". In milliseconds unit. Can be also set to string "infinite".
  - **off_duration** – integer/string, relevant only when cadence_or_am is set to "cadence". In milliseconds unit. Can be also set to string "infinite".
  - **off_type** – string, relevant only when cadence_or_am is set to "cadence". Allowed values: "silence", "transparent".
  - **am_freq** – integer, relevant only when cadence_or_am is set to "am".
  - **am_depth** – integer, relevant only when cadence_or_am is set to "am".
  - **phase_reversal_period** - integer, relevant only when cadence_or_am is set to "am".

5.1.3 Events

5.1.3.1 "inband_event_detected" event

This event is sent when the Event detector module detects an event.

Parameters:

- **“type”:”inband_event_detected”**
- **"event"**- string, event type that was detected by the EVD. Can be one of the following: MF0..MF9,MF_CODE11, MF_KP, MF_KP2, MF_ST, MF_CODE12, MFC_FWD1..MFC_FWD15, MFC_BACK1..MFC_BACK15, ANS, ANS_BAR, ANSAM, ANSAM_BAR, CNG, CNG_UNDER_NOISE, CT, V21FLAG, SS7, SS7DUAL, V8BIS_INI, V8BIS_RES, CAS, V22, SILENCE, V32_AA_AC, GR30, V23, VOICE, USER_DEF1..USER_DEF8

5.1.3.2 "RTP_event_detected" event

This event is sent when RTP tone is received from the network

Parameters:

- **“type”:”RTP_event_detected”**
- **"event"**- string, event type that was detected by the EVD. Can be one of the following: MF0..MF9,MF_CODE11, MF_KP, MF_KP2, MF_ST, MF_CODE12, MFC_FWD1..MFC_FWD15, MFC_BACK1..MFC_BACK15, ANS, ANS_BAR, ANSAM, ANSAM_BAR, CNG, CNG_UNDER_NOISE, CT, V21FLAG, SS7, SS7DUAL, V8BIS_INI, V8BIS_RES, CAS, V22, SILENCE, V32_AA_AC, GR30, V23, VOICE

5.1.3.3 "RTCP_received" event

This event is triggered when RTCP packet is received from the network. Only those fields are sent in the message that were received in RTCP packet

Parameters:

- **“type”:”RTCP_received"**
- **sender_SSRC** – integer
- **NTP_timestamp_MSW** – integer
- **NTP_timestamp_LSW** – integer
- **RTP_timestamp** – integer
- **sender_packet_count** – integer
- **sender_octet_count** – integer
- **received_SSRC** – integer
- **fraction_lost** – integer
- **cumulative_packet_lost** – integer
- **ext_high_seq_recv** – integer
- **interarrival_jitter** – integer
- **last_SR_ts** – integer
- **delay_since_last_SR_ts** – integer
- **cname** – string
- **name** – string
- **email** – string
- **phone** – string
- **loc** – string
- **tool** – string
- **note** - string

5.1.3.4 "RTCP_sent" event

This event is triggered when RTCP packet is sent to the network

Parameters:

- **“type”:”RTCP_sent"**

- The same as in RTCP_received event

5.1.3.5 "AMR_Codec_Mode_Request" event

This event is triggered CMR field in incoming AMR RTP packet is differs from its older value

Parameters:

- **“type”:”AMR_Codec_Mode_Request"**
- **"rate"** – string, contains rate that was requested

5.1.4 Statistics

5.1.4.1 RTP

Includes RTP related statistics.

Parameters:

- **“type” : ”RTP”**
- **PT_received** – integer, RTP payload type that was received from the far end.
- **SSRC_received** – RTP ssrc that was received from the remote side
- **received_packets** – integer, number of received packets
- **received_bytes** – integer, total number of received bytes
- **SSRC_sent** – integer, SSRC that is used for sending outgoing RTP stream
- **SSRC_collision** – Boolean, set to true when SSRC collision has occurred
- **sent_packets** – integer, number of sent packets
- **sent_bytes** – integer, total number of bytes sent
- **tx_errors** – integer, number of errors in sending

5.1.4.2 jitter_buffer

Jitter buffer related statistics.

Parameters:

- **“type” : ”jitter_buffer”**
- **too_early_packets_count** – integer, number of packets that were dropped due to too early arrival
- **too_late_packets_count** – integer, number of packets that were dropped due to too late arrival
- **packets_received** – integer, number of packets received by the jitter buffer
- **average_delay** – integer, jitter buffer average delay
- **adaptation_count** – integer, number of jitter buffer adaptations

5.1.4.3 decoder

Decoder related statistics.

Parameters:

- **“type” : ”decoder”**
- **errors** - integer, number of errors in decoder
- **frames_decoded** – total number of frames that were decoded
- **AMR_rate_received** – when decoder is configured to AMR, this parameter will contain the AMR rate that was actually received by the decoder

5.1.4.4 encoder

Encoder related statistics.

Parameters:

- **“type” : ”encoder”**
- **errors** - integer, number of errors in decoder
- **frames_encoded** – total number of frames that were encoded

5.1.4.5 EVG

Event generator related statistics.

Parameters:

- **“type” : ”EVG”**
- **is_active** – Boolean, is set to true if EVG is generating tone at this very moment

5.2 voice_mixer tool

voice_mixer tool is capable of creating voice conference of several participants. Each participant is can be represented by voice_fe_ip tool

5.2.1 Tool configuration

Parameters:

- **sampling_rate** – integer, sampling rate of the voice that is used in this conference, supported values: 8000, 16000. Default value:8000.
- **dominant_speakers** – integer, max number of concurrent "regular" speakers that can be mixed together. This number does not include the participants declared as "dominant". Supported values:0-5. Default value: 3.
- **hangover_period** – integer, period in milliseconds that represents the following: when a participant become a "dominant speaker", it will not be replaced by another participant even if does not produce any speech until this timeout is expired. Supported values:0-1000, Default value:100

- **participants** – array of objects where each object defines one participant and contains the following fields:
    - **id** – integer, identification number of the participant which can be used later to refer to this participant. Valid values: . Default value: .
    - **type** – string, participant type. Supported values: "regular","listener","dominant","whisper". Default value: "regular".
        - "regular" – standard participant that can speak and be one of the current speakers out of "dominant_speakers" number.
        - "listener" – only receives the conference output, but cannot speak.
        - "dominant" – participant whose voice is always mixed into the conference output regardless of its energy level.
        - "whisper" – participant that can only "whisper" to another participant but cannot speak to the whole conference.
    - **tool_id** – integer, tool ID of the front-end voice tool that represents the participant
    - **whisper_to** – integer, valid only if the type is set to "whisper". ID of the participant that this participant wants to whisper to.
    - **action** - string, action "add"/"remove" – add/update new/existing participant or remove existing participant. Valid values: "add","remove". Default value:"add".

5.2.2 Statistics

5.2.2.1 speakers

Includes statistics related to active speakers at the same moment. Does not include "always dominant" type speakers.

Parameters:

- **“type” : ”speakers”**
- **dominant_speakers** – array of objects. Each object contains info about single dominant speakers. Each object consists of the following fields
    - **id** – integer, id of the participant
    - **energy** – integer, measured energy of the participant voice signal

5.3 file_reader tool

File reader implements playing a list of supported file formats, one after the other.

5.3.1 Tool configuration

Parameters:

- **dst_tool_ids** – as described in section 3.1.1

Example:

```
{"tool_req":{
    "req_type":"set_config",
    "req_id":111,
    "tool_id":1,
    "data":{
        "tool_type":"file_reader",
        "dest_tool_ids":[2]}
}}
```

5.3.2 Commands

5.3.2.1 Playlist append

Append a new playlist of files

Parameters:

- **"cmd_type":"play_list_append"**
- **"files" -** array of objects. Each object contains info about a single file to play in the list
  - **name** – string, the file name
  - **duration** – float, number of seconds to play the file.
    if set to -1 the entire file will be played. Default value: -1

Example:

```
{"tool_req":{
    "req_type":"command",
    "req_id":222,
    "tool_id":1,
    "data":{
       "cmd_type":"play_list_append",
       "files":[{"name":"one.mp4", "duration":1.2},
                {"name":"two.mp4"},
                {"name":"three.mp4"}]
    }
}}
```

5.3.2.2 Playlist clear

Stops and clears a playlist

Parameters:

- **"cmd_type":"play_list_clear"**

Example:

```
{"tool_req":{
    "req_type":"command",
    "req_id":222,
```

```
        "tool_id":1,
        "data":{
           "cmd_type":"play_list_clear"
        }
    }}
```

5.3.2.3 Play file command

Starts playing files from the playlist

Parameters:

- **“cmd_type”:”play”**

Example:

```
    {"tool_req":{
        "req_type":"command",
        "req_id":222,
        "tool_id":1,
        "data":{
           "cmd_type":"play"
        }
    }}
```

5.3.2.4 Pause file command

Pauses playing files from the playlist.

Start playing again by *play* command

Parameters:

- **“cmd_type”:”pause”**

Example:

```
    {"tool_req":{
        "req_type":"command",
        "req_id":222,
        "tool_id":1,
        "data":{
           "cmd_type":"pause"
        }
    }}
```

5.3.3 Events

5.3.3.1 "File finished" event

This event is sent whenever a file from playlist configuration were finished playing.

Parameters:

- “**event_type”:”end_of_file”**
- **"file_name"** – string, the name of the file finished

- **"next_file_name"** – string, the name of the next file start being played

Example:

```
{"tool_inf":{
    "tool_id":1,
    "inf_type":"event",
    "data":{
       "event_type":"end_of_file",
       "file_name":"one.mp4",
       "next_file_name":"two.mp4"
    }
}}
```

5.3.3.2 "Playlist finished" event

This event is sent when all the files from playlist configuration were played.

Parameters:

- "**event_type":"end_of_play_list"**

Example:

```
{"tool_inf":{
    "tool_id":1,
    "inf_type":"event",
    "data":{
       "event_type":"end_of_play_list"
    }
}}
```

5.3.3.3 "File info" event

This event is sent whenever next file in the playlist is opened, before the voice channel is configured and processes. This information will be used for the configuration of the voice channel

Parameters:

- **"event_type":"file_info"**
- **"file_name"** – string, the name of the file
- **"codec_type"** - string, decoder coding standard, supported values: "linear"/"G.711alaw"/G.711ulaw"/"G.729"/"AMR_NB"/"AMR_WB". Default value:"G.711alaw"

Example:

```
{"tool_inf":{
    "tool_id":1,
    "inf_type":"event",
    "data":{
       "event_type":"file_info",
       "file_name":"one.wav",
       "codec_type":"G.711alaw"
```

```
        }
    }}
```

5.3.3.4 "Play started" event

This event is sent when the first frame of a file is actually inserted into voice tool for playing

Parameters:

- **“event_type”:”play_started”**
- **"file_name"** – string, the name of the file

Example:

```
{"tool_inf":{
    "tool_id":1,
    "inf_type":"event",
    "data":{
       "event_type":"play_started",
       "file_name":"one.wav"
    }
}}
```

5.3.3.5 "Error playing file" event

This event is sent when next file in the playlist is opened.

Parameters:

- **“event_type”:”error_reading_from_file”**
- **"file_name"** – string, the name of the file that an error occurred while playing it

Example:

```
{"tool_inf":{
    "tool_id":1,
    "inf_type":"event",
    "data":{
       "event_type":"error_reading_from_file",
       "file_name":"one.mp4"
    }
}}
```

5.4 opus_decoder tool

Opus audio decoder; receives either packetized rtp data from rtp_session tool or OPUS frames in from file reader tool. This tool decodes the input stream and creates audio frames in linear16 format (16 bits per sample); these frames are passed to the destination tools that are configured in set_config message.

5.4.1 Tool configuration

Parameters:

- **dst_tool_ids** – as described in section 3.1.1
- **out_sampling_rate** – integer, sampling rate of the output stream, if this value differs from input stream's sampling rate, the rate conversion algorithm will be applied automatically. If this parameter is not set or set to 0, the original input's sampling rate will not be changed. Valid values: 8000, 12000, 16000, 24000, 48000. Default value: 0
- **out_mono_stereo** – enumerator, accepts values "mono"/"stereo". Specifies output's stream channel configuration. If this configuration differs from the input stream, it will be converted automatically. If not set, the original stream configuration will not be changed.

5.5 opus_encoder tool

Opus audio encoder; receives audio frame in linear16 format (for example from opus_decoder or liner_decoder), encodes it, generates opus frame and passes it to the destination tools that were configured in set_config message.

5.5.1 Tool configuration

Parameters:

- **dst_tool_ids** – as described in section 3.1.1
- **out_sampling_rate** - integer, sampling rate of the output stream, if this value differs from input stream's sampling rate, the rate conversion algorithm will be applied automatically. If this parameter is not set or set to 0, the original input's sampling rate will not be changed. Valid values: 8000, 12000, 16000, 24000, 48000. Default value: 0
- **out_mono_stereo** - enumerator, accepts values "mono"/"stereo". Specifies output's stream channel configuration. If this configuration differs from the input stream, it will be converted automatically. If not set, the original stream configuration will not be changed.
- **bit_rate** – object, includes various parameters for bit-rate configuration, contains the following fields:
  - **rate** – integer, bit-rate in bits per second of the output stream. Valid values: 6000 – 512000. Default value: 128000.
  - **BRC** – enumerator, Bit Rate Control algorithm; valid values are "CBR"/"VBR" – Constant Bit-Rate and Variable Bit-Rate. Default value: "CBR".

5.6 linear_decoder tool

Audio linear16 decoder; receives either packetized RTP data (for example from rtp_session tool) or audio frame in linear16 format (for example from file reader tool). In case of RTP data this tool extracts linear16 frame and converts it, if needed, to the required format(sampling rate, mono/stereo). In case of input linear16 audio frame this tool just converts, if needed, the input frame to the required format (sampling rate, mono/stereo). The output frame is passed to the destination tools that are configured in set_config message.

5.6.1 Tool configuration

Parameters:

- **dst_tool_ids** – as described in section 3.1.1
- **out_sampling_rate** - integer, sampling rate of the output stream, if this value differs from input stream's sampling rate, the rate conversion algorithm will be applied automatically. If this parameter is not set or set to 0, the original input's sampling rate will not be changed. Valid values: 8000, 12000, 16000, 24000, 48000. Default value: 0
- **out_mono_stereo** - enumerator, accepts values "mono"/"stereo". Specifies output's stream channel configuration. If this configuration differs from the input stream, it will be converted automatically. If not set, the original stream configuration will not be changed.

5.7 rtp_session tool

RTP session is a full duplex tool that is responsible for RTP and RTP handling.

In one direction it receives UDP packets, for example from udp_socket tool, handles RTP header, inserts the payload into JB and fetches it towards output, for example rfc6184_depacketizer tool. In the other direction it receives RTP payload, adds RTP header and sends it, for example to udp_socket tool. In addition it handles RTCP: it receives RTCP packet and notifies the controller about that. In the other direction it generates RTCP packets according to its configuration and passes the RTCP packet to another tool, for example udp_socket.

5.7.1 Tool configuration

Parameters:

- **RTP_in** – object, contains configuration of the receive side of RTP
  - **enabled** – Boolean, enables/disables receive side of RTP. Default value: true
  - **dst_tool_ids** – as described in section 3.1.1
  - **payload_type** – integer, RTP payload type of the incoming stream. Default value: 96
- **RTP_out** – object, contains configuration of the transmit side of RTP
  - **enabled** – Boolean, enabled/disables transmit side of RTP. Default value: true
  - **dst_tool_ids** – as described in section 3.1.1
  - **payload_type** – integer, RTP payload type of the outgoing stream. If not set, any incoming payload type will be accepted
  - **init_seq_num** – integer, initial RTP sequence number of the RTP stream. Default value: random value
  - **ssrc** – integer, RTP SSRC used in the outgoing stream. Default value: random value.
  - **init_timestamp** – integer, initial RTP timestamp used in the outgoing stream. Default value: random value.

- **RTCP_in (not implemented)** – object, contains RTCP configuration for receive direction. Consists of the following fields:
  - **enabled** – Boolean, enable/disable receiving of rtcp frames. Default value: false
- **RTCP_out (not implemented)** – object, contains RTCP configuration for transmit direction. Consists of the following fields:
  - **enabled** – Boolean, enable/disable sending of RTCP frames. Default value: false
  - **dst_tool_ids** – as described in section 3.1.1
  - **report_interval** – integer, period between RTCP reports in seconds. Default value: 5
  - **cname** – string, RTCP cname. Default value: empty string
  - **name** – string, RTCP name. Default value: empty string
  - **email** – string, RTCP email. Default value: empty string
  - **phone** – string, RTCP phone. Default value: empty string
  - **loc** – string, RTCP loc. Default value: empty string
  - **tool** – string, RTCP tool. Default value: empty string
  - **note** – string, RTCP note. Default value: empty string

5.7.2 Commands

5.7.2.1 send_i_frame_request (not implemented)

- additional parameters: none

5.7.3 Events

5.7.3.1 "RTCP received" event (not implemented)

This event is sent when RTCP packet is received from the far end.

Parameters (seeRFC 3550 for detailed description of parameters):

- **“event_type”:”tone_generation_finished**
- **send_ssrc** – integer
- **NTP_timestamp** - integer
- **RTP_timestamp** - integer
- **sender_packet_count** - integer
- **sender_octet_count** - integer
- **receive_ssrc** - integer
- **frac_lost** - integer
- **total_packet_lost_count** - integer
- **ext_hi_seq_recv** - integer

- **interarrival_jitter** - integer
- **last_SR** - integer
- **delay_since_last_SR** - integer
- **sdes_strings** – array of strings

5.7.4 Statistics

5.7.4.1 RTP

Includes RTP related statistics.

Parameters:

- **"status_type" : "RTP_stat"**
- **sent_packet_count** – integer, number of sent packets
- **sent_bytes_count** – integer, total number of bytes sent
- **outgoing_bitrate** – ingerer, outgoing bit rate in bits per second
- **received_ssrc** – RTP ssrc that was received from the remote side
- **received_packet_count** – integer, number of received packets
- **received_bytes_count** – integer, total number of received bytes
- **incoming_bitrate** – integer, incoming bit rate in bits per second
- **packet_loss_count** – integer, number of lost packets
- **unrecognized_PT** – integer, number of packets that were dropped due to unrecognized payload type

5.8 udp_socket

Tool that encapsulates the UDP socket functionality. On the receive side it listens on UDP socket and passes incoming UDP packets to other tools that were configured in udp_socket tool configuration. On the transmit side it receives UDP packets from other tools, for example rtp_session, and sends it to the network

5.8.1 Tool configuration

Parameters:

- **dst_tool_ids** – as described in section 3.1.1
- **local_udp_port** – integer, specifies UDP port to listen for incoming packets
- **remote_udp_port** – integer, specifies remote UDP port that will be used to send the outgoing packets
- **remote_ip** – string, specifies the remote IP where the outgoing packets will be sent
- **eth_ifc** – string, specifies local linux ethernet device name, for example "eth0" that the UDP socket will be bind to. If set to empty string, the socket will not be bind to any

interface and the operating system will decide which interface to use. Default value: empty string.

5.8.2 Events

5.8.2.1 "source_address_changed" event

This event is sent when the incoming stream's source IP address or source UDP port change. This event is also sent when receiving UDP packet for the first time.

Parameters:

- **“event_type”:”source_address_changed"**
- **ip** – string, source IP in format "A.B.C.D"
- **port** – integer, source UDP port of the incoming stream

5.8.3 Statistics

5.8.3.1 general

General statistics.

Parameters:

- **“status_type” : ”general”**
- **sent_packets** – integer, total number of sent packets from the initialization of the tool
- **received_packets** – integer, total number of received packets from the initialization of the tool

5.9 h264_decoder tool (not implemented)

Receives full encoded frame, decodes it and sends it to output. If the incoming resolution does not match the configured width and height, resize it to the configured resolution. This tool can receive its input from rtp_session tool (packetized data) or from file reader tool (encoded frame not in packetized format)

5.9.1 Tool configuration

Parameters:

- **dst_tool_ids** – as described in section 3.1.1
- **out_width** – integer, defines width of the output frame, if it does not match the incoming width, the frame will be resized. If set to 0, the output will not be resized. Default value: 0
- **out_height** – integer, defines height of the output frame, if it does not match the incoming height, the frame will be resized. If set to 0, the output will not be resized. Default value: 0

5.9.2 Events

5.9.2.1 "new_stream" event

This event is sent when a stream with different parameters is received by the decoder or when extracting parameters for the first time

Parameters:

- **“event_type”:”new_stream"**
- **height** – integer, incoming frames height
- **width** – integer, incoming frames width

5.9.3 Statistics

5.9.3.1 general

General statistics.

Parameters:

- **“status_type” : ”general”**
- **errors** – integer, number of errors in decoding
- **warnings** – integer, number of decoder warnings
- **received_frames** – integer, number of received encoded frames
- **decoded_frames** – integer, number of decoded frames
- **decoded_i_frames** – integer, number of decoded intra frames
- **decoded_width** – integer, width of the decoded stream
- **decoded_height** – integer, height of the decoded stream
- **framerate** – integer, decoded frame rate in frames per second units

5.10 h264_encoder tool (not implemented)

Receives YUV frame, encodes it to h.264 format and sends to output. Input YUV frames can be obtained from decoder tools or video mixer tools.

5.10.1 Tool configuration

Parameters:

- **dst_tool_ids** – as described in section 3.1.1
- **bitrate** – integer, output stream bit rate in bits per second units. Default value:768000
- **in_framerate** – integer, input stream framerate, if not equal to out_framerate, the framerate conversion algorithm will be applied on the stream to convert in_framerate to

out_framerate. If out_framerate is not set or set to 0, in_framerate will be used to specify the output stream framerate. Default value: 0

- **out_framerate** – integer, frame rate of the outgoing stream in frames per second units. If this value differs from the actual frame rate received by the encoder, the tool will perform frame rate modification. If set to 0, no frame rate modification will be done. Default value:0
- **profile** – string, h.264 profile, allowed values: "baseline"/"main"/"high". Default value:"baseline"
- **level** – string, h.264 level, allowed values:"3.0" - "5.0". Default value:"3.0"
- **out_width** – integer, width of the outgoing stream's frames. If it does not match the incoming width, the frame will be resized prior to encoding. If set to 0, the incoming width will not be changed. Default value: 0.
- **out_height** – integer, height of the outgoing stream's frames. If it does not match the incoming height, the frame will be resized prior to encoding. If set to 0, the incoming height will not be changed. Default value: 0.
- **bitrate_control** – string, specifies the bit-rate control algorithm for encoding. Valid values: "VBR","CBR","conference". Default value:"VBR"
  - "VBR" – variable bit rate algorithm
  - "CBR" – constant bit rate algorithm
  - "conference" is a special algorithm optimized for video conferencing scenario – mostly static image with little motion. Default value: "VBR"

5.10.2 Commands

5.10.2.1 generate_idr_frame

- additional parameters: none

5.10.3 Events

5.10.4 Statistics

5.10.4.1 general

General statistics.

Parameters:

- **"status_type" : "general"**
- **errors** - integer, number of errors in encoding
- **encoded_frames** – integer, number of encoded frames
- **encoded_i_frames** – integer, number of encoded I-frames
- **input_width** – integer, width of the incoming YUV frames

- **input_height** – integer, height of the incoming YUV frames
- **framerate** – integer, encoding frame rate in frames per second units

5.11 vp8_decoder tool (not implemented)

Receives encoded vp8 frame, decodes it and sends the result YUV frame to the output

5.11.1 Tool configuration

Parameters:

- **dst_tool_ids** – as described in section 3.1.1
- **out_width** – integer, defines width of the output frame, if it does not match the incoming width, the frame will be resized. If set to 0, the output will not be resized. Default value: 0
- **out_height** – integer, defines height of the output frame, if it does not match the incoming height, the frame will be resized. If set to 0, the output will not be resized. Default value: 0

5.11.2 Commands

5.11.3 Events

5.11.3.1 "new_stream" event

This event is sent when new stream is recognized by the decoder.

Parameters:

- **“event_type”:”new_stream"**
- **height** – integer, incoming frames height
- **width** – integer, incoming frames width

5.11.4 Statistics

5.11.4.1 general

General statistics.

Parameters:

- **“status_type” : ”general”**
- **errors** – integer, number of errors in decoding
- **warnings** – integer, number of decoder warnings
- **received_frames** – integer, number of received encoded frames

- **decoded_frames** – integer, number of decoded frames
- **decoded_i_frames** – integer, number of decoded intra frames
- **decoded_width** – integer, width of the decoded stream
- **decoded_height** – integer, height of the decoded stream
- **framerate** – integer, decoded frame rate in frames per second units

5.12 vp8_encoder tool (not implemented)

Receives YUV frame, encodes it to VP-8 format and sends to output.

5.12.1 Tool configuration

Parameters:

- **dst_tool_ids** – as described in section 3.1.1
- **bitrate** – integer, output stream bit rate in bits per second units. Default value:768000
- **out_framerate** – integer, frame rate of the outgoing stream in frames per second units. If this value differs from the actual frame rate received by the encoder, the tool will perform frame rate modification. If set to 0, no frame rate modification will be done. Default value:0
- **profile** – integer, VP8 profile, allowed values: 0 - 3. Default value:0
- **out_width** – integer, width of the outgoing stream's frames. If it does not match the incoming width, the frame will be resized prior to encoding. If set to 0, the incoming width will not be changed. Default value: 0.
- **out_height** – integer, height of the outgoing stream's frames. If it does not match the incoming height, the frame will be resized prior to encoding. If set to 0, the incoming height will not be changed. Default value: 0.
- **bitrate_control** – string, specifies the bit-rate control algorithm for encoding. Valid values: "VBR","CBR","conference". Default value:"VBR"
    - "VBR" – variable bit rate algorithm
    - "CBR" – constant bit rate algorithm
    - "conference" is a special algorithm optimized for video conferencing scenario – mostly static image with little motion. Default value: "VBR"

5.12.2 Commands

5.12.2.1 generate_idr_frame

- additional parameters: none

5.12.3 Events

5.12.4 Statistics

5.12.4.1 general

General statistics.

Parameters:

- **“status_type” : ”general”**
- **errors** - integer, number of errors in encoding
- **encoded_frames** – integer, number of encoded frames
- **encoded_i_frames** – integer, number of encoded I-frames
- **input_width** – integer, width of the incoming YUV frames
- **input_height** – integer, height of the incoming YUV frames
- **framerate** – integer, encoding frame rate in frames per second units

5.13 video_mixer – TBD