Asynchronous event/state notifications in the Janus WebRTC server
Providing administrators and developers with more tools to manage a Janus instance

Lorenzo Miniero
@elminiero

FOSDEM 2017 Real Time devroom
4th February 2017, Brussels ☕️
Janus: a general purpose WebRTC gateway (application) server
Where were we? (FOSDEM 2016)
First take at monitoring: the Admin API

A new type of Janus plugins: Event Handlers
An asynchronous approach to monitoring/troubleshooting
Real example: integrating Homer as a monitoring framework

Playing with Event Handlers: a look at the real thing
Monitoring a single PeerConnection: Echo Test
Correlating multiple PeerConnections: Video Room

Next steps
Janus: a general purpose WebRTC gateway server

- A door between the communications past and future
  - Legacy technologies (the “past”)
  - WebRTC (the “future”)

Janus
General purpose, open source WebRTC gateway

- https://github.com/meetecho/janus-gateway
- Demos and documentation: https://janus.conf.meetecho.com
- Community: https://groups.google.com/forum/#!forum/meetecho-janus

Janus
Where were we?
Admin API
Monitoring
Event Handlers
Homer/HEP
Demo
EchoTest
VideoRoom
Next steps
A quick recap: modular architecture

- The core only implements the WebRTC stack
  - JSEP/SDP, ICE, DTLS-SRTP, Data Channels, ...
- Plugins expose Janus API over different “transports”
  - Currently HTTP / WebSockets / RabbitMQ / Unix Sockets / MQTT
- “Application” logic implemented in plugins too
  - Users attach to plugins via the Janus core
  - The core handles the WebRTC stuff
  - Plugins route/manipulate the media/data
- Plugins can be combined on client side as “bricks”
  - Video SFU, Audio MCU, SIP gatewaying, broadcasting, etc.

FOSDEM 2016 slides

Anything wrong? Check the Admin API!

- Requests/response API to interrogate Janus
  - Query server capabilities
  - Control some aspects (e.g., enable/disable debugging)
  - Inspect handles and WebRTC “internals”
  - ... assuming you know the identifiers to query (session/handle)

http://www.meetecho.com/blog/understanding-the-janus-admin-api/
Admin API overview: polling for information

- create session
  - success
  - attach handle
  - success
  - SDP offer
  - trickle candidate(s)
  - SDP answer

Admin API request
Admin API response

Next steps
Admin API overview: polling for information

```json
{
  "session_id": 6281631500841242,
  "handle_id": 6383008526518405,
  "plugin": "janus.plugin.echotest",
  [
  ]
  "plugin_specific": {
    [
    ]
  },
  "flags": {
    [
    ]
  },
  "sdps": {
    [
    ]
  },
  "streams": [
    [
    ]
  ]
}
```
Admin API overview: polling for information

```json
{
  "session_id": 6281631500841242,
  "handle_id": 6383008526518405,
  "plugin": "janus.plugin.echotest",
  "plugin_specific": {
    
  }
  "flags": {
    
  }
  "sdps": {
    
  }
  "streams": [
    
  ]
}
```
Admin API overview: polling for information

```json
{
    "session_id": 6281631500841242,
    "handle_id": 6383008526518405,
    "plugin": "janus.plugin.echotest",
    [..]
    "plugin_specific": {
        [..]
    },
    "flags": {
        [..]
    },
    "sdps": {
        [..]
    },
    "streams": [
        [..]
    ]
}
```
Admin API overview: polling for information

```json
{
    "session_id": 6281631500841242,
    "handle_id": 6383008526518405,
    "plugin": "janus.plugin.echotest",
    [..]
    "plugin_specific": {
        [..]
    },
    "flags": {
        [..]
    },
    "sdps": {
        [..]
    },
    "streams": [
        [..]
    ]
}
```
Admin API overview: polling for information

```json
{
    "session_id": "6281631500841242",
    "handle_id": "6383008526518405",
    "plugin": "janus.plugin.echotest",
    [..]
    "plugin_specific": {
        [..]
    },
    "flags": {
        [..]
    },
    "sdps": {
        [..]
    },
    "streams": [
        [..]
    ]
}
```
Admin API overview: polling for information

```json
{
    "session_id": 6281631500841242,
    "handle_id": 6383008526518405,
    "plugin": "janus.plugin.echotest",
    [..]
    "plugin_specific": {
        [..]
    },
    "flags": {
        [..]
    },
    "sdps": {
        [..]
    },
    "streams": [
        [..]
    ]
}
```
An asynchronous approach to monitoring/troubleshooting

- Admin API is cool, but is request/response...
  - Needs you to poll again to see changes
  - Information is lost when session/handle is gone
  - What about an asynchronous approach instead?

- A new mechanism: Event Handlers
  - Core and plugins generate events
    - Shared “header” (e.g., to identify target of event)
    - Different type of events allows for filtering
  - Custom modules can subscribe to and handle them
    - e.g., save to DB, send to external service, CDR, etc.
  - *Sample Event Handler* forwards JSON events via HTTP
    - Third-party plugins may provide integration with existing frameworks
An asynchronous approach to monitoring/troubleshooting

- Admin API is cool, but is request/response...
  - Needs you to poll again to see changes
  - Information is lost when session/handle is gone
  - What about an asynchronous approach instead?

- A new mechanism: Event Handlers
  - Core and plugins generate events
    - Shared “header” (e.g., to identify target of event)
    - Different type of events allows for filtering
  - Custom modules can subscribe to and handle them
    - e.g., save to DB, send to external service, CDR, etc.
  - Sample Event Handler forwards JSON events via HTTP
    - Third-party plugins may provide integration with existing frameworks
An asynchronous approach to monitoring/troubleshooting

- Admin API is cool, but is request/response...
  - Needs you to poll again to see changes
  - Information is lost when session/handle is gone
  - What about an asynchronous approach instead?

- A new mechanism: Event Handlers
  - Core and plugins generate events
    - Shared “header” (e.g., to identify target of event)
    - Different type of events allows for filtering
  - Custom modules can subscribe to and handle them
    - e.g., save to DB, send to external service, CDR, etc.

  - Sample Event Handler forwards JSON events via HTTP
    - Third-party plugins may provide integration with existing frameworks
An asynchronous approach to monitoring/troubleshooting

- Admin API is cool, but is request/response...
  - Needs you to poll again to see changes
  - Information is lost when session/handle is gone
  - What about an asynchronous approach instead?

- A new mechanism: Event Handlers
  - Core and plugins generate events
    - Shared “header” (e.g., to identify target of event)
    - Different type of events allows for filtering
  - Custom modules can subscribe to and handle them
    - e.g., save to DB, send to external service, CDR, etc.
  - *Sample Event Handler* forwards JSON events via HTTP
    - Third-party plugins may provide integration with existing frameworks
Event Handlers overview: routing and managing async events

1. Janus core/Plugins
2. Events Manager
3. Events Handler 1
4. Events Handler 2

**Notify event (type=X, content)**
- Format and forward to all handlers
- Enqueue event (type=X, content)
- Send to HTTP server as JSON
- Enqueue event (type=X, content)
- Parse / save to DB
Type of events

- Different events you can subscribe to
  - Session related events (e.g., session created/destroyed, etc.)
  - Handle related events (e.g., handle attached/detached, etc.)
  - JSEP related events (e.g., got/sent offer/answer)
  - WebRTC related events (e.g., PeerConnection up/down, ICE state changes, DTLS state, etc.)
  - Media related events (e.g., media started/stopped flowing, stats on packets/bytes, etc.)
  - Plugin-originated events (specific to the application)
  - Transport-originated (specific to the transport)

- Correlation possible on different identifiers
  - Transport instances that originate specific session
  - Opaque ID applications can set on handles of same “user”
  - Plugin-specific identifiers (e.g., in VideoRoom)
Type of events

- Different events you can subscribe to
  - Session related events (e.g., session created/destroyed, etc.)
  - Handle related events (e.g., handle attached/detached, etc.)
  - JSEP related events (e.g., got/sent offer/answer)
  - WebRTC related events (e.g., PeerConnection up/down, ICE state changes, DTLS state, etc.)
  - Media related events (e.g., media started/stopped flowing, stats on packets/bytes, etc.)
  - Plugin-originated events (specific to the application)
  - Transport-originated (specific to the transport)

- Correlation possible on different identifiers
  - Transport instances that originate specific session
  - Opaque ID applications can set on handles of same “user”
  - Plugin-specific identifiers (e.g., in VideoRoom)
Sample Event Handler: a first proof-of-concept

- Example of Event Handler we provide out of the box
- Simply forwards all events as JSON to an HTTP backend
  - Supports basic authentication
  - Can group events (i.e., JSON array vs. multiple JSON objects)
  - Implements simple retransmission (exponential back-off)
- Does nothing more than that: logic needs to be elsewhere
  - HTTP backend decides what to do with events, if anything
  - Demo in a minute to show exactly that!
- May be extended later on with different transports
  - e.g., WebSockets, RabbitMQ, etc.
  - ... or does this belong to different handlers entirely?
Sample Event Handler example: notifying an Admin application

- User
  - create session
  - success
  - attach handle
  - success

- Janus
  - notify event
  - Handle event
    - ex: Forward event (HTTP)
    - 200 OK

- Event Handlers
  - notify event
  - Handle event
    - ex: Forward event (HTTP)
    - 200 OK

- Admin
Sample Event Handler example: notifying an Admin application
A real-world example: Homer/HEP integration!

- Monitoring Janus sessions through a popular tool
  - Homer/HEP intercepts the events Janus sends
  - These events are correlated and displayed
- More on that in just a few minutes!
Meetecho+Janus+Homer: awesome on multiple levels!

https://www.youtube.com/watch?v=TkgDOMSv9PE
Monitoring a single PeerConnection: Echo Test

- Sample Event Handler forwards all events to web backend
- node.js web backend saves events to DB

http://www.meetecho.com/blog/event-handlers-a-practical-example/
Monitoring a single PeerConnection: Echo Test

{
    "type": 1,
    "timestamp": 1485872279008725,
    "session_id": 192621687169800,
    "event": {
        "name": "created",
        "transport": {
            "transport": "janus.transport.http",
            "id": "0x60e00003e160"
        }
    }
}
Monitoring a single PeerConnection: Echo Test

```json
{
  "type": 1,
  "timestamp": 1485872279008725,
  "session_id": 192621687169800,
  "event": {
    "name": "created",
    "transport": {
      "transport": "janus.transport.http",
      "id": "0x60e00003e160"
    }
  }
}
```
Monitoring a single PeerConnection: Echo Test

```json
{
    "type": 1,
    "timestamp": 1485872279008725,
    "session_id": 192621687169800,
    "event": {
        "name": "created",
        "transport": {
            "transport": "janus.transport.http",
            "id": "0x60e00003e160"
        }
    }
}
```
Monitoring a single PeerConnection: Echo Test

```json
{
    "type": 2,
    "timestamp": 1485872279029549,
    "session_id": 192621687169800,
    "handle_id": 7684775393694722,
    "event": {
        "name": "attached",
        "plugin": "janus.plugin.echotest",
        "opaque_id": "echotest-l7jLWaHvyU9q"
    }
}
```
Monitoring a single PeerConnection: Echo Test

```json
{
    "type": 2,
    "timestamp": 1485872279029549,
    "session_id": 192621687169800,
    "handle_id": 7684775393694722,
    "event": {
        "name": "attached",
        "plugin": "janus.plugin.echotest",
        "opaque_id": "echotest-17jLWaHvyU9q"
    }
}
```
Monitoring a single PeerConnection: Echo Test

```json
{
    "type": 2,
    "timestamp": 1485872279029549,
    "session_id": 192621687169800,
    "handle_id": 7684775393694722,
    "event": {
        "name": "attached",
        "plugin": "janus.plugin.echotest",
        "opaque_id": "echotest-l7jLWaHvyU9q"
    }
}
```
Monitoring a single PeerConnection: Echo Test

```json
{
    "type": 8,
    "timestamp": 1485872279129034,
    "session_id": 192621687169800,
    "handle_id": 7684775393694722,
    "event": {
        "owner": "remote",
        "jsep": {
            "type": "offer",
            "sdp": "v=0\r\n[..]"
        }
    }
}
```
Monitoring a single PeerConnection: Echo Test

```json
{
  "type": 8,
  "timestamp": 1485872279129034,
  "session_id": 192621687169800,
  "handle_id": 7684775393694722,
  "event": {
    "owner": "remote",
    "jsep": {
      "type": "offer",
      "sdp": "v=0\r\n[..]"
    }
  }
}
```
Monitoring a single PeerConnection: Echo Test

```
{
    "type": 8,
    "timestamp": 1485872279129034,
    "session_id": 192621687169800,
    "handle_id": 7684775393694722,
    "event": {
        "owner": "remote",
        "jsep": {
            "type": "offer",
            "sdp": "v=0\r\n[..]"
        }
    }
}
```
Monitoring a single PeerConnection: Echo Test

```json
{
    "type": 8,
    "timestamp": 1485872279131464,
    "session_id": 192621687169800,
    "handle_id": 7684775393694722,
    "event": {
        "owner": "local",
        "jsep": {
            "type": "answer",
            "sdp": "v=0\r\n[..]"
        }
    }
}
```
Monitoring a single PeerConnection: Echo Test

```json
{
    "type": 8,
    "timestamp": 1485872279131464,
    "session_id": 192621687169800,
    "handle_id": 7684775393694722,
    "event": {
        "owner": "local",
        "jsep": {
            "type": "answer",
            "sdp": "v=0\r\n[..]"
        }
    }
}
```
Monitoring a single PeerConnection: Echo Test

```json
{
    "type": 8,
    "timestamp": 1485872279131464,
    "session_id": 192621687169800,
    "handle_id": 7684775393694722,
    "event": {
        "owner": "local",
        "jsep": {
            "type": "answer",
            "sdp": "v=0\r\n[..]"
        }
    }
}
```
Monitoring a single PeerConnection: Echo Test

```
{
   "type": 16,
   "timestamp": 1485872279153835,
   "session_id": 192621687169800,
   "handle_id": 7684775393694722,
   "event": {
      "ice": "connecting",
      "stream_id": 1,
      "component_id": 1
   }
}
```
Monitoring a single PeerConnection: Echo Test

```json
{
  "type": 16,
  "timestamp": 1485872279153835,
  "session_id": 192621687169800,
  "handle_id": 7684775393694722,
  "event": {
    "ice": "connecting",
    "stream_id": 1,
    "component_id": 1
  }
}
```
Monitoring a single PeerConnection: Echo Test

```json
{
    "type": 16,
    "timestamp": 1485872279153835,
    "session_id": 192621687169800,
    "handle_id": 7684775393694722,
    "event": {
        "ice": "connecting",
        "stream_id": 1,
        "component_id": 1
    }
}
```
Monitoring a single PeerConnection: Echo Test

```json
{
    "type": 16,
    "timestamp": 1485872279230699,
    "session_id": 192621687169800,
    "handle_id": 7684775393694722,
    "event": {
        "ice": "connected",
        "stream_id": 1,
        "component_id": 1
    }
}
```
Monitoring a single PeerConnection: Echo Test

```json
{
  "type": 16,
  "timestamp": 1485872279230699,
  "session_id": 192621687169800,
  "handle_id": 7684775393694722,
  "event": {
    "ice": "connected",
    "stream_id": 1,
    "component_id": 1
  }
}
```
Monitoring a single PeerConnection: Echo Test

```json
{
    "type": 16,
    "timestamp": 1485872279349240,
    "session_id": 192621687169800,
    "handle_id": 7684775393694722,
    "event": {
        "selected-pair": "192.168.1.69:33755 [host,udp] <-> 192.168.1.69:55537 [host,udp]",
        "stream_id": 1,
        "component_id": 1
    }
}
```
Monitoring a single PeerConnection: Echo Test

```
{
  "type": 16,
  "timestamp": 1485872279349240,
  "session_id": 192621687169800,
  "handle_id": 7684775393694722,
  "event": {
    "selected-pair": "192.168.1.69:33755 [host,udp] <-> 192.168.1.69:55537 [host,udp]",
    "stream_id": 1,
    "component_id": 1
  }
}
```
Monitoring a single PeerConnection: Echo Test

```json
{
  "type": 16,
  "timestamp": 1485872279349713,
  "session_id": 192621687169800,
  "handle_id": 7684775393694722,
  "event": {
    "dtls": "trying",
    "stream_id": 1,
    "component_id": 1
  }
}
```
Monitoring a single PeerConnection: Echo Test

{  
  "type": 16,
  "timestamp": 1485872279349713,
  "session_id": 192621687169800,
  "handle_id": 7684775393694722,
  "event": {
    "dtls": "trying",
    "stream_id": 1,
    "component_id": 1
  }
}
Monitoring a single PeerConnection: Echo Test

```
{
    "type": 16,
    "timestamp": 1485872279349713,
    "session_id": 192621687169800,
    "handle_id": 7684775393694722,
    "event": {
        "dtls": "trying",
        "stream_id": 1,
        "component_id": 1
    }
}
```
Monitoring a single PeerConnection: Echo Test

```json
{
    "type": 16,
    "timestamp": 1485872279361899,
    "session_id": 192621687169800,
    "handle_id": 7684775393694722,
    "event": {
        "dtls": "connected",
        "stream_id": 1,
        "component_id": 1
    }
}
```
Monitoring a single PeerConnection: Echo Test

```json
{
    "type": 16,
    "timestamp": 1485872279361899,
    "session_id": 192621687169800,
    "handle_id": 7684775393694722,
    "event": {
        "dtls": "connected",
        "stream_id": 1,
        "component_id": 1
    }
}
```
Monitoring a single PeerConnection: Echo Test

```json
{
  "type": 16,
  "timestamp": 1485872279362615,
  "session_id": 192621687169800,
  "handle_id": 7684775393694722,
  "event": {
    "connection": "webrtcup"
  }
}
```
Monitoring a single PeerConnection: Echo Test

```json
{
    "type": 16,
    "timestamp": 1485872279362615,
    "session_id": 192621687169800,
    "handle_id": 7684775393694722,
    "event": {
        "connection": "webrtc-up"
    }
}
```
Monitoring a single PeerConnection: Echo Test

```
{
    "type": 16,
    "timestamp": 1485872279362615,
    "session_id": 192621687169800,
    "handle_id": 7684775393694722,
    "event": {
        "connection": "webrtcup"
    }
}
```
Monitoring a single PeerConnection: Echo Test

```json
{
    "type": 32,
    "timestamp": 1485872279386850,
    "session_id": 192621687169800,
    "handle_id": 7684775393694722,
    "event": {
        "media": "audio",
        "receiving": true
    }
}
```
Monitoring a single PeerConnection: Echo Test

```json
{
    "type": 32,
    "timestamp": 1485872279386850,
    "session_id": 192621687169800,
    "handle_id": 7684775393694722,
    "event": {
        "media": "audio",
        "receiving": true
    }
}
```
Monitoring a single PeerConnection: Echo Test

```
{
  "type": 32,
  "timestamp": 1485872279386850,
  "session_id": 192621687169800,
  "handle_id": 7684775393694722,
  "event": {
    "media": "audio",
    "receiving": true
  }
}
```
Monitoring a single PeerConnection: Echo Test

```json
{
    "type": 32,
    "timestamp": 1485872279559121,
    "session_id": 192621687169800,
    "handle_id": 7684775393694722,
    "event": {
        "media": "video",
        "receiving": true
    }
}
```
Monitoring a single PeerConnection: Echo Test

```json
{
  "type": 32,
  "timestamp": 1485872279559121,
  "session_id": 192621687169800,
  "handle_id": 7684775393694722,
  "event": {
    "media": "video",
    "receiving": true
  }
}
```
Monitoring a single PeerConnection: Echo Test

```
{
"type": 32,
"timestamp": 1485872279559121,
"session_id": 192621687169800,
"handle_id": 7684775393694722,
"event": {
  "media": "video",
  "receiving": true
}
}
```
Monitoring a single PeerConnection: Echo Test

```json
{
    "type": 32,
    "timestamp": 1485872280230983,
    "session_id": 192621687169800,
    "handle_id": 7684775393694722,
    "event": {
        "media": "audio",
        "base": 48000,
        "lsr": 454522762,
        "lost": 0,
        "lost-by-remote": 0,
        "jitter-local": 134488,
        "jitter-remote": 0,
        "packets-received": 43,
        "packets-sent": 43,
        "bytes-received": 4089,
        "bytes-sent": 4519,
        "nacks-received": 0,
        "nacks-sent": 0
    }
}
```
Monitoring a single PeerConnection: Echo Test

```
{
  "type": 32,
  "timestamp": 1485872280230983,
  "session_id": 192621687169800,
  "handle_id": 7684775393694722,
  "event": {
    "media": "audio",
    "base": 48000,
    "lsr": 454522762,
    "lost": 0,
    "lost-by-remote": 0,
    "jitter-local": 134488,
    "jitter-remote": 0,
    "packets-received": 43,
    "packets-sent": 43,
    "bytes-received": 4089,
    "bytes-sent": 4519,
    "nacks-received": 0,
    "nacks-sent": 0
  }
}
```
Monitoring a single PeerConnection: Echo Test

```json
{
    "type": 32,
    "timestamp": 1485872280230983,
    "session_id": 192621687169800,
    "handle_id": 7684775393694722,
    "event": {
        "media": "audio",
        "base": 48000,
        "lsr": 454522762,
        "lost": 0,
        "lost-by-remote": 0,
        "jitter-local": 134488,
        "jitter-remote": 0,
        "packets-received": 43,
        "packets-sent": 43,
        "bytes-received": 4089,
        "bytes-sent": 4519,
        "nacks-received": 0,
        "nacks-sent": 0
    }
}
```
Monitoring a single PeerConnection: Echo Test

```
{
    "type": 32,
    "timestamp": 1485872281236922,
    "session_id": 192621687169800,
    "handle_id": 7684775393694722,
    "event": {
        "media": "audio",
        "base": 48000,
        "lsr": 454522762,
        "lost": 0,
        "lost-by-remote": 0,
        "jitter-local": 5338,
        "jitter-remote": 0,
        "packets-received": 93,
        "packets-sent": 93,
        "bytes-received": 8940,
        "bytes-sent": 9870,
        "nacks-received": 0,
        "nacks-sent": 0
    }
}
```
Monitoring a single PeerConnection: Echo Test

{  
  "type": 32,  
  "timestamp": 1485872280231007,  
  "session_id": 192621687169800,  
  "handle_id": 7684775393694722,  
  "event": {  
    "media": "video",  
    "base": 90000,  
    "lsr": 454528520,  
    "lost": 0,  
    "lost-by-remote": 0,  
    "jitter-local": 42459,  
    "jitter-remote": 3,  
    "packets-received": 45,  
    "packets-sent": 45,  
    "bytes-received": 22779,  
    "bytes-sent": 23229,  
    "nacks-received": 0,  
    "nacks-sent": 0  
  }  
}
Monitoring a single PeerConnection: Echo Test

```
{
  "type": 32,
  "timestamp": 1485872280231007,
  "session_id": 192621687169800,
  "handle_id": 7684775393694722,
  "event": {
    "media": "video",
    "base": 90000,
    "lsr": 454528520,
    "lost": 0,
    "lost-by-remote": 0,
    "jitter-local": 42459,
    "jitter-remote": 3,
    "packets-received": 45,
    "packets-sent": 45,
    "bytes-received": 22779,
    "bytes-sent": 23229,
    "nacks-received": 0,
    "nacks-sent": 0
  }
}
```
Monitoring a single PeerConnection: Echo Test

```
{
    "type": 32,
    "timestamp": 1485872281236949,
    "session_id": 192621687169800,
    "handle_id": 7684775393694722,
    "event": {
        "media": "video",
        "base": 90000,
        "lsr": 454619874,
        "lost": 0,
        "lost-by-remote": 0,
        "jitter-local": 6537,
        "jitter-remote": 2,
        "packets-received": 74,
        "packets-sent": 74,
        "bytes-received": 39933,
        "bytes-sent": 40673,
        "nacks-received": 0,
        "nacks-sent": 0
    }
}
```
Monitoring a single PeerConnection: Echo Test

```json
{
    "type": 64,
    "timestamp": 1485872288429976,
    "session_id": 192621687169800,
    "handle_id": 7684775393694722,
    "event": {
        "plugin": "janus.plugin.echotest",
        "data": {
            "audio_active": true,
            "video_active": true,
            "bitrate": 256000
        }
    }
}
```
Monitoring a single PeerConnection: Echo Test

```json
{
    "type": 64,
    "timestamp": 1485872288429976,
    "session_id": 192621687169800,
    "handle_id": 7684775393694722,
    "event": {
        "plugin": "janus.plugin.echotest",
        "data": {
            "audio_active": true,
            "video_active": true,
            "bitrate": 256000
        }
    }
}
```
Monitoring a single PeerConnection: Echo Test

```json
{
    "type": 64,
    "timestamp": 1485872288429976,
    "session_id": 192621687169800,
    "handle_id": 7684775393694722,
    "event": {
        "plugin": "janus.plugin.echotest",
        "data": {
            "audio_active": true,
            "video_active": true,
            "bitrate": 256000
        }
    }
}
```
Correlating multiple PeerConnections: Video Room

- VideoRoom demo uses “opaque ID” with handles for correlation
- Again, node.js web backend stores events in updated DB

http://www.meetecho.com/blog/correlating-janus-event-handlers/
Correlating multiple PeerConnections: Video Room

```json
{
   "type": 64,
   "timestamp": 1485877400989640,
   "session_id": 6706708418803609,
   "handle_id": 5978921396405279,
   "event": {
      "plugin": "janus.plugin.videoroom",
      "data": {
         "event": "joined",
         "room": 1234,
         "id": 8361242010771549,
         "private_id": 2895980005,
         "display": "ciccio"
      }
   }
}
```
Correlating multiple PeerConnections: Video Room

```
{
  "type": 64,
  "timestamp": 1485877400989640,
  "session_id": 6706708418803609,
  "handle_id": 5978921396405279,
  "event": {
    "plugin": "janus.plugin.videoroom",
    "data": {
      "event": "joined",
      "room": 1234,
      "id": 8361242010771549,
      "private_id": 2895980005,
      "display": "ciccio"
    }
  }
}
```
Correlating multiple PeerConnections: Video Room

```json
{
    "type": 64,
    "timestamp": 1485877400989640,
    "session_id": 6706708418803609,
    "handle_id": 5978921396405279,
    "event": {
        "plugin": "janus.plugin.videoroom",
        "data": {
            "event": "joined",
            "room": 1234,
            "id": 8361242010771549,
            "private_id": 2895980005,
            "display": "ciccio"
        }
    }
}
```
Correlating multiple PeerConnections: Video Room

```json
{
    "type": 64,
    "timestamp": 1485877410046996,
    "session_id": 2273950556695883,
    "handle_id": 7656381184336435,
    "event": {
        "plugin": "janus.plugin.videoroom",
        "data": {
            "event": "joined",
            "room": 1234,
            "id": 5265065137454348,
            "private_id": 935695062,
            "display": "pippo"
        }
    }
}
```
Correlating multiple PeerConnections: Video Room

```
{
  "type": 64,
  "timestamp": 1485877410046996,
  "session_id": 2273950556695883,
  "handle_id": 7656381184336435,
  "event": {
    "plugin": "janus.plugin.videoroom",
    "data": {
      "event": "joined",
      "room": 1234,
      "id": 5265065137454348,
      "private_id": 935695062,
      "display": "pippo"
    }
  }
}
```
Correlating multiple PeerConnections: Video Room

```json
{
    "type": 64,
    "timestamp": 1485877410046996,
    "session_id": 2273950556695883,
    "handle_id": 7656381184336435,
    "event": {
        "plugin": "janus.plugin.videoroom",
        "data": {
            "event": "joined",
            "room": 1234,
            "id": 5265065137454348,
            "private_id": 935695062,
            "display": "pippo"
        }
    }
}
```
Correlating multiple PeerConnections: Video Room

```json
{
    "type": 64,
    "timestamp": 1485877410162143,
    "session_id": 2273950556695883,
    "handle_id": 3756409443147593,
    "event": {
        "plugin": "janus.plugin.videoroom",
        "data": {
            "event": "subscribing",
            "room": 1234,
            "feed": 8361242010771549,
            "private_id": 935695062
        }
    }
}
```
Correlating multiple PeerConnections: Video Room

```json
{
    "type": 64,
    "timestamp": 1485877410162143,
    "session_id": 2273950556695883,
    "handle_id": 3756409443147593,
    "event": {
        "plugin": "janus.plugin.videoroom",
        "data": {
            "event": "subscribing",
            "room": 1234,
            "feed": 8361242010771549,
            "private_id": 935695062
        }
    }
}
```
Correlating multiple PeerConnections: Video Room

```json
{
    "type": 64,
    "timestamp": 1485877410162143,
    "session_id": 2273950556695883,
    "handle_id": 3756409443147593,
    "event": {
        "plugin": "janus.plugin.videoroom",
        "data": {
            "event": "subscribing",
            "room": 1234,
            "feed": 8361242010771549,
            "private_id": 935695062
        }
    }
}
```
Correlating multiple PeerConnections: Video Room

```
{
    "type": 64,
    "timestamp": 1485877414055211,
    "session_id": 6706708418803609,
    "handle_id": 6827135960427146,
    "event": {
        "plugin": "janus.plugin.videoroom",
        "data": {
            "event": "subscribing",
            "room": 1234,
            "feed": 5265065137454348,
            "private_id": 2895980005
        }
    }
}
```
Correlating multiple PeerConnections: Video Room

{
  "type": 64,
  "timestamp": 1485877414055211,
  "session_id": 6706708418803609,
  "handle_id": 6827135960427146,
  "event": {
    "plugin": "janus.plugin.videoroom",
    "data": {
      "event": "subscribing",
      "room": 1234,
      "feed": 5265065137454348,
      "private_id": 2895980005
    }
  }
}
Correlating multiple PeerConnections: Video Room

```json
{
  "type": 64,
  "timestamp": 1485877414055211,
  "session_id": 6706708418803609,
  "handle_id": 6827135960427146,
  "event": {
    "plugin": "janus.plugin.videoroom",
    "data": {
      "event": "subscribing",
      "room": 1234,
      "feed": 5265065137454348,
      "private_id": 2895980005
    }
  }
}
```
Correlating multiple PeerConnections: Video Room

```

```
Correlating multiple PeerConnections: Video Room

```json
{
    "type": 2,
    "timestamp": 1485877399393018,
    "session_id": 6706708418803609,
    "handle_id": 5978921396405279,
    "event": {
        "name": "attached",
        "plugin": "janus.plugin.videoroom",
        "opaque_id": "videoroomtest-3REdGHcrZPMk"
    }
}
```
Correlating multiple PeerConnections: Video Room

```json
{
    "type": 2,
    "timestamp": 1485877399393018,
    "session_id": 6706708418803609,
    "handle_id": 5978921396405279,
    "event": {
        "name": "attached",
        "plugin": "janus.plugin.videoroom",
        "opaque_id": "videoroomtest-3REdGHcrZPMk"
    }
}
```

```json
{
    "type": 2,
    "timestamp": 1485877408504312,
    "session_id": 2273950556695883,
    "handle_id": 7656409443147593,
    "event": {
        "name": "attached",
        "plugin": "janus.plugin.videoroom",
        "opaque_id": "videoroomtest-vZEKMO4F5JiL"
    }
}
```

```json
{
    "type": 2,
    "timestamp": 1485877410137130,
    "session_id": 2273950556695883,
    "handle_id": 3756409443147593,
    "event": {
        "name": "attached",
        "plugin": "janus.plugin.videoroom",
        "opaque_id": "videoroomtest-vZEKMO4F5JiL"
    }
}
```

```json
{
    "type": 2,
    "timestamp": 1485877414039724,
    "session_id": 6706708418803609,
    "handle_id": 6827135960427146,
    "event": {
        "name": "attached",
        "plugin": "janus.plugin.videoroom",
        "opaque_id": "videoroomtest-3REdGHcrZPMk"
    }
}
```
Correlating multiple PeerConnections: Video Room

• **ciccio** (6706708418803609 / videoroomtest-3REdGHcrZPMk)
  • Publishing with ID **8361242010771549** (handle 5978921396405279)
  • Subscribed to ID **5265065137454348** (handle 6827135960427146)

• **pippo** (2273950556695883 / videoroomtest-vZEKMO4F5JiL)
  • Publishing with ID **5265065137454348** (handle 7656381184336435)
  • Subscribed to ID **8361242010771549** (handle 3756409443147593)

• Hey, they’re both publishing and subscribed to each other! (d’uh...)
  • PS: did you notice *private_id*? It can also help, in VideoRoom...
  • PPS: What about transports? (spoiler alert: both used HTTP)
Correlating multiple PeerConnections: Video Room

- **ciccio** (6706708418803609 / videoroomtest-3REdGHcrZPMk)
  - Publishing with ID **8361242010771549** (handle 5978921396405279)
  - Subscribed to ID **5265065137454348** (handle 6827135960427146)

- **pippo** (2273950556695883 / videoroomtest-vZEKMO4F5JiL)
  - Publishing with ID **5265065137454348** (handle 7656381184336435)
  - Subscribed to ID **8361242010771549** (handle 3756409443147593)

- Hey, they’re both publishing and subscribed to each other! (d’uh...)
  - PS: did you notice *private_id*? It can also help, in VideoRoom...
  - PPS: What about transports? (spoiler alert: both used HTTP)
Correlating multiple PeerConnections: Video Room

- **ciccio** (6706708418803609 / videoroomtest-3REdGHcrZPMk)
  - Publishing with ID **8361242010771549** (handle 5978921396405279)
  - Subscribed to ID **5265065137454348** (handle 6827135960427146)

- **pippo** (2273950556695883 / videoroomtest-vZEKMO4F5JiL)
  - Publishing with ID **5265065137454348** (handle 7656381184336435)
  - Subscribed to ID **8361242010771549** (handle 3756409443147593)

- Hey, they’re both publishing and subscribed to each other! (d’uh...)
  - PS: did you notice *private_id*? It can also help, in VideoRoom...
  - PPS: What about transports? (spoiler alert: both used HTTP)
Correlating multiple PeerConnections: Video Room

- **ciccio** (6706708418803609 / videoroomtest-3REdGHcrZPMk)
  - Publishing with ID **8361242010771549** (handle 5978921396405279)
  - Subscribed to ID **5265065137454348** (handle 6827135960427146)

- **pippo** (2273950556695883 / videoroomtest-vZEKMO4F5JiL)
  - Publishing with ID **5265065137454348** (handle 7656381184336435)
  - Subscribed to ID **8361242010771549** (handle 3756409443147593)

- Hey, they’re both publishing and subscribed to each other! (d’uh...)
  - PS: did you notice *private_id*? It can also help, in VideoRoom...
  - PPS: What about transports? (spoiler alert: both used HTTP)
Correlating multiple PeerConnections: Video Room

- **ciccio** (6706708418803609 / videoroomtest-3REdGHcrZPMk)
  - Publishing with ID **8361242010771549** (handle 5978921396405279)
  - Subscribed to ID **5265065137454348** (handle 6827135960427146)

- **pippo** (2273950556695883 / videoroomtest-vZEKMO4F5JiL)
  - Publishing with ID **5265065137454348** (handle 7656381184336435)
  - Subscribed to ID **8361242010771549** (handle 3756409443147593)

- Hey, they’re both publishing and subscribed to each other! (d’uh...)
  - PS: did you notice *private_id*? It can also help, in VideoRoom...
  - PPS: What about transports? (spoiler alert: both used HTTP)
What to do next?

- Event Handlers are a very recent addition
  - Still figuring out if existing events are enough
  - Are the correlation identifiers sufficient?
  - Experimenting with this will give us more info

- Hopefully more Event Handlers in the future
  - Homer/HEP native module on the way!
  - Other cool ideas for features/integration?

- Monitoring/troubleshooting framework
  - Student working on it (Master Degree thesis)

Help us improve this!

- Play with it, more testing is important
- Write your own applications/handlers!
What to do next?

- Event Handlers are a very recent addition
  - Still figuring out if existing events are enough
  - Are the correlation identifiers sufficient?
  - Experimenting with this will give us more info
- Hopefully more Event Handlers in the future
  - Homer/HEP native module on the way!
  - Other cool ideas for features/integration?
- Monitoring/troubleshooting framework
  - Student working on it (Master Degree thesis)

Help us improve this!

- Play with it, more testing is important
- Write your own applications/handlers!
What to do next?

- Event Handlers are a very recent addition
  - Still figuring out if existing events are enough
  - Are the correlation identifiers sufficient?
  - Experimenting with this will give us more info

- Hopefully more Event Handlers in the future
  - Homer/HEP native module on the way!
  - Other cool ideas for features/integration?

- Monitoring/troubleshooting framework
  - Student working on it (Master Degree thesis)

Help us improve this!
- Play with it, more testing is important
- Write your own applications/handlers!
What to do next?

- Event Handlers are a very recent addition
  - Still figuring out if existing events are enough
  - Are the correlation identifiers sufficient?
  - Experimenting with this will give us more info

- Hopefully more Event Handlers in the future
  - Homer/HEP native module on the way!
  - Other cool ideas for features/integration?

- Monitoring/troubleshooting framework
  - Student working on it (Master Degree thesis)

Help us improve this!

- Play with it, more testing is important
- Write your own applications/handlers!
Questions?

@elminiero
@meetecho