SURF HMP API reference guide

Document version: 1.54
Date: March 2017

SURF Communication Solutions, Ltd.

7 Hamada St., Yoqneam Industrial Park

YOQNEAM – 20667, Israel

Tel: +972 (0)73 714-0700

Fax: +972 (0)4 959-4055

Web site	www.surfsolutions.com,
Email	info@surfsolutions.com

Table of Contents

1. Introduction

1.1 Abstract

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

Media Processor is the process name handling HMP functionalities. For example; voice trans-coding or video mixing. It can best be described as an engine for performing an action.

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

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 the first 4 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 the parameters in the messages have default values. If not specified, the default value will be used. This both high level API when not specifying most of the fields, as well as a 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. The main purpose of this language is to replace heavy XML messages in various configuration protocols that do not use a 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, i.e., only 7 character bytes.

This is done to ensure that the other side supports the same protocol, and prevents a situation where the connection was made by mistake.

Immediately after the "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 establishing the TCP connection, without waiting for the other side's response. If one of the sides does not support any of the parameters, it closes the connection, though 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 an API version number that is supported by the sender. For the current version it is [1,2] (represented as an array of integers with major and minor versions).

The controller side may also include optional parameters 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 the TCP connection.

2.1.2 Disconnect message

"disconnect" message is sent by either controller side or Media Processor side before closing the TCP connection. It is also sent 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:

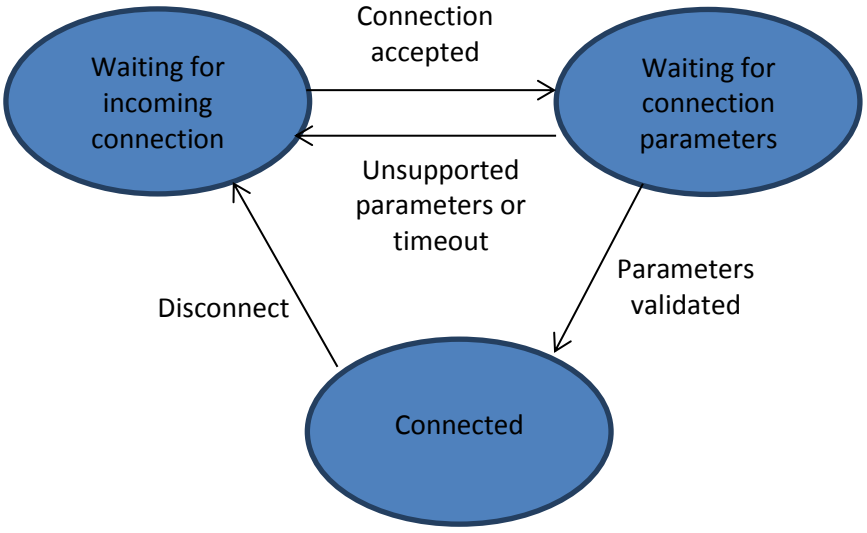
- 0 – "no error"
- 1 – "api version not supported"

"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 the "connect" message was received.

Any side that wishes to close the TCP connection must send a "disconnect" message first.

2.1.3 The Media Processor connection state machine



If the Media Processor enters the "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) if other messages were not sent during this period.

If no messages were received from the remote side during the 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 are enabled. This option is added to enable light weight scripts with Media Processor for testing purposes.

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

3. Message types

All messages in the API are divided into the following groups:

- System wide messages – these messages start with "*sys*_" and be related to the whole system like connect, or get status.
- Tool related messages – these messages start with "tool_" and make an impact only on a specific tool.

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

- Request messages: These messages issue a certain type of request, requires a response, always end with suffix "_req", and are only sent by the controller side.
- Response messages: These messages are sent only as a response to a request message, and end with suffix "_ans".
- Information messages: These messages are sent by the Media Processor to inform the controller of something (event, statistics, etc.). This message does not require a response.

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

3.1 "tool_req" messages

This message always contains a request for a specific tool, such as configure,remove,command, etc. To allow the controller to identify the response to this message (recall: the protocol is asynchronous), it contains a req_id parameter that represents the unique request id. The response message is named "tool_ans", and contains the same req_id parameter found in its body. Another mandatory parameter is the tool_id which identifies the tool handling this message.

Parameters:

- **req_id** – *integer,* request id, this number is sent back to the controller with a response message so controller can 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 tool, or configure an existing tool in the system. The rest of the parameters are default (at tool creation) or unchanged (at tool configuration stage).

Parameters:

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

- **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), and used by several tools. Consists of an array of integers representing unique tool IDs for tools which will receive this tool's output. If not set, output list is considered empty.
  Valid values: 0..65535
- **events** – Array of objects containing 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, if set to *"all"* value, all existing event types for this tool type will be enabled
  - **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, if set to *"all"* value, all existing status types for this tool type will be enabled
  - **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 containing configuration of debug traces. Every trace type represents a separate stream for each tool. If enabled, 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
  - **max_file_size_kb** – *integer,* when log rotate is enabled this parameter specifies max size of a single trace file in kilobytes
  - **max_files_num** – *integer,* specifies max number of trace files for log rotate. If set to 0, log rotate will be disabled.
    Default value: 0.
  - **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}}
```

Note! – tool_id of a tool that was removed can be used again only after remove acknowledge response.

3.1.3 "get_config" request type

Request to get current configuration of a tool.

Parameters: none

Examples:

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

3.1.4 "command" request type

Request for 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

Examples:

```
{"tool_req":{
  "req_type":"get_status",
  "req_id":4007,
  "tool_id":45,
  "data":{"type":"general"}
}}
```

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 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", "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 for controller 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 obtain 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:

- **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 containing configuration of system wide debug traces. If enabled, 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
    - **max_file_size_kb** – *integer,* when log rotate is enabled this parameter specifies max size of a single trace file in kilobytes.
    - **max_files_num** – *integer,* specifies max number of trace files for log rotate. If set to 0, log rotate will be disabled. Default value: 0.
    - **action** – *string,* add/remove trace: "add" opens an empty file and starts recording this trace, "remove" closes the given file.
- Other parameters can be added in the future.

Example:

```
{"sys_req":{
    "req_type":"set_config",
    "req_id":5006,
    "data":{
      status:[{"type":"performance", period:1000},
      {"type":"network", period:500},
      {"type":"video_performance", period:1000}]
      }}}
```

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:
- command specific parameters, depends on command type

3.3.5.1 clear_all_tools

Request to remove all existing tools in the system.

Parameters: none

Example:

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

3.3.6 "get_version" request type

System related command.

Command for sending information about Media Processor, such as supported tool types and software version.

Parameters: none

Example:

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

3.3.7 "get_status" request type

System related command.

Request to send sys_inf message with status information type of "status".

Parameters:

- **type** – *string,* type of the requested status.
  Valid values: "performance", "network", "video_performance"

Example:

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

3.4 "sys_ans" messages

Message sent by Media Processor as response to request message. It contains the same "req_id" that was set in the original request message, and req_type field 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" response

Result of a configuration that was applied by set_config.

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" response

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" response

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" response

Contains all currently configured Media Processor tools.

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" response

Contains the result of a command.

Parameters:

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

Example:

```
{"sys_ans":{
     "req_type":"command",
     "req_id":3004,
     "data":{
        "error_code":0,
        "description":"OK"
}}}
```

3.4.6 "get_version" response

Contains system version information and description of all supported tool types in Media Processor.

Parameters:

- **supported_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":"video_decoder"},
                {"type":"voice_p2p"}]
     },
     "version":"1.1.35 debug3"}}}
```

3.4.7 "get_status" response

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.
    - **network** – contains global network statistics
    - **video_performance** – contains video related global statistics

Example:

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

3.5 "tool_inf" messages

These messages are initiated and sent by Media Processor. They contain information related to specific tools, such as event or tools statistics.

Parameters:

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

The following subsections are structured according to inf_type value.

3.5.1 "status" information type

Periodic or on demand statistics 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}}}
```

3.5.2 "event" information type

Notification about tool event.

Parameters:

- **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":{"type":"iframe_request",
      "app_info":"debug_info"}}}
```

3.6 "sys_inf" messages

These messages are initiated and sent by Media Processor, and contain 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.

3.6.1 "performance" status message

Performance system statistics periodically sent by Media Processor.

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 percentage
    - **nof_late_sched_iterations** – *integer,* number of scheduled iterations where all required tools were not run during the iteration period.
- **memory_pools –** Array of objects containing 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,* number of memory blocks held by memory pool
    - **nof_free_blocks** – *integer,* number of memory blocks available for allocation inside Media Processor
- 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}
]
}}}
```

3.6.2 "network" status message

Network system statistics periodically sent by Media Processor.

Parameters:

- **"type":"network"** – for this type of message
- **packet_loss** – *integer,* global packet loss counter accumulated from all tools receiving UDP data from network. Can be used to monitor system under heavy load. Currently relevant only for voice_p2p and voice_fe_ip tools.

Example:

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

3.6.3 "video_performance" status message

Performance video statistics periodically sent by Media Processor.

Parameters:

- **"type":"video_performance"** – for this type of messages
- **GPU_usage** – *integer,* represents measured GPU utilization in 0-100 percent scale, average value calculated at user level
- **GPU_usage_render** – *float,* measured by driver GPU execution units usage in 0-100 percent scale
- **GPU_usage_video** - *float,* measured by driver GPU video codec module usage in 0-100 percent scale

- **GPU_usage_blitter** – *float,* measured by driver GPU blitter module usage in 0-100 percent scale
- **GPU_usage_video_enhancement** - *float,* measured by driver GPU video enhancement module usage in 0-100 percent scale
- **GPU_usage_video_2** - *float,* measured by driver GPU video codec box 2 (if exists) usage in 0-100 percent scale
- **mixer_missed_frames** – Total number of frames not generated on time by all video_mixer tools in the system.
- **decoder_missed_frames** – Total number of frames dropped by all decoders in the system due to lack of GPU performance resources.
- **encoder_missed_frames** – Total number of frames dropped by all encoders in the system due to lack of GPU performance resources.

Example:

```
{"sys_inf":{
    "inf_type":"status",
    "data":{
       "type":"video_performance",
       "GPU_usage":53,
       "mixer_missed_frames":0,
       "decoder_missed_frames":0,
       "encoder_missed_frames":0
 }}}
```

3.6.4 "license" status message

License information. Contains details about capabilities of current license.

Parameters:

- **"type":"license"** – for this type of message
- **audio_codecs** – *boolean,* informs about the audio codecs capability (enabled/disabled) in the current license.
- **audio_wb_codecs** – *boolean,* informs about the audio wide-band codecs capability (enabled/disabled) in the current license.
- **audio_mixer** – *boolean,* informs about the audio mixer capability (enabled/disabled) in the current license.
- **file_play** – *boolean,* informs about the file play capability (enabled/disabled) in the current license.
- **file_record** – *boolean,* informs about the file record capability (enabled/disabled) in the current license.
- **SRTP** – *boolean,* informs about the SRTP capability (enabled/disabled) in the current license.

- **video_codecs** – *boolean,* informs about the video codecs capability (enabled/disabled) in the current license.
- **video_mixer** – *boolean,* informs about the video mixer capability (enabled/disabled) in the current license.
- **max_end_points** – *integer,* informs about limitation of maximal number of end points in the current license.
- **cur_end_points** – *integer,* informs about number of currently used end points.
- **max_video_codec_tools** – *integer,* informs about limitation of maximal number of video codec tools (video_decoder and video_encoder) in the current license.
- **cur_video_codec_tools** – *integer,* informs about number of currently used video codec tools (video_decoder and video_encoder).
- **max_video_resolution** – *string,* informs about limitation of maximal video resolution in the current license. Valid values: "none","480p","720p","1080p","unlimited"
- **expiration_date** – *string,* contains expiration date of the license or "perpetual" if the license is not time limited. This field contains UTC time in format "DD/MM/YYYY HH:MM:SS"

Example:


```
{"sys_inf":{
    "inf_type":"status",
    "data":{
      "type":"license",
      "audio_codecs":true,
      "audio_wb_codecs":true,
      "audio_mixer":true,
      "file_play":true,
      "file_record":true,
      "SRTP":true,
      "video_codecs":true,
      "video_mixer":true,
      "max_end_points":1000,
      "cur_end_points":123,
      "max_video_codec_tools":100,
      "cur_video_codec_tools":12,
      "max_video_resolution":"1080p",
      "expiration_date":"31/12/2016 23:59:59"
 }}}
```

4. Reserved words

- ***default*** – can be used to set any parameter to its default value
- ***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. tool_set_req may contain only part of the parameters, while tool_getconfig_ans message contains full configuration of the tool and uses all supported parameters.

5.1 voice_p2p and voice_fe_ip tools

Voice packet to packet (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 other tool from the other side.

Used to connect to the voice_mixer tool from internal side or to other voice_fe_ip tool.

5.1.1 Tool configuration

Parameters:

- **input_from_RTP** – *boolean,* if set to true, voice source is taken from RTP (UDP socket). If set to false, it is taken from tool's input (and therefore received from other tools). Should be set to false when receiving voice from file.
  Default value: true
- **output_to_RTP** – *boolean,* if set to true, voice tool will send its encoded frames via RTP.
  Default value: true
- **backend_tool_id** – *integer,* relevant only for voice_fe_ip tool. Specifies the tool id of backend tool that will interact with this tool (full duplex, both sides are linear voice samples). When set to -1, no backend tool will interact with this tool.
  Valid values: (-1) – 0xFFFF
- **after_encoder_dst_tool_ids** – dst_tool_ids that will receive encoded frames after encoding by this tool. As described in section 3.1.1
- **encoder** – *object,* represents encoder configuration, contains the following fields:
  - **type** – *string,* encoder coding standard.
    Valid values: "linear" ,"G.711alaw" ,"G.711ulaw" ,"G.729" ,"G.722" ,"G.7221" ,"G.7231" ,"G.726" ,"AMR_NB", "AMR_WB", "G.711.1", "EVS", ”clear”
    Default value:"G.711alaw"
  - **packet_duration** – *integer,* duration of encoded voice in each packet in milliseconds.
    Valid values: 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
    Default value: false

- **mode** – *string*, specifies encoder mode (relevant only for "G.711.1" encoder type), R1 and R2A are narrow band, R2B and R3 are wide band,
  Valid values: "R1","R2A","R2B","R3"
  Default value: "R3"
- **law** – *string*, specifies G.711.1 coding (relevant only for "G.711.1" encoder type),
  Valid values: "alaw","ulaw"
  Default value: "ulaw"
- **rate** – *string,* specified encoder bit rate

Codec	**Bit Rates Supported**
AMR_NB	"4.75", "5.15", "5.90", "6.70", "7.40", "7.95", "10.2", "12.2". Default: "12.2"
AMR_WB	"6.60", "8.85", "12.65", "14.25", "15.85", "18.25", "19.85", "23.05", "23.85". Default: "23.85"
G.722	Not Relevant. Only one bit rate 64 kbps
G.7221	"16", "24", "32". Default: "32"
G.7231	"5.3", "6.3". Default: "6.3"
G.726	"16", "32". Default: "32"
G.729	Not Relevant. Only one bit rate 8 kbps
G.711alaw, G.711ulaw, Linear	Not Relevant. Only one bit rate 64 kbps
clear	Not Relevant. Only one bit rate 64 kbps
EVS	NB: "7.20", "8.00", "9.60", "12.65", "13.20", "16.40", "24.40" WB: "9.60", "16.40", "24.40", "32.00", "48.00", "64.00", "96.00",
clear	Not Relevant

- **packing** – *string,* specifies frame packing mode

Codec	**Packing Mode Supported**	
AMR_NB,AMR_WB	OA (Default)	Octed Aligned
	BE	Bandwidth Efficient
G.726	LE (Default)	Little Endian
	BE	Big Endian
Others	Not Relevant	

- **sample_rate** – *integer*, specifies at which sampling rate the encoder will function. Relevant only for EVS codec.
  Valid values: 8000, 16000
  Default value: 16000
- **VAD** – *object*, contains Voice Activity Detector configuration with the following fields:
  - **enabled** - *boolean,* enabled VAD functionality in the encoder.
    Default value: false
  - **type** – *string,*
    Valid values: "none", "light", "G.729B"
    Default value: "none" (Relevant only for codecs that don't support VAD natively, i.e., G.711; and if 'enabled' is set)
  - **enable_SID** – *boolean,* enables SID silence packet sending.
    Default value: true (Relevant only for codecs that don't support VAD natively, i.e., G.711; and if 'enabled' is set)

- **decoder** – *object*, represents decoder configuration, contains the following fields:
  - **type** – *string,* decoder coding standard,
    Valid values: "linear" ,"G.711alaw" ,G.711ulaw" ,"G.729" ,"G.722" ,"G.7221" ,"G.7231" ,"G.726" ,"AMR_NB" ,"AMR_WB ","G.711.1", "EVS", "clear".
    Default value:"G.711alaw"
  - **mode** – *string*, specifies encoder mode (relevant only for "G.711.1" encoder type), R1 and R2A are narrow band, R2B and R3 are wide band,
    Valid values: "R1","R2A","R2B","R3"
    Default value: "R3"
  - **law** – *string*, specifies G.711.1 coding (relevant only for "G.711.1" encoder type),
    Valid values: "alaw","ulaw"
    Default value: "ulaw"
  - **rate** – *string*, specifies decoder bit rate

Codec	**Bit Rates Supported**
AMR_NB	Not Relevant. Determined by received framed
AMR_WB	Not Relevant. Determined by received framed
G.722	"48", "56", "64"". Default: "64"
G.7221	"16", "24", "32"". Default: "32"
G.7231	"5.3", "6.3"". Default: "6.3"
G.726	"16", "32". Default: "32"
G.729	Not Relevant. Only one bit rate 8 kbps
G.711alaw, G.711ulaw, Linear	Not Relevant. Only one bit rate 64 kbps
G.711.1	Not Relevant. Determined by codec mode
clear	Not Relevant. Only one bit rate 64 kbps

EVS	Detected automatically, no need to set rate for decoder

- **packing** – *string,* specifies frame packing mode

Codec	**Packing Mode Supported**	
AMR_NB,AMR_WB	OA (Default)	Octed Aligned
	BE	Bandwidth Efficient
G.726	LE (Default)	Little Endian
	BE	Big Endian
Others	Not Relevant	

- **sample_rate** – *integer*, specifies which sampling rate the decoder will accept. Relevant only for EVS codec.
  Valid values: 8000, 16000
  Default value: 16000
- **PLC** – *boolean,* enables Packet Loss Concealment mechanism. Default value: false (Relevant only for codecs that don't support VAD natively, i.e., 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. Valid values: -72.. 72.
    Default value: 0
  - **CNI_enable** – *boolean,* enable/disable Comfort Noise Injection (regardless of SID packet receive, and depending on input signal level).
    Default value: false
  - **CNI_min_level** – *integer,* input signal level threshold for injection of comfort noise.
    Valid values: 0 - 32767
    Default value: 0
  - **CNI_level** – *integer,* comfort noise to be injected if the input signal level is below CNI_min_level.
    Valid values: 0 – 255
    Default value: 0

- **RTP** – *object*, contains various rtp configuration parameters:
  - **local_udp_port** – *integer,* RTP will listen to incoming stream on this port
  - **remote_udp_port** – *integer,* RTP will send output stream to this destination port
  - **override_udp_src_port** – *integer,* outgoing packets source port will be overridden with this value. If not set or set to 0, original source port will be preserved.

Default value: 0
(Note: correct functionality of this feature requires setting valid "local_ip" parameter, and root privileges for Media Processor on HMP machine.)

- **local_ip** – *string,* IP address of interface used to send,receive RTP packets. If not set or set to "0.0.0.0", interface will be chosen automatically.
  Default value: "0.0.0.0"
- **remote_ip** – *string,* remote RTP side IP address in format “a.b.c.d”
- **TOS** – *integer,* Type Of Service byte, that will be used in outgoing RTP packets, as defined in IPv4 specification.
  Default value: 0
- **out_payload_type** – *integer,* configures dynamic RTP payload type for encoder.
  Default value: taken from RTP payload static table
- **in_payload_type** – *integer,* configures dynamic RTP payload type for 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.
  Valid 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, the translator mode is used when the tool receives RTP from one side, and sends RTP to the other side while maintaining the same ssrc, seq and timestamp as received.
  Valid values: "end_system”, ”translator"
  Default value:"end_system"
- **header** – *object,* defines configuration of rtp initial header fields:
  - **ssrc** – *integer,* RTP SSRC value used for outgoing RTP stream. If not specified, will be generated randomly
  - **seq** – *integer,* initial RTP sequence number for outgoing stream. If not specified, will be generated randomly
  - **timestamp** – *integer,* initial RTP time stamp for 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 used in 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
    - **TOS** – *integer,* Type Of Service byte, that will be used in outgoing RTCP packets, as defined in IPv4 specification.
      Default value: 0
    - **coupled_tool_id** – *integer,* ID of tool represents other direction of RTP session to obtain required statistics for RTCP reports. If not set or set to -1, reading statistics from another tool will not be performed
      Valid values: (-1) – 0xFFFF
      Default value: -1
    - **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.
      Valid values: 0 – 300
      Default value: 70
    - **adaptation_type** – *string,* type of the adaptation algorithm:
      Valid value: "none", "full_adaptive", "short_adaptive", "non_managed_network"
      Default value: "short_adaptive"
    - **depth** – *integer,* jitter buffer depth in milliseconds.
      Valid values: 0 – 300
      Default value: 200

- **min_delay** – *integer,* minimal delay in milliseconds, used by adaptation algorithm.
  Valid values: 0 – 300
  Default value: 40
- **max_delay** – *integer,* maximal delay in milliseconds, used by adaptation algorithm.
  Valid 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.
  Valid 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 samples coming from this tool's input and are sent to the RTP side. decoder_side applies to samples received from IP (using RTP), and 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.
    Valid value: 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.
    Valid values: -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.
    Valid values: -37 -6
    Default value: -10
  - **step_level** – *integer,* the gain step level in 0.1 dB,sec units.
    Valid values: 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.
    Valid values: -60 -30
    Default value: -30
  - **limit_gain** – *boolean,* determines if max_gain value is applied and used.
    Default value: false

- **max_gain** – *integer,* the maximum value the applied gain may take in 1db units.
  Valid values: -31 - 18
  Default value: 18

- **EVG** – *object,* defines Event Generator configuration and Out-Of-Band tonal event (RFC) detection. 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 preferred event that should be generated at the same time.
    Valid values: "delay", "delete"
    Default value: "delay"
  - **convert_RTP_events_to_inband** – *string,* automatically generate In-Band tone if corresponding tonal event was received from RTP (RFC).
    Valid values: "none", "CPS", "modem", "all"
    Default value: "none"
  - **convert_inband_events_to_RTP** – *boolean,* automatically generate RTP tonal events (RFC) when a corresponding tone was detected In-Band by the EVD object.In "voice_fe_ip" it corresponds only to the EVD of "encoder_side".
    Default value: false
  - **relay_RTP_events** – *boolean,* relays input RTP events to RTP output.
    In "voice_fe_ip", it corresponds only to the EVD of "decoder_side".
    Default value: false

- **EVD** – object, defines In-Band Event Detector configuration to detect tonal events that are carried within the voice payload.

  - "voice_p2p" tool includes one set of EVD parameters (as listed below).
  - "voice_fe_ip" tool includes 2 sets of EVD parameters, each is a sub-object:"encoder_side" and "decoder_side".
  - "encoder_side" applies to the samples that are coming from its backend_tool_id 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 as its backend_tool_id

  The EVD set of parameters includes the following fields:

  - **enabled** – *boolean,* enables/disables EVD.
    Default value: false
  - **events** – *string,* defines event types detected by EVD.
    Valid 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
    Default value: empty

- **tone_suppression** – *string,* defines type of tone suppression applied to voice samples.
  Valid 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
- **user_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.
    Valid values: "USER_DEF1", "USER_DEF8"
  - **description** – *string,* event description.
    Valid values: "undefined", "dial_tone", "ringback_tone", "busy_tone", congestion_tone", "information_tone"
  - **duration_tolerance** – integer
  - **min_power** – integer
  - **type** – *string,*
    Valid 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.

  Note! - In order to detect RTP events (OOB) tones, the EVG object should be set to "enable".

- **SRTP** – *object,* contains SRTP configuration. Consists of the following fields:

- **rtp_in_ctx_id** – *integer,* ID of SRTP context defined in the "contexts" array that will be used for incoming RTP packets decryption,authentication. If set to -1 no SRTP will be used for this direction.
  Valid values: (-1) – 3
  Default value: -1
- **rtp_out_ctx_id** – *integer,* ID of SRTP context defined in the "contexts" array that will be used for outgoing RTP packets encryption. If set to -1, no SRTP will be used for this direction.
  Valid values: (-1) – 3
  Default value: -1
- **rtcp_in_ctx_id** – *integer,* ID of SRTP context defined in the "contexts" array that will be used for incoming RTCP packets decryption,authentication. If set to -1 no SRTP will be used for this direction.
  Valid values: (-1) – 3
  Default value: -1
- **rtcp_out_ctx_id**- *integer,* ID of SRTP context defined in the "contexts" array that will be used for outgoing RTCP packets encryption. If set to -1 no SRTP will be used for this direction.
  Valid values: (-1) – 3
  Default value: -1
- **contexts** – array of objects, contains SRTP contexts configuration. Each object contains the following parameters:
  - **id** – *integer,* identifier of the context. Used in rtp,rtcp_in,out_ctx_id parameters to specify which context will be used for specific direction.
    Valid values: 0 – 3
  - **encryption** – *string,* specifies encryption type that is used in this context.
    Valid values: "none", "aes_icm"
    Default value: "aes_icm"
  - **authentication** – *string,* specifies authentication type that will be used in this context.
    Valid values: "none", "hmac_sha1"
    Default value: "hmac_sha1"
  - **auth_tag_len** – *integer,* specifies the length of the SRTP authentication tag in this context.
    Valid values: 0 – 20
    Default value: 10
  - **master_key** – *string,* specifies the master key for this SRTP context. The string must be in hex format and contain the whole key even when there are several zeros at the beginning of the key.
    (i.e., "000102030405060708090a0b0c0d0e0f").
    Valid length of the master key: 0 – 32 octects (0 – 256 bits)
  - **salt_key** – string, specifies salt key for this SRTP context. String must be in hex format and contain the whole key even when there are several zeros at the beginning of the key.
    (i.e., "000102030405060708090a0b0c0d").
    Valid length of the salt key: 14 octects (112 bits)

5.1.2 Commands

5.1.2.1 Stop tone generation command

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 events to generate in millisecond. If value exceeds sum of all event durations, generator will start generating event from the beginning of the sequence until total required duration is reached. Can be also set to string "infinite".
  Valid values: 0-2^32
  Default value: N/A
- **rtp_or_inband** – *string,* defines whether this event will be generated in-band or out-of-band.
  Valid values: "RTP", "inband"
  Default value: N/A
- **direction** – *string,* relevant only if rtp_or_inband is set to "inband" and for voice_fe_ip. Defines if generated in-band is sent to backend or directly to IP.
  Valid values: "backend", "IP"
  Default value: "backend"
- **tone_type** – *string,* relevant only if rtp_or_inband is set to "inband".
  Valid values: "predefined", "user_defined"
- **ip_event** – *object,* relevant only if rtp_or_inband is set to "RTP". Contains the following field:
  - **named_event** – *string,*
    Valid 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.
  Valid 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 item in milliseconds. Can be also set to string "infinite".
  - **cadence_or_am** – *string,* defined how item will be described.
    Valid 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".
    Valid 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 Event detector module of the voice_p2p tool detects an event.

Parameters:

- **“type”:”inband_event_detected”**
- **"event"**- *string,* event type that was detected by the EVD.
  Valid 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, USER_DEF1..USER_DEF8

5.1.3.2 "inband_event_ended" event

This event is sent when the Event detector module of the voice_p2p tool detects and of an event.

Parameters:

- **“type”:”inband_event_ended”**

5.1.3.3 "inband_event_detected_from_ip" event

This event is sent when the Event detector module of the decoder_side of voice_fe_ip tool detects an event.

Parameters:

- **“type”:”inband_event_detected_from_ip”**
- **"event"**- *string,* event type that was detected by the EVD.
  Valid 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, USER_DEF1..USER_DEF8

5.1.3.4 "inband_event_ended_from_ip" event

This event is sent when the Event detector module of the decoder side of voice_fe_ip tool detects end of an event.

Parameters:

- **“type”:”inband_event_ended_from_ip”**

5.1.3.5 "inband_event_detected_from_fe" event

This event is sent when the Event detector module of the encoder_side of voice_fe_ip tool detects an event.

Parameters:

- **“type”:”inband_event_detected_from_fe”**
- **"event"**- *string,* event type that was detected by the EVD.
  Valid 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, USER_DEF1..USER_DEF8

5.1.3.6 "inband_event_ended_from_fe" event

This event is sent when the Event detector module of the encoder side of voice_fe_ip tool detects end of an event.

Parameters:

- **“type”:”inband_event_ended_from_fe”**

5.1.3.7 "RTP_event_detected" event

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

Parameters:

- **“type”:”RTP_event_detected”**
- **"event"**- *string,* event type that was detected by the EVD.
  Valid 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

5.1.3.8 "RTP_event_ended" event

This event is sent when RTP tone, that has being received from the network as OOB, ends

Parameters:

- **“type”:”RTP_event_ended”**

5.1.3.9 "RTCP_received" event

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

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

5.1.3.10 "RTCP_sent" event

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

Parameters:

- **“type”:”RTCP_sent"**
- The same as in RTCP_received event

5.1.3.11 "AMR_Codec_Mode_Request" event

This event is triggered in the CMR field if incoming AMR RTP packet is different from its older value.

Parameters:

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

5.1.4 Statuses

5.1.4.1 RTP

Includes RTP related statistics.

Parameters:

- **“type” : ”RTP”**
- **PT_received** – *integer,* RTP payload type received from the far end
- **SSRC_received** – RTP ssrc received from the remote side
- **received_packets** – *integer,* number of received packets
- **received_bytes** – *integer,* total number of received bytes
- **SSRC_sent** – *integer,* SSRC 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** – *integer,* total number of frames that were decoded
- **AMR_rate_received** – *string*, when decoder is configured to AMR, this parameter will contain the AMR rate that was actually received by the decoder. See AMR rates table in decoder/encoder configuration for more details

5.1.4.4 encoder

Encoder related statistics.

Parameters:

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

5.1.4.5 EVG

Event generator related statistics.

Parameters:

- **"type" : "EVG"**
- **is_active** – *boolean,* set to true if EVG is currently generating tone

5.2 voice_mixer tool

voice_mixer tool can create a voice conference of several participants. Each participant 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.
  Valid 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".
  Valid values: 0-5
  Default value: 3
- **hangover_period** – *integer,* period in milliseconds representing a participant becoming a "dominant speaker". It will not be replaced by another participant even if does not produce any speech until this timeout is expired.
  Valid 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.
  - **type** – *string,* participant type.
    Valid 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 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 Statuses

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 that include audio/video streams, one after the other.

5.3.1 Tool configuration

Parameters:

- **audio_enabled** – *boolean,* enables/disables audio stream.
  Default value: true
- **audio_dst_tool_ids** – *dst_tool_ids* for audio stream. as described in section 3.1.1
- **video_enabled** – *boolean,* enables/disables audio stream.
  Default value: false
- **video_dst_tool_ids** – *dst_tool_ids* for video stream. 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"
          "audio_enabled":true,
           "audio_dst_tool_ids":[2]",
           "video_enabled":true,
           "video_dst_tool_ids":[3]}
}}
```

5.3.2 Commands

5.3.2.1 Playlist append

Append files to the playlist

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. The full path including the file name is required.
  - **duration** – *float,* number of seconds to play the file.

If the value is less than the actual file duration, the file is played *duration* seconds.

If the value is more than the actual file duration, the file is played several times in order to achieve this duration. Several full times, and the last time can be the whole file or part of it.

If set to -1, the entire file will be played.

Default value: -1

- **format** – *string,* special file format, relevant only for files without extension
  Valid values: "none",
  "flat_a" – a file without any header, includes only G.711alaw data
  "flat_u" – a file without any header, includes only G.71ulaw data
- **segment** – *int*, the VOX file format segment to play.
  Relevant only for VOX file format.
  Valid values: 0 to ACTIVIDX (VOX number of indices in messages)
  Default value: 0

- **"repetitions" -** *int,* number of times to play the set of files

  Default value: 1

- **"duration" -** *float,* number of seconds to play the set of files

  Determines the amount of time the set of files shall be played. The total set of files is played several times in order to achieve this duration. The last file played can be whole or part.

  If set to -1, the set of files will be played once.

  Default value: -1

  Note: it is allowed to specify set of files duration or repetition but not both.
  If both are specify, the repetitions parameter is ignored

  File formats and codecs supported:

File Format (ext)	Codecs Supported		Description
	Audio	**Video**	
.wav	G.711alaw,G.711ulaw,linear		A Standard Waveform Audio File Format
.amr	NB_AMR OA,WB_AMR OA		A standard Adaptive Multi-Rate ACELP codec file
.vox	G.711alaw,G.711ulaw		Dialogic VOX file format
.WebM	Opus		A standard Matroska WebM video files for web

.opus	Opus		A standard Lossy audio coding format
.3gp, .mp4	NB_AMR OA,WB_AMR OA	H264	standard Multimedia container formats
.avi	G.711alaw,G.711ulaw, NB_AMR OA,WB_AMR OA	H264	A standard multimedia container format

Example:

```
{"tool_req":{
    "req_type":"command",
    "req_id":222,
    "tool_id":1,
    "data":{
       "cmd_type":"play_list_append",
        "repetitions": 3,
       "files":[{"name":"one.3gp"},
               {"name":"two.3gp", "duration":10.5},
               {"name":"three.3gp"}]
    }
}}
```

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 "Play started" event

This event is sent when the first frame of a file is actually inserted into voice tool and,or video tool for playing.

Parameters:

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

Example:

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

5.3.3.2 "File info" event

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

This information may be used for the configuration of the audio/video tools.

Parameters:

- **“type”:”file_info”**
- **"file_name"** – *string,* the name of the file
- **"file_info" -** *array of objects,* each object contains info about a single stream in the file
    - **"stream_id"** – *integer,* the stream index in the file

- **"codec_type"** - *string,* decoder coding standard.
  Valid values:
  "linear", "G.711alaw",G.711ulaw", "G.729", "AMR_NB", "AMR_WB",
  "OPUS", "H.264". (If the codec is not supported: "not supported".)

Example:

```
{"tool_inf":{
    "tool_id":1,
    "inf_type":"event",
    "data":{
       "type":"file_info",
       "file_name":"one.wav",
       "file_info":[{"stream_id":0,
                     "codec_type":"G.711alaw"},
                    {"stream_id":1,
                     "codec_type":"H.264"}]
    }
}}
```

5.3.3.3 "Error reading from file" event

This event is sent when there is an error opening,reading from file.

Parameters:

- **"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":{
       "type":"error_reading_from_file",
       "file_name":"one.mp4"}
}}
```

5.3.3.4 "End of file" event

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

Parameters:

- "**type":"end_of_file"**
- **"file_name"** – *string,* the name of the finished file
- **"next_file_name"** – *string,* the name of the next file to be played

Example:

```
{"tool_inf":{
    "tool_id":1,
    "inf_type":"event",
```

```
        "data":{
           "type":"end_of_file",
           "file_name":"one.mp4",
           "next_file_name":"two.mp4"
        }
    }}
```

5.3.3.5 "End of playlist" event

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

Parameters:

- "**type":"end_of_playlist"**

Example:

```
    {"tool_inf":{
        "tool_id":1,
        "inf_type":"event",
        "data":{
           "type":"end_of_playlist"
        }
    }}
```

5.3.4 Statuses

5.3.4.1 general

File reader related statistics.

Parameters:

- **"type" : "general"**
- **sent_packets_to_audio** – total number of packets sent to audio
- **sent_packets_to_video** – total number of frames sent to video
- **errors** - *integer,* number of errors in file reader

5.4 File_writer tool

File writer implements record audio/video streams to supported file formats.

Tested on: VLC media player and Windows media player.

5.4.1 Tool configuration

Parameters:

None

Example:

```
    {"tool_req":{
```

```
        "req_type":"set_config",
        "req_id":111,
        "tool_id":1,
        "data":{
             "tool_type":"file_writer"
}}
```

5.4.2 Commands

5.4.2.1 record

Start record streams to file.

After “pause” command - continue to record from the same location.

Parameters:

- **“cmd_type”:”record”**
- **"file_name" -** *string,* the file name including extension. The full path including the file name is required.

  File formats and codecs supported:

File Format (ext)	**Codecs Supported**	
	Audio	**Video**
.wav	G.711alaw,G.711ulaw,linear	
.amr	NB_AMR,WB_AMR OA	
.WebM	opus	
.opus	opus	
.3gp, .mp4	NB_AMR OA,WB_AMR OA	H.264
.avi	G.711alaw,G.711ulaw, NB_AMR OA,WB_AMR OA	H.264

- **“max_size”** – *integer,* maximum file size in Kbytes.
  Default value: 100000
  (If size is reached before "stop" command, handle as if “stop” command was requested.)
- **audio_enabled** – *boolean,* enables/disables audio stream.
  Default value: true
- **video_enabled** – *boolean,* enables/disables audio stream.
  Default value: false
- **stream_timeout** – *integer,* the time out waiting for first packet of the streams, in ms.
  Default value: 5000
  If both streams are enabled, but one of the streams arrived before timeout, this stream

is recorded alone.

Example:

```
{"tool_req":{
    "req_type":"command",
    "req_id":222,
    "tool_id":1,
    "data":{
       "cmd_type":"record",
       "file_name":"test1_out.wav"
       "max_size": 500000
       "audio_enabled":true,
       "video_enabled":true,
        "stream_timeout": 3000
    }
}}
```

5.4.2.2 pause

Pauses record to file, and the file remains open. Start playing again by "*record*" command.

Example:

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

5.4.2.3 stop

Stop record to file. Close the file.

Example:

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

5.4.3 Events

5.4.3.1 "Record started" event

This event is sent when the first frame of each audio/video stream is written to the file.

Parameters:

- **“type”:”record_started”**
- **"file_name"** – *string,* the name of the file
- **"stream_id"** – *integer,* the stream index in file

Example:

```
{"tool_inf":{
    "tool_id":1,
    "inf_type":"event",
    "data":{
       "type":"record_started",
       "file_name":"one.wav"
        "stream_id":0
    }
 }}
```

5.4.3.2 "File info" event

This event is sent when the file is opened.

This information may be used for the configuration of the audio/video tools.

Parameters:

- **“type”:”file_info”**
- **"file_name"** – *string,* the name of the file
- **"file_info" -** *array of objects,* each object contains information about a single stream in the file
  - **"stream_id"** – *integer,* the stream index in the file
  - **"codec_type"** - *string,* encoder coding standard
    Valid values: "linear”, ”G.711alaw”, ”G.711ulaw”, ”AMR_NB”, ”AMR_WB”, ”H.264" ,"OPUS”, ”G.729”, ”G.722"
    Some of the codecs are supported only in part of the file formats, see 5.4.2.1 table for more details.

Example:

```
{"tool_inf":{
    "tool_id":1,
    "inf_type":"event",
    "data":{
       "type":"file_info",
       "file_name":"one.wav",
```

```
            "file_info":[{"stream_id":0,
                          "codec_type":"G.711alaw"},
                         {"stream_id":1,
                          "codec_type":"H.264"}]
        }
    }}
```

5.4.3.3 "Record error" event

This event is sent when there is an error opening,recording to file.

Parameters:

- **"type":"record_error"**
- **"file_name"** – *string,* the name of the file
- **"error_code"** - *integer,* a number that represent the error
- **"description"** - *string,* a brief description of the error

Error codes and corresponding description

Error code	Description string	description
1	wrong_path	The path doesn't exist
2	file_exist	The file already exists
3	I,O_open_error	I,O open error
4	format_not_supported	The file format (extension) is not supported
5	bad_streams	None of the streams are suitable for record. codecs are not supported, or,and codecs don't match format
6	open_file_error	All other possible open file errors allocation, etc.
7	writing_file_error	Error writing to file – allocation, etc.

Example:

```
{"tool_inf":{
    "tool_id":1,
    "inf_type":"event",
    "data":{
       "type":"recorde_rror",
       "file_name":"one.avi",
        "error_code":"2",
```

```
            "error_description":"file_exist"
    }}}
```

5.4.3.4 "Record Stopped" event

This event is sent when record to file is ended, and the file is closed

Parameters:

- **“type”:”record_stopped”**
- **"file_name"** – *string,* the name of the file

Example:

```
{"tool_inf":{
    "tool_id":1,
    "inf_type":"event",
    "data":{
       "type":"record_stopped",
       "file_name":"two.wav"
    }
}}
```

5.4.3.5 "Stream_timeout" event

This event is sent when an audio/video stream hasn't arrived before timeout occurred

Parameters:

- **“type”:”stream_timeout”**
- **"media_type"** – *string,* the name of the media type.
  Valid values: “audio”, ”video”

Example:

```
{"tool_inf":{
    "tool_id":1,
    "inf_type":"event",
    "data":{
       "type":"stream_timeout",
       "file_name":"two.wav",
        "media_type":"video"
    }
}}
```

5.4.4 Statuses

5.4.4.1 general

File writer related statistics.

Parameters:

- **“type” : ”general”**
- **audio_packets_received** – total number of packets received from audio
- **video_packets_received** – total number of frames received from video
- **errors** - *integer,* number of errors in file writer

5.5 audio_decoder tool

Receives audio coded frames. 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.

5.5.1 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** – *string,* 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.
  Valid values: "mono”, ”stereo"
  Default value: “mono”

5.5.2 Statuses

5.5.2.1 general

General statistics.

Parameters:

- **“status_type” : ”general”**
- **errors** - *integer,* total number of errors in decoding.
- **warnings** – *integer,* total number of warnings in decoding.
- **decoded_frames** – *integer,* number of decoded frames.
- **input_sampling_rate** – *integer,* sampling rate of the input stream.
- **input_mono_stereo**– *string,* channel configuration of the input stream.
- **input_bitrate** – *integer,* output bitrate measured in interval between 2 status reports.

5.6 audio_encoder tool

Receives audio frame in linear16 format (for example from audio_decoder), encodes it, generates coded frame and passes it to the destination tools that were configured in set_config message.

5.6.1 Tool configuration

Parameters:

- **dst_tool_ids** – as described in section 3.1.1
- **codec** – *string,* codec type that will be used for encoding.
  Valid values: "opus", "linear"
  Default value: "opus"
- **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, 16000, 24000, 48000
  Default value: 0
- **out_mono_stereo** - *string,* 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.
  Valid values: "mono"
  Default value: "mono"
- **opus** – *object,* contains opus codec specific configuration parameters, relevant only when codec field is set to "opus", contains the following fields:
  - **mode** – *string,* specifies the opus encoder mode.
    Valid values: "audio", "voice", "restricted_low_delay"
    Default value: "audio"
  - **complexity** – *integer,* opus codec complexity.
    Valid values: 0 – 10
    Default value: 10
  - **bitrate** – *integer,* bit-rate in bits per second of the output stream.
    Valid values: 6000 – 512000
    Default value: 128000
  - **bitrate_control** – *string,* Bit Rate Control algorithm.
    Valid values: "CBR", "VBR" (Constant Bit-Rate , Variable Bit-Rate)
    Default value: "CBR"

5.6.2 Statuses

5.6.2.1 general

General statistics.

Parameters:

- **“status_type” : ”general”**
- **errors** - *integer,* total number of errors in encoding
- **warnings** – *integer,* total number of warnings in encoding
- **encoded_frames** – *integer,* number of encoded frames
- **input_sampling_rate** – *integer,* sampling rate of the input stream
- **input_mono_stereo**– *string,* channel configuration of the input stream
- **output_bitrate** – *integer,* output bitrate measured in interval between 2 status reports

5.7 rtp_session tool

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

In one direction it receives UDP packets, (i.e., from udp_socket tool), handles RTP header, inserts the payload into JB and fetches it towards output (i.e., rfc6184_depacketizer tool).

In the other direction it receives RTP payload, adds RTP header and sends it (i.e., to udp_socket tool). It also handles RTCP; receives RTCP packet and notifies the controller.

In the other direction it generates RTCP packets according to its configuration and passes the RTCP packet to another tool, (i.e., 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_types** – *array of objects,* specifies RTP payload type to CODEC type mapping. Each object contains the following values:
    - **payload_type** – *integer,* RTP payload type.
      Valid values: 0 – 127
    - **codec** – *string,* specifies codec type of the given RTP payload type.
      Valid values: "opus", "linear", "H.264"
    - **RTP_TS_freq** – *integer,* RTP Time Stamp Frequency., Specifies RTP clock frequency for the given payload type. If set to 0, the default value for the given codec will be used: opus – 48000, linear – 8000, H.264 – 90000.
      Default value: 0
- **rtp_out** – object, contains configuration of the transmit side of RTP
  - **enabled** – *boolean,* enables/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** – *object,* contains RTCP configuration for receive direction. Consists of the following fields:
  - **enabled** – *boolean,* enable/disable receiving of RTCP packets.
    Default value: false
  - **vid_enc_fb_tool_id** – *integer,* tool id of the video encoder that uses this rtp_session tool to packetize its output data. This feedback is used to automatically pass RTCP requests like FIR and PLI to the encoder tool. If set to -1, the feedback is not passed to the video_encoder tool.
    Valid values: (-1) – 0xFFFF

- **rtcp_out** – *object,* contains RTCP configuration for transmit direction. Consists of the following fields:
  - **enabled** – *boolean,* enable/disable sending of RTCP packets.
    Default value: false
  - **dst_tool_ids** – destination tools for created RTCP packets in format as described in section 3.1.1
  - **report_interval** – *integer,* period in milliseconds between RTCP reports in seconds. According to RFC 3550, time between 2 RTCP outgoing packets will be random in the range of 0.5X – 1.5X this value.
    Default value: 5000
  - **cname** – *string,* RTCP cname. If not set or set to empty string, CNAME will not be sent in outgoing RTCP packets.
    Default value: empty string2
  - **reduced_size** – *boolean,* enable/disable reduced size RTCP packets support (RFC 5506).
    Default value: true

- **srtp** – *object,* contains SRTP configuration. Consists of the following fields:
  - **rtp_in_ctx_id** – *integer,* id of the SRTP context which is defined in the "contexts" array that will be used for incoming RTP packets decryption,authentication. If set to -1 no SRTP will be used for this direction.
    Valid values: (-1) – 3
    Default value: -1
  - **rtp_out_ctx_id** – *integer,* id of the SRTP context which is defined in the "contexts" array that will be used for outgoing RTP packets encryption. If set to -1 no SRTP will be used for this direction.
    Valid values: (-1) – 3
    Default value: -1

- **rtcp_in_ctx_id** – *integer,* id of the SRTP context which is defined in the "contexts" array that will be used for incoming RTCP packets decryption, authentication. If set to -1 no SRTP will be used for this direction.
  Valid values: (-1) – 3
  Default value: -1
- **rtcp_out_ctx_id**- *integer,* id of the SRTP context which is defined in the "contexts" array that will be used for outgoing RTCP packets encryption. If set to -1 no SRTP will be used for this direction.
  Valid values: (-1) – 3
  Default value: -1
- **contexts** – *array of objects,* contains SRTP contexts configuration. Each object contains the following parameters:
  - **id** – *integer,* identifier of the context. Used in rtp,rtcp_in,out_ctx_id parameters to specify which context will be used for specific direction.
    Valid values: 0 – 3
  - **encryption** – *string,* specifies encryption type that is used in this context.
    Valid values: "none", "aes_icm"
    Default value: "aes_icm"
  - **authentication** – *string,* specifies authentication type that will be used in this context.
    Valid values: "none", "hmac_sha1"
    Default value: "hmac_sha1"
  - **auth_tag_len** – *integer,* specifies the length of the SRTP authentication tag in this context.
    Valid values: 0 – 20
    Default value: 10
  - **master_key** – *string,* specifies the master key for this SRTP context. The string must be in hex format and contain the whole key even when there are several zeros at the beginning of the key.
    (i.e., "000102030405060708090a0b0c0d0e0f").
    Valid length of the master key: 0 – 32 octects (0 – 256 bits)
  - **salt_key** – *string,* specifies the salt key for this SRTP context. The string must be in hex format and contain the whole key even when there are several zeros at the beginning of the key.
    (i.e., "000102030405060708090a0b0c0d").
    Valid length of the salt key: 14 octects (112 bits).

5.7.2 Commands

5.7.2.1 send_FIR

Sends RTCP AVPF FIR message to the far end. See RFC 5104 for message details.

- additional parameters: none

5.7.2.2 send_PLI

Sends RTCP AVPF PLI message to the far end. See RFC 4585 for message details.

- additional parameters: none

5.7.3 Events

5.7.3.1 "received_RTCP_SR" event

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

Parameters (seeRFC 3550 for detailed description of parameters):

- **“type”:”received_RTCP_SR"**
- **ssrc** – *integer,* SSRC of the RTCP packet sender
- **NTP_high** – *integer,* NTP timestamp (higher 32 bits)
- **NTP_low** – *integer,* NTP timestamp (lower 32 bits)
- **RTP_ts** – *integer,* current RTP timestamp
- **packet_count** - *integer,*
- **octet_count** – *integer,*

5.7.3.2 "received_RTCP_RR" event

This event is sent when Receiver Report Block is received from the far end (it can be received inside both RR and SR packets).

Parameters (seeRFC 3550 for detailed description of parameters):

- **“type”:”received_RTCP_RR"**
- **ssrc** – *integer,* SSRC of the RTCP packet sender
- **source_ssrc** – *integer,* SSRC of sender of stream which statistics are being reported
- **frac_lost** – *integer,* fraction lost as defined in RFC 3550
- **cumulative_packet_lost** – *integer,* as defined in RFC 3550
- **ext_hi_seq_recv** – *integer,* as defined in RFC 3550
- **interarrival_jitter** - *integer,* as defined in RFC 3550
- **round_trip_delay -** *integer,* locally calculated round trip delay in milliseconds. The calculation is based on data received in Received Report.

5.7.3.3 "received_RTCP_SDES" event

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

Parameters (seeRFC 3550 for detailed description of parameters):

- **“type”:”received_RTCP_SDES"**
- **cname** – *string,* canonical name as defined in RFC 3550

5.7.3.4 "received_RTCP_BYE" event

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

Parameters (seeRFC 3550 for detailed description of parameters):

- **“type”:”received_RTCP_BYE"**

5.7.3.5 "received_RTCP_AVPF_FIR" event

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

Parameters (seeRFC 5104 for detailed description of parameters):

- **“type”:”received_RTCP_AVPF_FIR"**

5.7.3.6 "received_RTCP_AVPF_PLI" event

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

Parameters (seeRFC 4585 for detailed description of parameters):

“type”:”received_RTCP_AVPF_PLI"

5.7.4 Statuses

5.7.4.1 RTP

Includes RTP related statistics.

Parameters:

- **“status_type” : “RTP_stat”**
- **sent_packets** – *integer,* number of sent packets
- **sent_bytes** – *integer,* total number of bytes sent
- **outgoing_bitrate** – *integer,* outgoing bit rate in bits per second
- **sent_ssrc** – *integer,* SSRC used for the outgoing stream
- **received_ssrc** – *integer,* RTP ssrc that was received from the remote side
- **received_payload_type** – *integer,* received RTP payload type
- **received_codec** – *string,* received codec, obtained from RTP payload type using RTP_in payload_types array for mapping.
- **received_packets** – *integer,* number of received packets
- **received_bytes** – *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, (i.e., 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 used to send outgoing packets
- **override_udp_src_port** – *integer,* outgoing packets source port will be overridden with this value. If not set or set to 0, the original source port will be preserved.
  Default value: 0
  (**Note:** correct functionality of this feature requires setting valid "local_ip" parameter, and root privileges for Media Processor on the HMP machine.)
- **local_ip** – *string,* IP address of the interface that will be used to send,receive RTP packets. If not set or set to "0.0.0.0", the interface will be chosen automatically.
  Default value: "0.0.0.0"
- **remote_ip** – *string,* specifies remote IP where outgoing packets will be sent
- **rcv_buff_size** – *integer,* socket buffer size for RX side.
  Default value: 0x8000
- **send_buff_size** – *integer,* socket buffer size for TX side.
  Default value: 0x8000
- **rcv_multicast_addr** – *string,* if specified the socket will subscribe to receive input from this multicast address
- **TOS** – *integer,* Type Of Service byte as defined in IPv4 specification.
  Default value: 0

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 is changed. This event is also sent when receiving UDP packet for the first time.

Parameters:

- **“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 Statuses

5.8.3.1 general

General statistics.

Parameters:

- **“status_type” : ”general”**
- **sent_packets** – *integer,* total number of sent packets
- **received_packets** – *integer,* total number of received packets
- **sent_bytes** – *integer,* total number of bytes sent
- **received_bytes** – *integer,* total number of bytes received
- **outgoing_bitrate** – average outgoing bitrate calculated for the period between 2 status reports
- **incoming_bitrate** - average incoming bitrate calculated for the period between 2 status reports

5.9 video_decoder tool

Receives full encoded frame, decodes it and sends it to output.

If the incoming resolution does not match the configured width and height, it resizes it to the configured resolution. This tool can receive its input from any other tool generating video coded frames as rtp_session,video_encoder,file_reader.

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.
  Valid values: 0 - 3840
  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.
  Valid values: 0 - 2160
  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:

- **“type”:”new_stream"**
- **codec** – *string,* CODEC type of the input stream ("H.264")
- **height** – *integer,* incoming frames height
- **width** – *integer,* incoming frames width

5.9.3 Statuses

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
- **bitrate** – *integer,* bitrate of the incoming stream, calculated as average bitrate in period between 2 status reports
- **waiting frames** – *integer,* current size of the input frames queue, if this value is constantly higher than 0 it can mean that the system is experiencing performance issues
- **codec** – *string,* codec type of the incoming stream ("H.264")

5.10 video_encoder tool

Receives YUV frame, encodes it and sends to output. Input YUV frames can be obtained from video_decoder tool or video_mixer tool.

5.10.1 Tool configuration

Parameters:

- **dst_tool_ids** – as described in section 3.1.1
- **codec** – *string,* CODEC type that is used for the encoding.
  Valid values: "H.264"

- **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.
  Valid values: 0 – 3840
  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.
  Valid values: 0 - 2160
  Default value: 0
- **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 (currently raising framerate is not supported). 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.
  Default value: 0
- **bitrate** – *integer,* output stream bit rate in Kbits per second units.
  Default value:768
- **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
- **idr_interval** – *integer,* maximal interval between 2 IDR frames in the stream. If set to 0, no limitation is applied.
  Default value: 0
- **preset** – *string,* internal settings preset for the encoder.
  Valid values: "speed", "balanced", "quality"
  Default value: "balanced"
- **min_fir_interval** – *integer,* minimal period in milliseconds between handling of FIR,PLI feedbacks that are received from rtp_session tool. When video_encoder tool receives FIR,PLI request from rtp_session tool, it automatically generates IDR frame. If this parameter is set to -1, those messages are ignored. If FIR,PLI request was received before the minimal interval passed, generation of the new IDR frame is delayed till the minimal period. If this parameter is set to 0, any FIR,PLI feedback is handled immediately.
  Valid values: (-1) – 0x7fffffff
  Default value: (-1)
- **H.264** – *object,* contains specific parameters for H.264 encoder, is relevant only when codec is set to "H.264". Contains the following parameters:

- **profile** – *string,* h.264 profile.
  Valid values: "baseline", "main", "high"
  Default value: "baseline"
- **level** – *string,* h.264 level.
  Valid values: "3.0" - "5.2"
  Default value: "3.0"

5.10.2 Commands

5.10.2.1 generate_idr_frame

- additional parameters: none

5.10.3 Events

5.10.4 Statuses

5.10.4.1 general

General statistics.

Parameters:

- **"status_type" : "general"**
- **errors** - *integer,* total number of errors in encoding
- **warnings** – *integer,* total number of warnings 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
- **waiting_frames** – *integer,* current size of the input frames queue, if this value is constantly higher than 0 it can mean that the system is experiencing performance issues
- **input_framerate** – *integer,* incoming frame rate in frames per second units measured in interval between 2 status reports
- **output_bitrate** – *integer,* average encoding bitrate measured in interval between 2 status reports

5.11 video_mixer tool

Video mixer is capable of receiving YUV frames from multiple sources (tools) and creating a single composition output that is also in form of YUV frame.

5.11.1 Tool configuration

Parameters:

- **dst_tool_ids** – as described in section 3.1.1
- **width** – *integer,* output YUV frame width
  Valid values: 16 – 3840
  Default value: 1920
- **height** – *integer,* output YUV frame height
  Valid values: 16 – 2160
  Default value: 1080
- **framerate** – *integer,* frame rate of the output composition stream in frames per second units.
  Default value: 30
- **R, G, B** – *integer,* specifies the background color of the output image, "red", "green" and "blue" components of the color.
  Valid values: 0 – 255
- **participants** – *array of objects,* describes video_mixer tool inputs. Used for both adding,updating,removing inputs to the tool. Each object in this array contains the following fields:
  - **par_id** – *integer,* participant unique identifier.
    Valid values: 0 – 0xFFFFFFFF
  - **action** – *string,* describes the configuration action that should be performed.
    Valid values: "add", "remove". "add" is capable of both adding and updating the participant
    Default value: "add"
  - **tool_id** – *integer,* tool id of the source tool which originates the input YUV stream.
    Valid values: 0 – 0xFFFF
- **layout** – *array of objects,* describes the output composition configuration. Each object represents a "draw" command. Each object contains the following fields:
  - **par_id** – *integer,* unique identified of the participant that is supposed to be presented, this is the ID that was configured in "participants" array.
    Valid values: 0 – 0xFFFFFFFF
  - **src_x, src_y** – *integers,* coordinates of the left top corner of the rectangle on participant frame that should be presented. Must be in range of the source frame's size.
  - **src_w, src_h** – *integer,* width and height of the rectangle on participant frame that will be presented. The whole rectangle must be in range of the source frame's size.
  - **dst_x, dst_y** – *integer,* coordinates of the left top corner of the rectangle on the output frame that will contain the newly presented image. The rectangle must be in range of the output frame's size.

- **dst_w, dst_h** – *integer,* width and height of the rectangle on the output frame that will contain the newly presented image. The rectangle must be in range of the output frame's size.
- **preserve_aspect_ratio** – *boolean,* if set to true uses cropping on the source image rectangle in order to present it with the same aspect ratio as on the source frame.
  Default value: false

5.11.2 Statuses

5.11.2.1 general

General statistics.

Parameters:

- **"status_type" : "general"**
  - **generated_frames** - *integer,* total number of generated frames
  - **missed_frames** – *integer,* total number of frames that were not generated (missed) due to performance peaks of other issues

6. IWF tools API:

Surf Inter-Working Function (IWF) tools

IWF for GSM and UMTS Overview

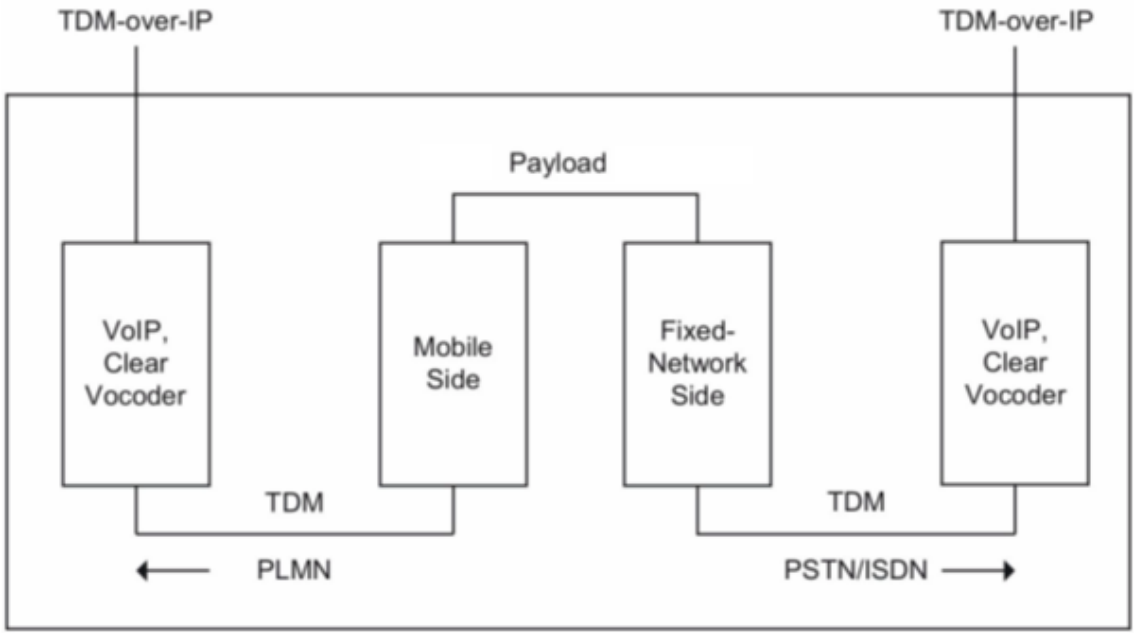
A Mobile circuit-switched IWF is a functional unit of the Mobile Core Network MSC (Mobile Switching Center), which is a part of the Mobile network's CS (Circuit Switched) domain.

The *Surf* IWF circuit is a combination of two tools:

- **Mobile** tool: This tool is connected to the mobile radio-link network via the A-Interface and supporting mobile rate adaptation schemes, for Bearer (Data) or Transparent Group3 Fax service types.
- One of the Fixed-Network side tools: a **modem**, an **isdn** or a **Gsm0345** (Transparent Fax relay).

The two tools belonging to the same IWF circuit transfer the payload between them using an internal mechanism of Surf-HMP, and transfers the TDM payload with the external entities using 2 clear voice tools (clear voice is a voice_fe_ip tool configured with codec 'clear'):

- A clear-voice tool is used to carry the TDM between the **mobile** tool and the mobile radio-link network.
- A clear-voice tool carries the TDM between the PSTN/ISDN side and the **modem/isdn/GSM0345** tools.



IWF specific configuration

1. In order to use IWF tools, **block_size** parameter must be set as a system global parameter and for each of the IWF tools as follow:
   - A global **block_size** parameter must be set in the configuration file of Surf-HMP (**config.json**) as follow: "block_size":10,

   - A **block_size** parameter must be set for each tool of the IWF circuit, with the value according the following table:

Tool type / Mode	block_size
mobile / Non-transparent	10
mobile / Transparent	30
isdn	30
modem	30
GSM0345	30
clear voice	Same as the tool that it is connected to on the TDM side

   Note: the block_size parameter is exist for each tool, although it is not documented in each tool's configuration section

2. Fixed side configuration in NT calls:
   - isdn V.110 doesn't support flow control, therefore if possible, it is recommended to use V.120

3. Mobile side configuration in NT calls:
   - RLP V2 has better throughput performance compared to RLP V0 and RLP V1, therefore if possible, RLP V2 should be used
   - In case that RLP V0 is required, it is recommended to use the following parameters of the RLP:
     - **iwf_to_ue_window_size**: 30
     - **ue_to_iwf_window_size**: 30

4. **modem** tool should be set only once, it is not legitimate to perform addition set_config action during its operation

5. **GSM0345** tool should be set only once, it is not legitimate to perform addition set_config action during its operation

6.1 isdn tool

6.1.1 Tool configuration

Parameters:

- **tdm_side_tool_id** – *integer,* Specifies the tool id of the voice_fe_ip tool through which TDM data is sent and received.
  Valid values: 0 – 0xFFFF
- **data_side_tool_id** – *integer,* Specifies the tool id of the IWF tool for the purpose of sending and receiving raw payload data (full duplex, both sides are IWF tools).
  Valid values: 0 – 0xFFFF
- **rate_adapter_mode** – *string,* sets the rate adapter mode for the current session.
  Valid values: "V110_RA", "V120_RA",
  Default value: "V110_RA"
- **bit_rate** – *integer,* sets the bit rate for the current session.
  Valid values: 300,600,1200,2400,4800,7200,9600,12000,14400,19200,28800,38400
  Default value: 19200
- **num_data_bits** – *integer,* number of data bits.
  Valid values: 7,8
  Default value: 8
- **num_stop_bits** – *integer,* number of stop bits.
  Valid values: 1,2
  Default value: 1
- **parity_type** – *string,* parity type.
  Valid values: "NONE", "ODD", "EVEN", "FORCED_0", "FORCED_1"
  Default value: "NONE"
- **flow_control** – *string,* in IWF tool configuration, flow_control parameter MUST be configured to NONE.
  Valid values: "NONE", "OOB", "INBAND"
  Default value: "NONE"
- **iwf_mode** – *string,* defines IWF; when used with Mobile Channel. Defines if transparent or not.
  Valid values: "T_IWF", "NT_IWF"
  Default value: "T_IWF"
- **num_init_sync**– *integer,* number of correct patterns for initial synchronization.
  Valid values: 1-255
  Default value: 2
- **num_loss_sync**– *integer,* number of patterns for loss of synchronization.
  Valid values: 1-255
  Default value: 3
- **num_recovery_sync**– *integer,* number of correct patterns for recovery from loss of synchronization.

Valid values: 1-255
Default value: 2

6.1.2 Events

6.1.2.1 "sb_status_received" event

An event occurs on each change of this connection status bit.
This event represents the received SB status bit transmitted from the remote ISDN terminal or remote ISDN tool. The IWF is able to transfer data through it when both SB and X bits are active and flow even is true.

Parameters:

- **“type”:”sb_status_received"**
- **sb_status** – *integer,* 0,1

6.1.2.2 "x_status_received" event

An event occurs on each change of this connection status bit.
This event represents the received X status bit transmitted from the remote ISDN terminal or remote ISDN tool. The IWF is able to transfer data through it when both SB and X bits are active and flow even is true.

Parameters:

- **“type”:”x_status_received"**
- **x_status** – *integer,* 0,1

6.1.2.3 "flow" event

An event occurs on changing of flow state between "stop data" and "send data".
This event becomes TRUE upon successful synchronization or re-synchronization of the ISDN tool. This event becomes FLASE upon synchronization loss of the ISDN tool.
The IWF is able to transfer data through it when both SB and X bits are active and flow even is true.

Parameters:

- **“type”:”flow"**
- **active** – *Boolean*

6.1.2.4 "break" event

An event that reflects break signal as defined in the link-layer protocols.

Parameters:

- **“type”:”break"**
- **break_type**– *string,* "NONE", "DESTRUCTIVE", "NDE", "NDNE"
  NDE – stands for: Non-Destructive expedited break
  NDNE – stands for: Non-Destructive Non-Expedited break

6.2 mobile tool

6.2.1 Tool configuration

Parameters:

- **tdm_side_tool_id** – integer, Specifies the tool id of the voice_fe_ip tool through which TDM data is sent and received.
  Valid values: 0 – 0xFFFF
- **data_side_tool_id** – integer, Specifies the tool id of the IWF tool for the purpose of sending and receiving raw payload data (full duplex, both sides are IWF tools).
  Valid values: 0 – 0xFFFF
- **transparent_mode** – *boolean,* enables/disables the transparent connection mode.
  Valid values: true/false
  Default value: true
- **channel_coding** – *string,* defines the desired channel coding for call set-up that the mobile tool is preferred to connect with.
  However, if remote side does not support this preferred channel_coding, the chosen channel coding shall be based upon the negotiation between both sides according to their acceptable_coding settings.
  The channel_coding parameter needs to be one of the supported enabled coding in the acceptable_coding parameter.
  Valid values: "ATRAUI", "ATRAU", "V110I_96", "V110I_48"
  Default value: "V110I_96"
- **max_num_sub_channels** – *integer,* defines the maximum number of sub-channels allowed during current session.
  Valid values: 1-4
  Default value: 1
- **acceptable_coding** – *array of strings,* allowed coding type.
  This is the list of all channel codings that are supported by the mobile tool. If the locally configured channel_coding is not supported by the remote side, the channel coding that will be chosen shall be the one that is supported by both the acceptable_coding in both sides.
  Valid values: ["ATRAUI", "ATRAU", "V110I"]
  Default value: "V110I"
- **num_init_sync** – *integer,* defines the number of correct patterns needed for initial synchronization.
  Valid values: 1-255
  Default value: 3
- **num_loss_sync**– *integer,* defines the number of incorrect patterns needed for loss of synchronization.
  Valid values: 1-255
  Default value: 2
- **num_recovery_sync**– *integer,* defines the number of correct patterns needed for recovery from loss of synchronization.
  Valid values: 1-255
  Default value:3

- **connection –** Includes control information related to connection parameters
  - **num_data_bits** – *integer,* number of data bits.
    Valid values: 7-8
    Default value: 8
  - **num_stop_bits** - *integer,* number of stop bits.
    Valid values: 1-2
    Default value: 1
  - **parity_type** – *string,* parity type.
    Valid values: "NONE", "ODD", "EVEN", "FORCED_0", "FORCED_1"
    Default value: "NONE"
  - **flow_control** – *string,* flow control type.
    Valid values: "NONE", "OOB", "INBAND"
    Default value: "NONE"
- **transparent** – Includes control information related to transparent calls
  - **service_type** – *string,* type of IWF service.
    Valid values: "BEARER", "FAX"
    Default value: "BEARER"
  - **user_rate** – *integer,* user rate for transparent call.
    Valid values: 300 ,1200 ,2400 ,4800 ,9600 ,14400 ,19200 ,28800 ,33600 ,38400 ,43200 ,57600
    Default value: 9600
- **non_transparent** – Includes control information related to non-transparent calls
  - **num_sub_channels**– *integer,* desired number of sub-channel at channel initialization.
    Valid values: 1-4
    Default value: 1
  - **aiur**- *integer,* air interface user rate.
    Valid values: 9600 ,14400 ,19200 ,28800 ,38400 ,43200 ,57600
    Default value: 14400
  - **num_rlp_frame_repeat**– *integer,* number of RLP frame repetitions.
    Valid values: 0-225
    Default value: 4
  - **selective_reject**– *string,* defines the type of reject frames used: REJ,SREJ,Info+SREJ.
    Valid values: "DISABLED", "S", "S_I"
    Default value: "DISABLED"
- **rlp_params** – Includes control information related to RLP parameters
  - **rlp_version** – *integer,* negotiated RLP version.
    Valid values: 0-2
    Default value: 2
  - **iwf_to_ue_window_size**- *integer,* IWF to UE window size – One of the XID parameters.
    Valid values: 0-495
    Default value: 240

- **ue_to_iwf_window_size** – *integer,* UE to IWF window size– One of the XID parameters.
  Valid values: 0-495
  Default value: 240
- **acknowledgement_timer_t1**– *integer,* one of the XID parameters.
  Valid values: 380-1000
  Default value: 520
- **replay_delay_timer_t2** - *integer,* one of the XID parameters.
  Valid values: 0-80
  Default value: 75
- **retransmission_attempts_n2** - *integer,* one of the XID parameters.
  Valid values: 6-250
  Default value: 6
- **resequencing_timer_t4 -** *integer,* one of the XID parameters.
  Valid values: 25-500
  Default value: 50

- **nt_v42bis–** Includes control information related to V.42bis parameters
  - **data_compression_mode_pt**– *string,* defines the data compression mode as either V.42bis or None.
    Valid values: "NONE", "V42BIS_MNP5"
    Default value: "V42BIS_MNP5"
  - **direction**- *string,* data compression direction.
    Valid values: "NONE", "TX", "RX", "BOTH"
    Default value: "BOTH"
  - **dictionary_size** – *integer,* define the V.42bis dictionary size "P1".
    Valid values: 512,1024,2048
    Default value: 1024
  - **max_string_size** – *integer,* data compression maximum string size.
    Valid values: 6-250
    Default value: 20

Note: When performing handover, reconfiguration of the mobile tool is required, except for the following cases:

1. When original channel coding was V.110'_9600, and new channel coding is ATRAU, and allowed channel coding contains both V.110' and ATRAU.
2. When original channel coding was ATRAU, and new channel coding is V.110'_9600, and allowed channel coding contains both V.110' and ATRAU.
3. When original channel coding was ATRAU', and new channel coding is V.110'_9600, and allowed channel coding contains both V.110', ATRAU and ATRAU'.
4. When original channel coding was V.110'_9600, and new channel coding is ATRAU', and allowed channel coding contains both V.110', ATRAU and ATRAU'.
5. When original channel coding was ATRAU', and new channel coding is ATRAU, and allowed channel coding contains both V.110', ATRAU and ATRAU'.
6. When original channel coding was ATRAU, and new channel coding is ATRAU', and allowed channel coding contains both V.110', ATRAU and ATRAU'.

In all other cases, reconfirmation is needed.

When performing reconfiguration of a mobile connection, the mobile tool first needs to be removed, and then created and reconfigured to the new set of parameters.

6.2.2 Events

6.2.2.1 "sb_status_received" event

An event occurs on each change of this connection status bit.
It represents the received SB status bit from the remote side.
The behavior of X, SA and SB is according to the V.110 standard (section 5.1.2.3 in the standard).

Parameters:

- **“type”: ”sb_status_received"**
- **sb_status** – *integer,* 0,1

6.2.2.2 "sa_status_received" event

An event occurs on each change of this connection status bit.
It represents the received SA status bit from the remote side.
The behavior of X, SA and SB is according to the V.110 standard (section 5.1.2.3 in the standard).

Parameters:

- **“type”: ”sa_status_received"**
- **sa_status** – *integer,* 0,1

6.2.2.3 "x_status_received" event

An event occurs on each change of this connection status bit.
It represents the received X status bit from the remote side.
The behavior of X, SA and SB is according to the V.110 standard (section 5.1.2.3 in the standard).

Parameters:

- **“type”: ”x_status_received"**
- **x_status** – *integer,* 0,1

6.2.2.4 "sc_state_change" event

Event on a state change in the mobile tool’s state machine.
(SC = split/combine) This event is set to CONNECTED when mobile tool is synchronized.
This event is set to SEQ_LOSS when mobile tool loses its synchronization.

Parameters:

- **“type”: ”sc_state_change"**
- **sc_state** – *string,* "DISCONNECTED”, ”CONNECTED”, ”SEQ_LOSS"

6.2.2.5 "rlp_hang_up" event

Event on hang up.

Parameters:

- **“type”: ”rlp_hang_up "**

6.2.2.6 "rlp_link" event

Event on link established.

Parameters:

- **“type”: ” rlp_link "**
- **rlp_hang_up** – *Boolean,* true/false

6.2.3 Statuses

6.2.3.1 sync

Synchronization statistics.

Parameters:

- **“type” : ”sync”**
- **sync_state_1** – *string,* indicates the synchronization state of sub- channel 1.
  Valid values: "NO_INITIAL_SYNC", "SYNCHRONIZED", "SYNC_ERR_ON", "SYNC_ERR_IMPULSE"
  Default value: "NO_INITIAL_SYNC"
- **sync_state_2** – *string,* indicates the synchronization state of sub- channel 2.
  Valid values: "NO_INITIAL_SYNC", "SYNCHRONIZED”, “SYNC_ERR_ON", "SYNC_ERR_IMPULSE"
  Default value: "NO_INITIAL_SYNC"
- **sync_state_3** – *string,* indicates the synchronization state of sub- channel 3.
  Valid values: "NO_INITIAL_SYNC", "SYNCHRONIZED", "SYNC_ERR_ON", "SYNC_ERR_IMPULSE"
  Default value: "NO_INITIAL_SYNC"
- **sync_state_4** – *string,* indicates the synchronization state of sub- channel 4.
  Valid values: "NO_INITIAL_SYNC”, “SYNCHRONIZED”, “SYNC_ERR_ON”, “SYNC_ERR_IMPULSE"
  Default value: "NO_INITIAL_SYNC"

6.2.3.2 general

Synchronization statistics.

Parameters:

- **“type” : ”general”**

- **channel_coding** – *string,* indicates used channel coding for the current connection.
  Valid values: "ATRAUI", "ATRAU", "V110I_96", "V110I_48"
  Default value: "ATRAUI"
- **num_sub_channels** – *integer,* indicates the number of established sub channels.
  Valid values: 1-4
  Default value: 1

6.2.3.3 connection

Connection statistics.

Parameters:

- **"type" : "connection"**
- **atraul_rx_data_rate** – *integer,* indicates the RX-used data rate (received in the first ATRAU frame).
  Valid values: 9600,14400,19200,28800,38400,43200,57600
  Default value: 9600
- **atraul_tx_data_rate**– *integer,* indicates the TX used data rate (received in the second ATRAU frame).
  Valid values: 9600,14400,19200,28800,38400,43200,57600
  Default value: 9600

6.2.3.4 non_transparent

Non transparent statistics.

Parameters:

- **"type" : "non_transparent"**
- **ra_num_rec_rlp_frames** - *integer,* indicates the number of received RLP frames.
  Valid values: 0 - (2^32-1)
  Default value: 0
- **ra_num_rec_rlp_dtx_1** – *integer,* indicates the number of received RLP frames with DTX=1.
  Valid values: 0 - (2^32-1)
  Default value: 0
- **a_num_corrupted_rlp_frames** – *integer,* indicates the number of corrupted RLP frames (FCS incorrect).
  Valid values: 0 - (2^32-1)
  Default value: 0
- **ra_num_invalid_rlp_frames** – *integer,* indicates the number of invalid RLP frames (FSI incorrect).
  Valid values: 0 - (2^32-1)
  Default value: 0
- **RLP_nof_received_frames** – *integer,* indicates the number of frames received by the RLP.

Valid values: 0 - (2^32-1)
Default value: 0

- **RLP_nof_resent_frames** – *integer,* indicates the number of frames that were resent.
  Valid values: 0 - (2^32-1)
  Default value: 0
- **RLP_nof_sent_bytes** – *integer,* indicates the number of information bytes that were sent by the RLP.
  Valid values: 0 - (2^32-1)
  Default value: 0
- **RLP_nof_received_bytes** – *integer,* indicates the number of bytes received by the RLP.
  Valid values: 0 - (2^32-1)
  Default value: 0
- **RLP_nof_sent_frames** – *integer,* indicates the number of frames sent by the RLP.
  Valid values: 0 - (2^32-1)
  Default value: 0

6.2.3.5 nt_rlp

General statistics.

Parameters:

- **"type" : "nt_rlp"**
- **rlp_version** – *integer,* negotiated RLP version.
  Valid values: 0,1,2
  Default value: 0
- **iwf_to_ue_window_size** – *integer,* one of the XID parameters.
  Valid values: 0-495
  Default value: 0
- **ue_to_iwf_window_size** – *integer,* one of the XID parameters.
  Valid values: 0-495
  Default value: 0
- **acknowledgement_timer_t1** – *integer,* one of the XID parameters.
  Valid values: 380 - 2^32-1
  Default value: 0
- **replay_delay_timer_t2** – *integer,* one of the XID parameters.
  Valid values: 0 - 80
  Default value: 0
- **retransmission_attempts_n2** – *integer,* one of the XID parameters.
  Valid values: 0 - 2^32-1
  Default value: 0
- **resequencing_timer_t4** – *integer,* one of the XID parameters.
  Valid values: 25 - 2^32-1
  Default value: 0

6.2.3.6 nt_v42bis

General statistics.

Parameters:

- **“type” : ”nt_v42bis”**
- **rlp_rx_compression_efficiency** - *integer,* compression efficiency on data received.
  Valid values: 0 - 100
  Default value: 100
- **rlp_tx_compression_efficiency** – *integer,* compression efficiency on data transmitted.
  Valid values: 0 - 100
  Default value: 100
- **current_v42bis_dictionary_size** – *integer,* describes the used V42bis dictionary size after XID negotiation.
  Valid values: 512 - 65535
  Default value: 512
- **data_compression_mode_pt** – *string,* describes the type of compression used as explained in the RLP standard.
  Valid values: "NONE”, “V42BIS_MNP5”, “V44"
  Default value: "V42BIS_MNP5"
- **direction_p0** – *string,* compression direction.
  Valid values: "NONE”, ”TX”, ”RX”, ”BOTH"
  Default value: "BOTH"
- **dictionary_size_p1** – *integer,* XID configured compression dictionary size.
  Valid values: 512,1024,2048
  Default value: 512
- **max_string_size_p2** – *integer,* maximal compression string size.
  Valid values: 6 - 250
  Default value: 6

6.3 modem tool

6.3.1 Tool configuration

Parameters:

- **tdm_side_tool_id** – integer, Specifies the tool id of the voice_fe_ip tool through which TDM data is sent and received.
  Valid values: 0 – 0xFFFF
- **data_side_tool_id** – integer, Specifies the tool id of the IWF tool for the purpose of sending and receiving raw payload data (full duplex, both sides are IWF tools).
  Valid values: 0 – 0xFFFF
- **org_ans -** *string,* configures the tool to be a call originator or call answerer of the call.
  Valid values: "ORG”, “ANS"
  Default value: “ORG”
- **coding** – *string,* defines if the coding type of the TDM samples are G.711 A-Low or G.711 Mu-Low.
  Valid values "ALAW”, “ULAW"
  Default value: "ALAW"

- **DP_protocol**
    - **modem_protocol**– *string,* defines the desired Modem protocol.
      Valid values: "V21", "V22BIS", "V23", "V32BIS", "V34", "BELL103", "BELL212A", "V34CLIENT_BS_10MS"
      Default value: "V21"
    - **auto_modulation** – *string,* enables/disables the auto modulation.
      Valid values: "V8", "AUTO", "V8_AUTO"
      Default value: "V8_AUTO"
    - **min_rx_bit_rate** – *integer,* defines the minimum Rx rate in bps.
      Valid values: 0, 75, 300, 600, 1200, 2400, 4800, 7200, 9600, 12000, 14400, 16800, 19200, 21600, 24000, 26400, 28800, 31200, 33600
      Default value: 1200
    - **max_rx_bit_rate** – *integer,* defines the maximum Rx rate in bps.
      Valid values: 0, 75, 300, 600, 1200, 2400, 4800, 7200, 9600, 12000, 14400, 16800, 19200, 21600, 24000, 26400, 28800, 31200, 33600
      Default value: 33600
    - **min_tx_bit_rate**– *integer,* defines the minimum Tx rate in bps.
      Valid values: 0, 75, 300, 600, 1200, 2400, 4800, 7200, 9600, 12000, 14400, 16800, 19200, 21600, 24000, 26400, 28800, 31200, 33600
      Default value: 1200
    - **max_tx_bit_rate**– *integer,* defines the maximum Tx rate in bps.
      Valid values: 0, 75, 300, 600, 1200, 2400, 4800, 7200, 9600, 12000, 14400, 16800, 19200, 21600, 24000, 26400, 28800, 31200, 33600
      Default value: 33600
- **signal_control**–
    - **nominal_transmit_power** – *integer,* defines the digital Modem nominal transmit power for phase 2.
      Valid values: (-21) - (-6)
      Default value: -14
    - **symbol_rates** – *array of integers,* sets the allowed symbol rates for V.34 and V.90.
      Valid values: [2400, 2743, 2800, 3000, 3200, 3429]
    - **tx_power** – *integer,* defines the modem transmission power in dBm units.
      Valid values: (-43) - 10
      Default value: -12
- **error_correction** –
    - **error_correction_mode** – *string,* defines the error correction mode.
      Valid values: "RELIABLE_LINK", "AUTO_RELIABLE"
      Default value: "AUTO_RELIABLE"
    - **error_correction_protocol** – *string,* defines the error correction protocol.
      Valid values: "NONE", "MNP", "V42_LAPM", "V42_OR_MNP"
      Default value: "V42_LAPM"
    - **frame_check_sequence** – *string,* defines the V.42 frame check sequence length.
      Valid values: "CRC_16", "CRC_32"
      Default value: "CRC_16"

- **selective_reject**– *boolean,* enable/disable V.42 selective reject.
  Valid value: true/false
  Default value: false
- **rx_window_size** – *integer,* V.42 Rx window size.
  Valid values: 1-15
  Default value:15
- **tx_window_size** – *integer,* V.42 Tx window size.
  Valid values: 1-15
  Default value:15
- **v42_n400** – *integer,* retry counter N400 for V.42.
  Valid values: 1-255
  Default value:12

- **EC_connection**
  - **num_data_bits** – *integer,* number of data bits.
    Valid values: 7-8
    Default value: 8
  - **num_stop_bits** - *integer,* number of stop bits.
    Valid values: 1-2
    Default value: 1
  - **parity_type** – *string,* parity type.
    Valid values: "NONE", "ODD", "EVEN", "FORCED_0", "FORCED_1"
    Default value: "NONE"
  - **flow_control** – *string,* flow control type.
    Valid values: "NONE", "OOB", "INBAND"
    Default value: "NONE"

- **data_compression**
  - **mode** – *string,* defines the data compression mode V.42bis, NONE.
    Valid values: "NONE", "V42BIS_MNP5".
    Default value: "V42BIS_MNP5"
  - **direction** – *string,* data compression direction, same for both protocols.
    Valid values: "NONE", "TX", "RX", "BOTH"
    Default value: "BOTH"
  - **v42bis** – V.42bis related controls.
    - **dictionary_size** – *integer,* defines the V.42bis dictionary size P1.
      Valid values: 512,1024,2048
      Default value: 512
    - **max_string_size** – *integer,* data compression maximum string size.
      Valid values: 6-250
      Default value: 50

6.3.2 Commands

6.3.2.1 initiate_retrain

- additional parameters: none

6.3.2.2 initiate_rate_renegotiation

- additional parameters: none

6.3.2.3 hangup

Hangup the call.

- **“cmd_type”:” hangup”**

6.3.3 Events

6.3.3.1 "connection_failed" event

This event provides clarification on the reason for failure of the modem to connect with the remote modem.

Parameters:

- **“type”: ”connection_failed"**
- **"LOCAL_DISCONNECT"** – *string,* "LOCAL_DISCONNECT”, “REMOTE_DISCONNECT”, “NO_ANSWER”, “DP_PROTOCOL_INCOMPATABILITY”, “CDC_PROTOCOL_INCOMPATABILITY”, “FAILURE_DURING_TRAINING”, “FAILURE_DURING_RETRAIN”, “CARRIER_LOST”, “INACTIVITY_TIMEOUT”, “SWITCH_TO_AUDIO”, “SWITCH_TO_VBD"

6.3.3.2 "data_pump_state_change" event

This event reports on the current data-pump connection phase.

Parameters:

- **“type”: ” data_pump_state_change"**
- **data_pump_state** – *string,* "PHASE1”, “PHASE2”, “PHASE3”, “PHASE4”, “DATA"

6.3.3.3 "state_change" event

An event occurs on changing of the connection status between Training phase and Data phase.

Parameters:

- **“type”: ” state_change"**
- **state_change** – *string,* "TRAINING”, “DATA"

6.3.3.4 "sc_state_change" event

This event reports the state change of the error correction layer.

Parameters:

- **“type”: ”sc_state_change"**
- **sc_state** – *string,* "DISCONNECTED”, ”CONNECTED”, ”SEQ_LOSS"

6.3.4 Statuses

6.3.4.1 link

Link statistics.

Parameters:

- **“type” : ”link”**
- **current_line_transmit_rate** - *integer,* indicates the transmission rate of the current connection in bit/s.
  Valid values 0, 75, 300, 600, 1200, 2400, 4800, 7200, 9600, 12000, 14400, 16800, 19200, 21600, 24000, 26400, 28800, 31200, 33600
- **current_line_receive_rate** – *integer,* indicates the receiving rate of the current connection in bit/s.
  Valid values: 0, 75, 300, 600, 1200, 2400, 4800, 7200, 9600, 12000, 14400, 16800, 19200, 21600, 24000, 26400, 28800, 31200, 33600
- **modem_protocol** – *string,* indicates the Modem DP protocol, used in the current or the previous call.
  Valid values: "V21”, ”V22BIS”, ”V23”, ”V32BIS”, ”V34”, ”BELL103”, ”BELL212A”, ”V34CLIENT_BS_10MS"
- **rx_level_dBm** – *integer,* indicates the Rx power level in dBm.
  Valid values: (-43) - 10
- **tx_level_dBm** – *integer,* indicates the Tx power level in dBm units.
  Valid values: (-43) - 10
- **rx_symbol_rate** – *integer,* indicates the Rx symbol rate in symbols sec.
  Valid values: 0, 75, 300, 600, 1200, 2400, 2743, 2800, 3000, 3200, 3429, 8000
- **tx_symbol_rate** – *integer,* indicates the Tx symbol rate in symbol, sec.
  Valid values: 0, 75, 300, 600, 1200, 2400, 2743, 2800, 3000, 3200, 3429, 8000
- **pdsnr** – *integer,* indicates the Post Detection Signal to Noise Ratio in dB.
  Valid values: 0-50 dB
- **rtd** – *integer,* indicates the Round Trip Delay in 125µs units (PCM sample.
  Valid values: 0-8000 samples
- **timing_offset** – *integer,* indicates the timing offset.
  Valid values: (-500) - 500 PPM

- **carrier_offset** – *integer,* indicates the carrier offset.
  Valid values: (-10) - 10 Hz

6.3.4.2 modem

Modem statistics.

Parameters:

- **"type" : "modem"**
- **num_of_retrains** – *integer,* indicates the number of retrains performed during the current call.
  Valid values: 0 - 2^32-1
- **num_of_rate_renegotiations** – *integer,* indicates the number of rate renegotiations performed during the current call.
  Valid values: 0 - 2^32-1
- **local_renegotiation_count** – *integer,* indicates the number of local renegotiations performed during the current call.
  Valid values: 0 - 2^32-1
- **remote_renegotiation_count** – *integer,* indicates the number of remote renegotiations performed during the current call.
  Valid values: 0 - 2^32-1

6.3.4.3 error correction

Error correction statistics.

Parameters:

- **"type" : " error_correction"**
- **ec_protocol -** *string,* EC protocol, used in current or previous call.
  Valid values: "NONE", "MNP", "V42_LAPM"
- **frame_check_sequence** – *string,* indicates the chosen V.42 frame check sequence.
  Valid values: CRC_16", "CRC_32"

6.3.4.4 data compression

Compression statistics.

Parameters:

- **"type" : " data_compression"**
- **dc_protocol -** *string,* the data compression protocol used.
  Valid values: "NONE", "MNP5", "V42BIS"
- **direction** – *integer,* the data compression direction used.
  Valid values: "NONE", "TX", "RX", "BOTH"
- **tx_compression_efficiency** – *integer,* the transmission compression efficiency.
  Valid values: 0 - 100
- **rx_compression_efficiency** – *integer,* the receive compression efficiency.
  Valid values: 0 - 100

- **v42bis**
    - **dictionary_size** – *integer,* indicates the V.42bis selected dictionary size.
      Valid values: 512,1024,2048
    - **max_string_size** – *integer,* indicates the V.42bis chosen maximum string size.
      Valid values: 6-250

6.3.4.5 V42 frames

V424 frame statistics.

Parameters:

- **"type" : " v42_frames"**
- **num_of_sent_data_frames -** *integer,* indicates the number of data frames sent on the line interface.
  Valid values: 0 - 2^32-1
- **num_of_received_data_frames** – *integer,* indicates the number of data frames received on the line interface.
  Valid values: 0 - 2^32-1
- **num_of_resent_frames -** *integer,* indicates the number of retransmitted frames.
  Valid values: 0 - 2^32-1
- **num_of_error_frames** – *integer,* indicates the number of error frames received on the line interface.
  Valid values: 0 - 2^32-1
- **num_of_missing_frames -** *integer,* indicates the number of missing frames received on the line interface.
  Valid values: 0 - 2^32-1

6.4 GSM0345 tool

6.4.1 Tool configuration

Parameters:

- **tdm_side_tool_id** – integer, Specifies the tool id of the voice_fe_ip tool through which TDM data is sent and received.
  Valid values: 0 – 0xFFFF
- **data_side_tool_id** – integer, Specifies the tool id of the IWF tool for the purpose of sending and receiving raw payload data (full duplex, both sides are IWF tools).
  Valid values: 0 – 0xFFFF
- **org_ans -** *string,* configure the tool to be a call originator or call answerer of the call.
  Valid values: "ORG", "ANS"
  Default value: "ORG"

- **nsf_manipulation** – *boolean,* enable only T.30 fax session.
  Valid values: true/false
  Default value: false
- **max_fax_rate** – *integer,* defines the maximum data rate for the fax tool. Max_fax_rate must be equal to the value of the user_rate field of the transparent parameters group within the Mobile channel of the same IWF circuit.
  Valid values: 2400, 4800, 7200, 9600, 12000, 14400
  Default value: 2400

6.4.2 Commands

6.4.2.1 hangup

Hangup the call.

- **“cmd_type”:” hangup”**

6.4.3 Events

6.4.3.1 "connection_failed" event

This event provides information on the call disconnect reason.

Parameters:

- **“type”: ”connection_failed"**
- **"LOCAL_DISCONNECT"** – *string,* "LOCAL_DISCONNECT”, “REMOTE_DISCONNECT”, “NO_ANSWER", "DP_PROTOCOL_INCOMPATABILITY”, “ECDC_PROTOCOL_INCOMPATABILITY", "FAILURE_DURING_TRAINING" ,"FAILURE_DURING_RETRAIN", "CARRIER_LOST" ,"INACTIVITY_TIMEOUT", "SWITCH_TO_AUDIO", "SWITCH_TO_VBD", "WRONG_FAX_SPEED"

6.4.3.2 "state_change" event

An event occurs on changing of the connection status between Training phase and Data phase.

Parameters:

- **“type”: ” state_change"**
- **state_change** – *string,* "TRAINING”, “DATA"

6.4.4 Statuses

6.4.4.1 connection

Connection statistics.

Parameters:

- **“type” : ” connection”**
- **fax_protocol –** *string,* indicates the current phase Fax protocol is in use.
  Valid values: "V17_TX", "V17_RX", "V21_TX", "V21_RX", "V27TER_TX", "V27TER_RX", "V29_TX”, “V29_RX”
  Default value: “V17_TX"
- **fax_state** – *string,* indicates the state of the Fax.
  Valid values: "IDLE”, “START", "V21RX_START", "V21RX_PREAMBLE", "V21RX_T30", "V21RX_END”, “V21TX_START”, “V21TX_PREAMBLE", "V21TX_T30", "V21TX_END", "IMGRX_START", "IMGRX_TRN", "IMGRX_TCF", "IMGRX_DATA", "IMGRX_END", "IMGTX_START", "IMGTX_TRN”, “IMGTX_TCF”, “IMGTX_END"
  Default value: “IDLE"
- **call_fax_phase -** *string,* indicates the current phase (T.30) of the Fax call.
  Valid values: "T30_IDLE", "T30_PHASE_A", "T30_PHASE_B", "T30_PHASE_C", "T30_PHASE_D", "T30_PHASE_E"
  Default value: "T30_IDLE"
- **carrier_type** – *string,* detected carrier.
  Valid values: "CNG”, “CED”, “BCS”, “MSG”, “CARR_OFF"
  Default value: "CNG"
- **fax_rate-** *string,* indicates the rate of the Fax DP.
  Valid values "V17144S", "V17144L”, “V1712S", "V1712L", "V1796S”, “V1796L", "V1772S", "V1772L", "V2996", "V2972", "V2748", "V2724", "V21HI"
  Default value: "V17144S"
- **rx_state -** *string,* fax adaptor receives.
  Valid values "IDLE”, “BCS_REC”, “MSG_TRA”, “MSG_REC”, “DATA”, “NO_SYNC"
  Default value: "IDLE"
- **tx_state -** *string,* fax adaptor transmits.
  Valid values "IDLE”, “BCS_REC”, “MSG_TRA”, “MSG_REC”, “DATA”, “NO_SYNC"
  Default value: "IDLE"

6.4.4.2 frame

Frame statistics.

Parameters:

- **“type” : ” frame”**
- **num_rec_bcs_frames -** *integer,* number of received BCS frames.
  Valid values: 0 - 2^32-1
  Default value: 0

- **num_tran_bcs_frames –** *integer,* number of transmitted BCS frames.
  Valid values: 0 - 2^32-1
  Default value: 0
- **num_corrupted_bcs_frames -** *integer,* number of corrupted BCS frames.
  Valid values: 0 - 2^32-1
  Default value: 0

6.4.4.3 info

Information statistics.

Parameters:

- **“type” : ” info”**
- **transfer_direction -** *string,* transmission direction of BCS frame.
  Valid values: "MS2NW”, “NW2MS"
  Default value: "MS2NW"
- **bcs_frame_valid** – *boolean,* BCS frame validity.
  Valid values: true/false
  Default value: true
- **addr_valid** – *boolean,* address valid.
  Valid values: true/false
  Default value: true
- **ctrl** – *string,* control field status.
  Valid values: "FIN”, “NON_FIN”, “NON_VALID"
  Default value: "FIN"

7. Error codes

Here is the list of error codes that Media Processor may return:

ERROR_OK	0
ERROR_TOOL_PARAMETER_SET_FAILED	1
ERROR_GENERAL	10000
ERROR_TOOL_DOES_NOT_EXIST,	10001
ERROR_FEATURE_IS_DISABLED,	10002
ERROR_MAX_NUM_END_POINTS_REACHED,	10003
ERROR_MAX_NUM_VIDEO_CODEC_TOOLS_REACHED,	10004
ERROR_MAX_RESOLUTION_IS_NOT_ALLOWED,	10005
ERROR_MEMORY_ALLOCATION_FAILED,	10006
ERROR_CANNOT_BE_CHANGED,	10007
ERROR_SOCKET_ALLOCATION_FAILED,	10008
ERROR_SOCKET_BIND_FAILED,	10009
ERROR_ILLEGAL_COMMAND,	10010
// Voice	
ERROR_VOICE_CMD_EVG_DISABLED,	10011
// Video	
ERROR_VIDEO_FILE_READER_CONFIG_ERROR,	10012
ERROR_VIDEO_MIXER_CONFIG_ERROR,	10013
// File format	
ERROR_FILE_CANT_PERFORM_COMMAND_IN_STATE,	10014
ERROR_FILE_CREATION_ERROR,	10015
ERROR_FILE_STREAM_CREATION_ERROR,	10016