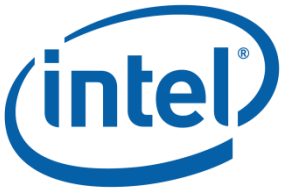


Intel® Collaboration Suite for WebRTC

(Intel® CS for WebRTC)

Conference Server User Guide

Version 2.8

September 30, 2015

Contents

List of Tables

1 Overview

1.1 Introduction

Welcome to the Conference Server User Guide for the Intel® Collaboration Suite for WebRTC (Intel® CS for WebRTC). This guide describes how to install and configure the Intel CS for WebRTC multipoint control unit (MCU). This guide also explains how to install and launch the peer server.

The Intel CS for WebRTC Conference Server provides an efficient WebRTC-based video conference service that scales a single WebRTC stream out to many endpoints. The following list briefly explains the purpose of each section in this guide:

- Section 1. Introduction and conventions used in this guide.
- Section 2. Installing and configuring the MCU.
- Section 3. Installing the MCU sample application server.
- Section 4. Installing and launching the peer server.

Installation requirements and dependencies for the MCU, sample application server, and peer server are described in their associated sections.

1.2 Conventions

This guide uses several pseudo-code and typographic conventions.

1.2.1 Pseudo-code conventions

Pseudo code is presented to describe algorithms in a more concise form. The algorithms in this document are not intended to be compiled directly. The code is presented at a level corresponding to the surrounding text.

In describing variables, a list is an unordered collection of homogeneous objects. A queue is an ordered list of homogeneous objects. Unless otherwise noted, the ordering is assumed to be first-in-first-out (FIFO).

Pseudo code is presented in a C-like format, using C conventions where appropriate.

The coding style, particularly the indentation style, is used for readability and does not necessarily comply with an implementation of Android* or JavaScript*.

1.2.2 Conventions

This document uses these conventions:

Plain text (blue)	Indicates an active link.
Bold	Identifies a processor register name or a command.
`Monospace`	Indicates computer code, example code segments, pseudo code, or a prototype code segment. These code listings typically appear in one or more separate paragraphs, though words or segments can also be embedded in a normal text paragraph.
`italic Monospace`	Indicates placeholder names for variable information (i.e., arguments) that must be supplied.
`foo@foo:~$`	A user-defined command prompt for Linux-based command lines used in the examples in this manual.

1.3 Terminology

This manual uses the following acronyms and terms:

ADT	Android Developer Toolkit
API	Application programming interface
GPU	Graphics processing unit
IDE	Integrated development environment
JS	JavaScript programming language
MCU	Multipoint control unit
MSML	Media server markup language
P2P	Peer-to-peer
QoS	Quality of service
ReST	Representational state transfer

RTC	Real-time communication
RTCP	RTP Control Protocol
RTSP	Real Time Streaming Protocol
RTP	Real Time Transport Protocol
SDK	Software development kit
SDP	Session Description Protocol
SIP	Session Initiation Protocol
XMPP	Extensible Messaging and Presence Protocol
WebRTC	Web real-time communication

1.4 For more information

For more information, visit the following Web pages:

- Intel HTML Developer Zone:

  https://software.intel.com/en-us/html5/tools

- Intel Collaboration Suite for WebRTC:

  https://software.intel.com/webrtc

  https://software.intel.com/en-us/forums/webrtc

  https://software.intel.com/zh-cn/forums/webrtc

- The Internet Engineering Task Force (IETF®) Working Group:

  http://tools.ietf.org/wg/rtcweb/

- W3C WebRTC Working Group:

  http://www.w3.org/2011/04/webrtc/

- WebRTC Open Project:

  www.webrtc.org

2 MCU Installation

2.1 Introduction

This section describes the system requirements for installing the MCU server, and the compatibility with its client.

Note: Installation requirements for the peer server are described in Section 5 of this guide.

2.2 Requirements and compatibility

Table 2-1 describes the system requirements for installing the MCU server. Table 2-2 provides compatibility information between the MCU and the client.

Table 2-1. Server requirements

Application name	OS version
MCU server	Ubuntu 14.04 LTS* 64-bit

Install the OpenH264 library to support H.264 in the MCU system when required, refer to the Install the Cisco OpenH264 Library section for more details.

Note: The OpenH264 library is not required for GPU-accelerated MCU when forward RTSP stream subscription is not used.

If you need to set up mix mode video conferences, which require GPU-accelerated media processing, you must install the following server side SDK:

- Intel® Media Server Studio 2015 for Linux*

Note: For download and installation instructions of the Intel® Media Server Studio package for Ubuntu 14.04, please contact webrtc_support@intel.com.

Table 2-2. Client compatibility

Application Name	Google Chrome* 45	Mozilla Firefox* 40	Microsoft Internet Explorer* (IE) 9, 10, 11	Intel CS for WebRTC Client SDK for Android
MCU Client	YES	YES	YES	YES
Management Console	YES	YES	N/A	N/A

2.3 Install the MCU server

This section describes the dependencies and steps for installing the MCU.

2.3.1 Dependencies

Table 2-3. Dependencies

Name	Version	Remarks
Node.js	0.10.*	Website: http://nodejs.org/
Node modules	Specified	N/A
MongoDB	2.4.9	Website: http://mongodb.org
System libraries	Latest	N/A

All dependencies, except system libraries and node, are provided with the release package.

All essential system libraries are installed when you install the MCU package using the Ubuntu's package management system, aka, apt-get or aptitude.

Regarding Node.js*, make sure it's installed in your system prior to installing the MCU. We recommend version 0.10.38. Refer to http://nodejs.org/ for the details and installation.

Before installing the MCU, make sure your login account has sys-admin privileges; i.e. the ability to execute *sudo*.

2.3.2 Configure the MCU server machine

In order for the MCU server to deliver the best performance on video conferencing, the following system configuration is recommended:

1. Add or update the maximum numbers of open files to a large enough number by adding the following two lines to /etc/security/limits.conf:

```
* hard nofile 163840
* soft nofile 163840
```

   If you only want to target these setting to specific user or group rather than all with "*", please follow the configuration rules of the /etc/security/limits.conf file.

2. Make sure pam_limits.so appears in /etc/pam.d/login as following:

```
session required pam_limits.so
```

   So that the updated limits.conf takes effect after your next login.

3. Add or update the following lines to /etc/sysctl.conf:

```
fs.file-max=200000
net.core.rmem_max=16777216
net.core.wmem_max=16777216
net.core.rmem_default=16777216
net.core.wmem_default=16777216
net.ipv4.udp_mem = 4096 87380 16777216
```

4. Now run command /sbin/sysctl -p to activate the new configuration, or just restart your MCU machine.

5. You can run command "ulimit -a" to make sure the new setting in limits.conf is correct as you set.

2.3.3 Install the MCU package

In the server machine, un-archive the package file first, and then invoke init.sh to initialize the package.

```
tar xf CS_WebRTC_Conference_Server_MCU.v<Version>.tgz
cd Release-<Version>/
bin/init.sh --deps [--hardware]
```

*Note: If you have already installed the required system libraries, you can omit the **--deps** option. What's more, if you want to run the GPU-accelerated video conferences, add **--hardware** to the init command.*

2.3.4 Install the Cisco OpenH264 Library

The default H.264 library does not provide any media logic. To enable H.264 support in non GPU-accelerated MCU system, you must install the Cisco OpenH264 library.

Follow these steps to install the library:

1. Go to the following URL and get the binary package:

http://ciscobinary.openh264.org/libopenh264-1.4.0-linux64.so.bz2.

```
curl –O http://ciscobinary.openh264.org/libopenh264-1.4.0-linux64.so.bz2
```

2. Unzip this package with following command:

```
bzip2 -d libopenh264-1.4.0-linux64.so.bz2
```

3. Copy the lib file libopenh264-1.4.0-linux64.so to Release-<Version>/lib folder, and rename it to libopenh264.so.0 to replace the existing pseudo one.

2.3.5 Use your own certificate

The default certificate (`certificate.pfx`) for the MCU is located in the Release-<Version>/cert folder.

When using HTTPS and/or secure *socket.io* connection, you should use your own certificate for each server. First you should edit `etc/woogeen_config.js` to provide the path of each certificate for each server, under the key `config.XXX.keystorePath`. E.g., `config.nuve.keystorePath` is for nuve HTTPS server. See below table for

details. If `etc/woogeen_config.js` does not exist, use `` `bin/init.sh` `` to create it.

We use PFX formatted certificates in MCU. See https://nodejs.org/api/tls.html for how to generate a self-signed certificate by `openssl` utility. We recommend using 2048-bits private key for the certificates.

After editing the configuration file, you should run `` `bin/initcert.js all` `` to input your passphrases for the certificates, which would then store them in an encrypted file. Be aware that you should have `node` binary in your shell's `$PATH` to run the JS script. `bin/initcert.js` accepts 'nuve', 'erizo', erizoController', and 'sample' (or 'all' for all) as parameters to update the passphrase for a certain certificate.

	key in configuration file	initcert.js parameter
nuve HTTPS	config.nuve.keystorePath	nuve
erizoController secured Socket.io	config.erizoController.keystorePath	erizoController
DTLS-SRTP	config.erizo.keystorePath	erizo
Sample App HTTPS	N/A	sample

2.3.6 Launch the MCU server

To launch the MCU server, follow steps below:

1. Run the following commands to start the MCU:

```
cd Release-<Version>/
bin/start-all.sh
```

2. To verify whether the server started successfully, launch your browser and connect to the MCU server at http://XXXXX:3001. Replace XXXXX with the IP address or machine name of your MCU server.

Note that the procedures in this guide use the default room in the sample.

2.3.7 Stop the MCU server

Run the following commands to stop the MCU:

```
cd Release-<Version>/
bin/stop-all.sh
```

2.3.8 Set up the MCU cluster

Follow the steps below to set up an MCU cluster which comprises several runtime nodes:

1. Make sure you have installed the MCU package on each machine before launching the cluster which has been described in section Install the MCU package.
2. Choose a primary machine.
3. Start all components as described earlier, in section Launch the MCU Server. Alternatively, you can run nuve, mcu controller and the application on the primary machine without a MCU runtime by running the following commands:

```
cd Release-<Version>/
bin/daemon.sh start nuve
bin/daemon.sh start mcu
bin/daemon.sh start app
```

4. Choose a slave machine.
5. Edit the configuration file

   `Release-<Version>/etc/woogeen_config.js` on the slave machines:

   i. Make sure the `config.rabbit.port` and `config.rabbit.host` point to the RabbitMQ server.

6. Run the following commands to launch the MCU runtime nodes on the slave machines:

```
cd Release-<Version>/
bin/daemon.sh start agent
```

7. Repeat step 4 to 6 to launch as many MCU slave machines as you need.

2.3.9 Stop the MCU cluster

To stop the MCU cluster, run the stop command on each machine including primary and slave node as described in section Stop the MCU Server.

Also, you can run the fine-grained stop command for each component you started,

1. Run the following commands to stop the sample web application, mcu controller and nuve.

```
cd Release-<Version>/
bin/daemon.sh stop app
bin/daemon.sh stop mcu
bin/daemon.sh stop nuve
```

2. Run the following commands to stop the MCU runtime node.

```
cd Release-<Version>/
bin/daemon.sh stop agent
```

2.4 Security Recommendations

Intel Corporation does not host any conference cluster/service, instead, the entire suite is provided so you can build your own video conference system and host your own server cluster.

Customers must be familiar with industry standards and best practices for deploying server clusters. Intel Corporation assumes no responsibility for loss caused from potential improper deployment and mismanagement.

The following instructions are provided only as recommendations regarding security best practices and by no means are they fully complete:

1. For the key pair access on MCU server, make sure only people with high enough privilege can have the clearance.
2. Regular system state audits or system change auto-detection. For example, MCU server system changes notification mechanism by third-party tool.
3. Establish policy of file based operation history for the tracking purpose.
4. Establish policy disallowing saving credentials for remote system access on MCU server.
5. Use a policy for account revocation when appropriate and regular password expiration.
6. Use only per user account credentials, not account groups on MCU server.
7. Automated virus scans using approved software on MCU server.
8. Establish policy designed to ensure only allowed use of the server for external connections.
9. MCU server should install only approved software and required components, and verify default features status. For example, only enable what are needed.
10. Use a firewall and close any ports not specifically used on MCU server.
11. Run a vulnerability scan regularly for checking software updates against trusted databases, and monitor publically communicated findings and patch MCU system immediately.
12. Configure appropriate connection attempt timeouts and automated IP filtering, in order to do the rejection of unauthenticated requests.
13. If all possible, use or develop an automated health monitoring component of server service availability on MCU server.
14. Use a log analyzer solution to help detect attack attempts.

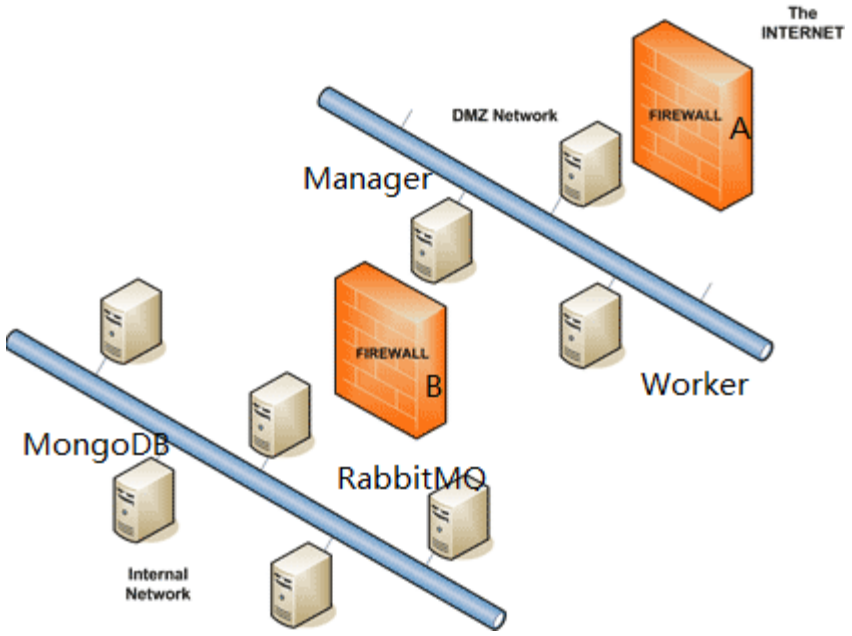
15. During server development, avoid processing of unexpected headers or query string variables, and only read those defined for the API and limit overall message size.

16. Avoid excessive error specificity and verbose details that reveal how the MCU server works.

17. During server and client development, suggest running some kind of security development lifecycle/process and conduct internal and external security testing and static code analysis with tools by trusted roles.

18. If all possible, backup all the log data to a dedicated log server with protection.

19. Deploy MCU cluster(Manager + Workers) inside Demilitarized Zone(DMZ) area, utilize external firewall A to protect them against possible attacks, e.g. DoS attack; Deploy Mongo DB and RabbitMQ behind DMZ, configure internal firewall B to make sure only cluster machines can connect to RabbitMQ server and access to MongoDB data resources.

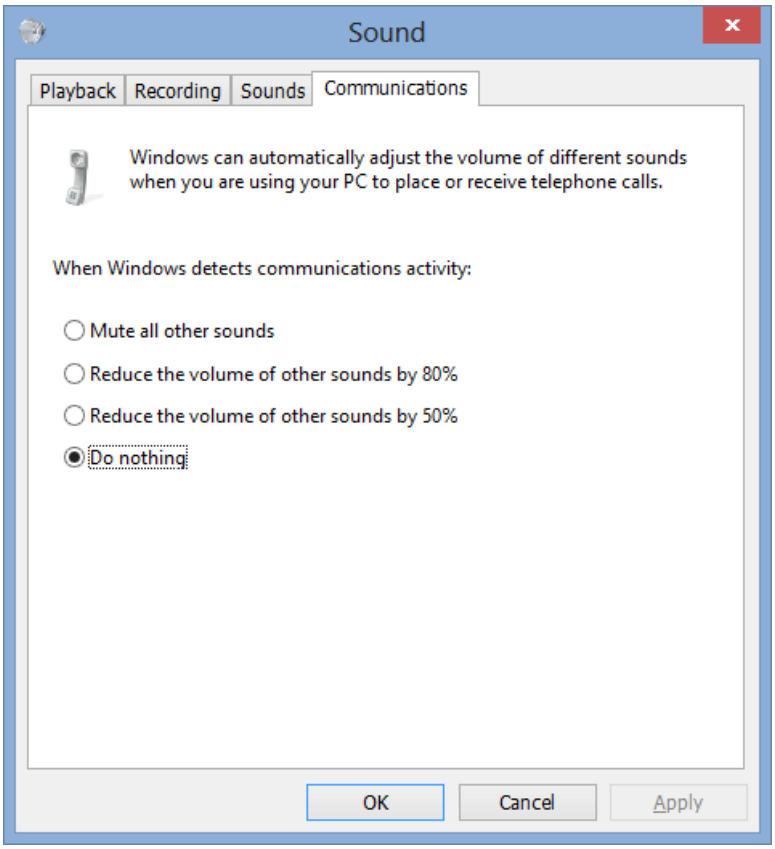


2.5 FAQ

1. Sudden low volume when connecting Chrome on Windows to MCU

Resolution:

Both the Chrome browser and Windows system itself can reduce the volume during a connection to the MCU server. To resolve this issue, disable the following Communications feature found in the Sound Settings using the Windows Control Panel.



2. Failed to start MCU server, and receive the following error message: "child_process.js:948; throw errnoException(process._errno, 'spawn'); Error: spawn EMFILE"

Resolution:

Use the proper Node.js version as outlined in the Dependencies section.

3. Failed to start MCU server, and receive the following error message: "Creating superservice in localhost/nuvedb SuperService ID: Sat Feb 7 19:10:32.456 TypeError: Cannot read property '_id' of null SuperService KEY: Sat Feb 7 19:10:32.479 TypeError: Cannot read property 'key' of null"

Resolution:

Use the proper MongoDB version as outlined in the Dependencies section.

4. Run into network port conflicts on MCU, and probably some error message like "net::ERR_CONNECTION_TIMED_OUT" or "ERROR: server connection failed: connection_error"

Resolution:

In the MCU server, the following default ports have been assigned for MCU usage: 5672 (configurable), 8080 (configurable), 3000. Make sure they are always available for MCU. Also, in order to configure the two configurable ports above to a value smaller than the 1024 limitation, use the following command to enable it:

```
sudo setcap cap_net_bind_service=+ep ~/.nvm/v0.10.38/bin/node
```

If you are still not able to bypass the 1024 port limitation, remember to put the MCU library path into /etc/ld.so.conf.d.

3 MCU Management Console Brief Guide

3.1 Introduction

The MCU Management Console is the frontend console to manage the MCU server. It is built with MCU's server-side APIs and it provides the management interface to MCU administrators.

3.2 Access

Once you have launched the MCU servers, you can then access the console via a browser at http://XXXX:3000/console/. You will be asked for your the `service-id` and `service-key` in order to access the service.

Service is the tenant management unit of the MCU server and each service holds `service-id` and `service-key` as its credential. There are 2 types of independent services: super service and normal service. Super service can manage the normal service. The MCU initialization script init.sh, as described in 2.3.3, creates the super service and 1 normal service, and prints their id and key on screen. The normal service is used by the MCU sample application server described in chapter 4.

After inputting your `service-id` and `service-key` in the dialog prompt, choose 'remember me' and click 'save changes' to save your session.

To switch to another service, click the 'profile' button on the upper-right corner to get back to the dialog prompt and perform similar procedure.

To logout from the management console, click the red 'clear cookie' button in the dialog prompt.

If you have not launched MCU severs, you should launch the `nuve` server before accessing the management console as follows:

```
cd Release-<Version>/
bin/daemon.sh start nuve
```

3.3 Source Code

The source code of the management console is in Release-<Version>/nuve/nuveAPI/public/.

3.4 Service Management

Only super service user can access service management, in the 'overview' tab to create or delete services. Note that super service cannot be deleted.

3.5 Room Management

Any service user can do room management inside the service, including create, delete or modify rooms.

Specifically for modifying rooms, user can choose room mode, room publish limit, user limit and media mixing configuration (only for hybrid mode) for its own preference. When room mode is hybrid, user can define a configuration set for media mixing: resolution, bitrate, background color, layout, etc. For VAD, set `avCoordinated` to true to enable VAD in the room.

Enabling multi streaming allows the MCU generate two or more mixed streams with different resolutions to fulfill different devices. For the layout, choose a base layout template and customize it to your preferred one, which will be combined as a whole for rendering mixed video.

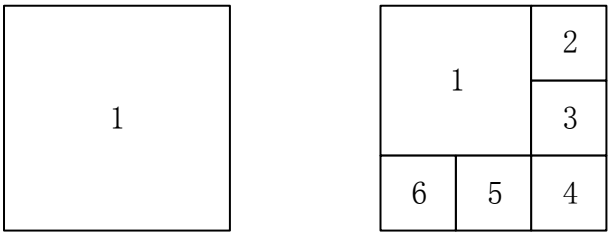
Note that if base layout is set to '`void`', the user must input customized layout for the room; otherwise, the video layout is treated as invalid. Refer to Section 3.5.1 for details of customized layout. `maxInput` indicates the maximum number of video frame inputs for the video layout definition.

3.5.1 Customized video layout

The MCU server supports the mixed video layout configuration which is compliant with RFC5707 Media Server Markup Language (MSML).

A valid customized video layout should be a JSON string which represents an array of video layout definition.

The following example shows how to express the 2 sample layouts:




```
// Video layout for the case of 1 input or 6 inputs
[
  {
    "region": [
      {
        "id": "1",
        "left": 0,
        "top": 0,
        "relativesize": 1,
      }
    ]
  },
  {
    "region": [
      {
        "id": "1",
        "left": 0,
        "top": 0,
        "relativesize": 0.667,
      },
      {
        "id": "2",
        "left": 0.667,
        "top": 0,
        "relativesize": 0.333,
      },
      {
        "id": "3",
        "left": 0.667,
        "top": 0.333,
        "relativesize": 0.333,
```

```
      },
      {
        "id": "4",
        "left": 0.667,
        "top": 0.667,
        "relativesize": 0.333,
      },
      {
        "id": "5",
        "left": 0.333,
        "top": 0.667,
        "relativesize": 0.333,
      },
      {
        "id": "6",
        "left": 0,
        "top": 0.667,
        "relativesize": 0.333,
      }
    ]
  }
]
```

Each “region” defines video panes that are used to display participant video streams.

Regions are rendered on top of the root mixed stream. “id” is the identifier for each video layout region.

The size of a region is specified relative to the size of the root mixed stream using the "relativesize" attribute.

Regions are located on the root window based on the value of the position attributes "top" and "left". These attributes define the position of the top left corner of the region as an offset from the top left corner of the root mixed stream, which is a percent of the vertical or horizontal dimension of the root mixed stream.

3.6 Runtime Configuration

Only super service user can access runtime configuration. Current management console implementation just provides the MCU cluster runtime configuration viewer.

3.7 Special Notes

- Due to server-side APIs' security consideration, API calls from a service should be synchronized and sequential. Otherwise, server may respond with 401 if the request's timestamp is earlier than the previous request. As a result, API calls of single service are from different machines while these machines' time are not calibrated, one or another machine would encounter server response with 401 since their timestamps are messed.

4 MCU Sample Application Server User Guide

4.1 Introduction

The MCU sample application server is a Web application demo that shows how to host audio/video conference services powered by the Intel CS for WebRTC MCU. The sample application server is based on MCU runtime components. Refer to Section 2 of this guide, for system requirements and launch/stop instructions.

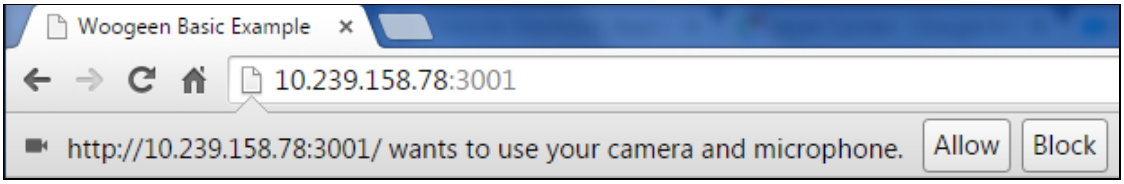
This section explains how to start a conference and then connect to a conference using different qualifiers, such as a specific video resolution.

4.2 Start a conference through the MCU sample application server

These general steps show how to start a conference:

1. Start up the MCU server components.
2. Launch your Google Chrome* browser from the client machine.
3. Connect to the MCU sample application server at: http://XXXXX:3001. Replace XXXXX with the IP address or machine name of the MCU sample application server.

   As soon as the MCU sample application server is connected successfully, the system displays a pop-up media device access confirmation:



4. Click **Allow** to start your conference.

Note: The examples in this section use the default room created by the sample application server.

4.2.1 Connect to an MCU conference with specific room

You can connect to a particular conference room. To do this, simply specify your room ID via a query string in your URL: room.

For example, connect to the MCU sample application server XXXXX with the following URL:

```
http://XXXXX:3001/?room=some_particular_room_id
```

This will direct the conference connection to the MCU room with the ID some_particular_room_id.

4.2.2 Connect to an MCU conference to subscribe mix or forward streams

Since MCU room can now produce both forward streams and mix stream at the same time, including the screen sharing stream, the client is able to subscribe specified stream(s) by a query string in your URL: mix. The default value for the key word is true.

For example, to subscribe mix stream and screen sharing stream from MCU, connect to the MCU sample application server XXXXX with the following URL:

```
http://XXXXX:3001/?mix=true
```

4.2.3 Connect to an MCU conference with screen sharing

The client can connect to the MCU conference with screen sharing stream.

To share your screen, use a query string in your URL, via the key word: screen. The default value for the key word is false.

To do this, connect to the MCU sample application server XXXXX with the following URL:

```
https://XXXXX:3004/?screen=true
```

Note: The screen sharing example in this section requires a Chrome extension named as "WebRTC Desktop Sharing Extension" from Chrome Web Store.

4.2.4 Connect to an MCU conference with a specific video resolution

In most cases, you can customize the video capture device on the client machine to produce video streams with different resolutions. To specify your local video resolution and send that resolution value to the MCU, use a query string in your URL with the key word "resolution".

The supported video resolution list includes:

'sif': (320 x 240), 'vga': (640 x 480), 'hd720p': (1280 x 720), 'hd1080p': (1920 x 1080).

For example, if you want to generate a 720P local video stream and publish that to MCU server, you can connect to the MCU sample application server XXXXX with the following URL:

```
http://XXXXX:3001/?resolution=hd720p
```

Note The specified resolution acts only as a target value. This means that the actual generated video resolution might be different depending on the hardware of your local media capture device.

4.2.5 Connect to an MCU conference with a RTSP input

The MCU conference supports external stream input from devices that support RTSP protocol, like IP Camera.

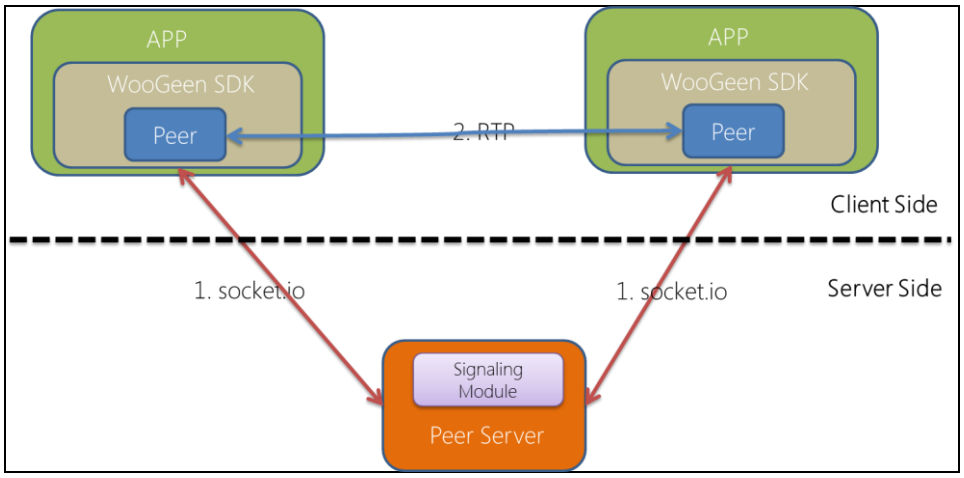
For example, connect to the MCU sample application server XXXXX with the following URL:

```
http://XXXXX:3001/?url=rtsp_stream_url
```

5 Peer Server

5.1 Introduction

The peer server is the default signaling server of the Intel CS for WebRTC. The peer server provides the ability to exchange WebRTC signaling messages over Socket.IO between different clients, as well as provides chat room management.



5.2 Installation requirements

The installation requirements for the peer server are listed in Table 5-1 and 5-2.

Table 5-1. Installation requirements

Component name	OS version
Peer server	Ubuntu 14.04 LTS, 64-bit

Note: *The peer server has been fully tested on Ubuntu14.04 LTS, 64-bit.*

Table 5-2. Dependencies

Name	Version	Remarks

Node.js	0.10.*	Website: http://nodejs.org/
Node modules	Specified	N/A

Regarding Node.js*, make sure it's installed in your system prior to installing the Peer Server. We recommend version 0.10.38. Refer to http://nodejs.org/ for installation details.

5.3 Installation

On the server machine, unpack the peer server release package, and install node modules

```
tar –zxvf CS_WebRTC_Conference_Server_Peer.v<Version>.tgz
cd PeerServer-Release-<Version>
npm install
```

The default http port is 8095, and the default secured port is 8096. Those default values can be modified in the file config.json.

5.4 Use your own certificate

If you want to use a secured socket.io connection, you should use your own certificate for the service. A default certificate is stored in cert with two files: cert.pem and key.pem. Replace these files with your own certificate and key files.

5.5 Launch the peer server

Run the following commands to launch the peer server:

```
cd PeerServer-Release-<Version>
node peerserver.js
```

5.6 Stop the peer server

Run the **kill** command directly from the terminal to stop the peer server.