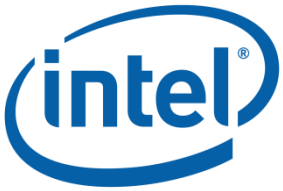


Intel® Collaboration Suite for WebRTC

(Intel® CS for WebRTC)

Conference Server User Guide

Version 2.0

December 30, 2014

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
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
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

- 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 4 of this guide.

2.2 Requirements and compatibility

Table 2-1 describes the system requirements for installing the MCU server. Table 2-2 gives an overview of MCU compatibility with the client.

Table 2-1. Server requirements

Application name	OS version
MCU server	Ubuntu 12.04 LTS* 64-bit

The H.264 support in MCU system requires the deployment of OpenH264 library. See Deploy OpenH264 Library later in this section for more details. **Note:** This doesn't need for GPU-accelerated MCU.

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 – Driver, SDK for Linux*

For installation instructions, contact webrtc_support@intel.com.

Table 2-2. Client compatibility

Application Name	Google Chrome 39*	Mozilla Firefox 33*, 34*	Intel CS for WebRTC Client Android SDK
MCU Client	YES	YES	YES

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	0.10.*	Website: http://nodejs.org/
Node modules	Specified	N/A
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 0.10.33. 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. 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
```

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

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.v2.0.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 Deploy Cisco OpenH264* Library

The default H264 library installed is a pseudo one without any media logic. To enable H264 support in non GPU-accelerated MCU system, the deployment of Cisco OpenH264* library is required; follow these steps:

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

   http://ciscobinary.openh264.org/libopenh264-1.2.0-linux64.so.bz2.

```
curl -O http://ciscobinary.openh264.org/libopenh264-1.2.0-linux64.so.bz2
```

2. Unzip this package with following command:

```
bzip2 -d libopenh264-1.2.0-linux64.so.bz2
```

3. Copy the lib file libopenh264-1.2.0-linux64.so to Release-<Version>/lib folder, and rename it to libopenh264.so to replace the existing pseudo one.

2.3.5 Customized video layout

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

The default video layout style is "fluid", which is set by the configuration "config.erizo.videolayout.type" in woogeen_config.js. To use the customized video layout, set the configuration item to "custom".

At the same time, default size of the root mixed stream is provided as "vga" by "config.erizo.videolayout.defaultrootsize". The tested configuration candidates are as following:

'sif': (320 x 240), 'vga': (640 x 480), 'hd_720p': (1280 x 720), 'hd_1080p': (1920 x 1080).

The default background color of the root mixed stream is "black" by the configuration "config.erizo.videolayout.defaultbackgroundcolor". The current candidate values are: black, white.

The layout configuration file locates at Release-<Version>/etc with the name custom_video_layout.js. The following example shows the details:

```
// Video layout for the case of no more than 2 inputs
{
    "maxinput": 2,
    "root": {
        "backgroundcolor": "black"
    }
    "region": [{
       "id": "1",
       "left": 0,
       "top": 0.25,
       "relativesize": 0.5
```

```
    }, {
        "id": "2",
        "left": 0.5,
        "top": 0.25,
        "relativesize": 0.5
    }]
}
```

"maxinput" indicates the maximum number of video frame inputs for this video layout definition.

"root" section defines the mixed video stream attributes. Currently the background color configuration is supported.

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.

2.3.6 Use your own certificate

When using HTTPS and/or secure *socket.io* connection, you should use your own certificate for each server.

The default certificates (cert.pem and key.pem) for the MCU worker are located in the Release-<Version>/cert folder. The default certificates (cert.pem and key.pem) for the MCU sample are located in the sample/cert folder.

Replace those default certificates with your own certificate and key files.

2.3.7 Launch the MCU server

To launch the MCU server, follow these two simple steps:

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.

2.3.8 Stop the MCU server

Run the following commands to stop the MCU:

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

2.3.9 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 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 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.

   ii. `config.nuve.superserviceID` and `config.nuve.superserviceKey` should be equivalent to the values on primary machine.
6. Run the following commands to launch the MCU runtime node on the slave machines:

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

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

2.3.10 Stop the MCU cluster

You can 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 following commands to stop the nuve node and sample web application.

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

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

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

3 MCU Sample Application Server User Guide

3.1 Introduction

The MCU sample application server is a demo Web application 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.

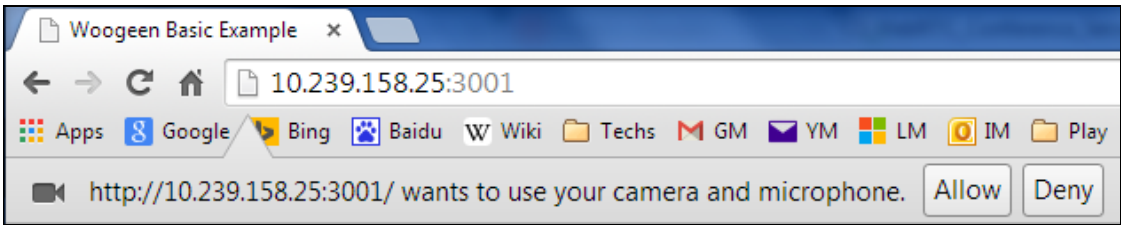
This section explains how to start a conference and then connect to a conference using different qualifiers, such as a specific video resolution.

3.2 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.*

3.2.1 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.

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

Since MCU room now can produce both forward streams and mix stream at the same time, including the screen sharing stream, the client is now 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
```

3.2.3 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.

3.2.4 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:

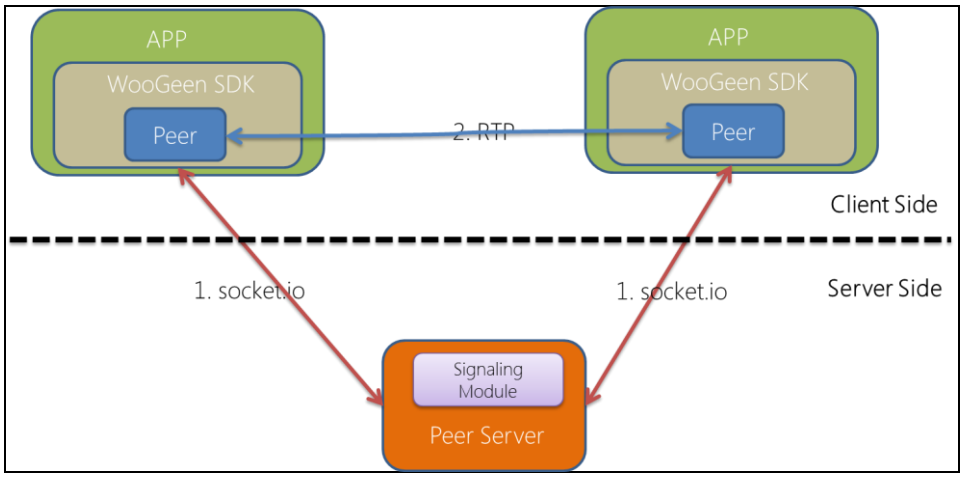
```
http://XXXXX:3001/?resolution=hd720p
```

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 Peer Server

4.1 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.



4.2 Installation requirement

The installation requirement for the peer server are listed in Table 4-1.

Table 4-1. Installation requirement

Component name	OS version
Peer server	Ubuntu 12.04 LTS, 64bit

Note: *The peer server is tested fully only on Ubuntu12.04 LTS, 64 bit. To ensure proper operation, make sure your environment is the same as Ubuntu12.04 LTS, 64 bit.*

4.3 Installation

On the server machine, unpack the peer server release package. This might require sys-admin privileges.

```
tar –zxvf CS_WebRTC_Conference_Server_Peer.v2.0.tgz
```

The default http port is 8095, and the default secured port is 8096. If you want to change either port, modify config.json.

4.4 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 bin/cert with two files: cert.pem and key.pem. Replace these files with your own certificate and key files.

4.5 Launch the peer server

Run the following commands to launch the peer server:

```
cd PeerServer-Release-xxxxx
bin/peerserver &
```

4.6 Stop the peer server

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