Web Real-Time Communications Deployment Guide

Web Real-Time Communications 8.5.2

12/29/2021

Table of Contents

Deployment Guide

This deployment guide can be used to install the Genesys WebRTC Service in your environment. It includes the following information:

- New In This Release—An overview of new features and improvements included with each release of the Genesys WebRTC Service.
- Product Overview—An overview of the Genesys WebRTC Service.
- Deploying WebRTC Service—Step-by-step guide to installing and deploying the Genesys WebRTC Service on your computer.
- Deployment Models—A description of how the Genesys WebRTC Service can be deployed in a production environment, taking into account a variety of typical deployment types.
- Log Events Reference—Descriptions of the log events generated by the Genesys WebRTC Gateway.
- Configuration Options Reference—Descriptions of the configuration options available for Genesys WebRTC applications.
- Hardware Sizing Information—Useful hardware sizing and performance information.

Next Steps

After you have successfully installed the Genesys WebRTC Service, you might want to read the Genesys WebRTC Service Developer's Guide to find general guidelines and programming advice when working with your Genesys WebRTC Service deployment.

New in this Release

Check out the new features that have been added in the latest releases of the Genesys WebRTC Service.

New in Release 8.5.2

Release 8.5.208.16 of the WebRTC Gateway

- A boolean option, `[rsmp] web-disable-sdes`, has been added with a default value of `true`, which disables SDES-SRTP in initial SDP offers to the web client. With a value of `true`, the RTP profile in an initial offer SDP from the gateway to a web client will use `"UDP/TLS/RTP/SAVPF"` instead of `"RTP/SAVPF"`, adhering to the WebRTC standard. Even with a value of `true`, incoming offers with SDES-SRTP will still be accepted, though the use of SDES-SRTP is obsolete and discouraged. (MWA-575)
- The Genesys WebRTC Gateway now has G.722 audio codec enabled by default. Note that G.722 codec is given higher priority than G.711 codecs by the browsers as well as the WebRTC gateway, therefore G.722 may be chosen over G.711 during call establishment. (MWA-98)

Release 8.5.206.25 of the WebRTC Gateway

- The following configuration option has been added, which helps work around some renegotiation issues with Firefox. When an older version of Firefox without bundle support is used (version 37 and lower), set this option to false. (MWA-547)
  web-offer-bundle-only
  Section: rsmp
  Default Value: true
  Valid Values: true, false

Release 8.5.201.95 of the WebRTC Gateway and **Release 8.5.210.03 of the WebRTC JavaScript API**

- The Genesys WebRTC Service now supports adding video to an audio-only call
- The Genesys WebRTC Gateway now supports remote CTI control by providing the SIP extensions event package known as the BroadSoft SIP extensions

Release 8.5.201.30 of the WebRTC Gateway

- Support for third-party call control (3pcc) basic functions and two-step procedures

Release 8.5.200.95 of the WebRTC Gateway and **Release 8.5.200.07 of the WebRTC JavaScript API**

This first release of the Genesys WebRTC Service includes:

- Communications:
    - Supports two-way audio-only calls
    - Supports two-way audio and video calls
    - Supports transcoding on supported media types in both directions
    - Supports transitions between audio-only and video/audio sessions (within browser limitations)
    - Supports incoming calls from either the web or the SIP side
    - Supports sending context data from a web client to the SIP Server as attached data when a call is established
    - Supports sending mid-call user data either to SIP Server as mapped data, or to the remote peer
    - Supports call transfer with Genesys SIP Server
    - Supports sending DTMF tones as telephone-events
- Web browser support:
    - Google Chrome
    - Google Chrome for Android
    - Mozilla Firefox
    - Mozilla Firefox for Android
    - Opera (Desktop and Mobile)
- Supports anonymous access from the web (such as in click-to-dial scenarios)
- Flexible deployment on:
    - A dedicated server
    - A shared server with other Genesys components
    - Multiple load-balanced instances for scalability
- Security:
    - HTTPS, SIP TLS, and secured connection to Configuration Server
    - Web-side encryption of RTP traffic according to IETF recommendations (SDES-SRTP and DTLS-SRTP)
    - Can be configured to use SIP-side SRTP when outside the trusted area
    - Supports Client Side Port Definition (CSPD), allowing for customization of the:
        - SIP-side RTP port
        - Web-side RTP port
    - Transport address and port for a client-side connection
    - Use of multiple NICs, to separate public and private interfaces when both are used (DMZ deployment)
    - Firewall traversal with support of ICE
- Codecs:
    - G.711 A-law and µ-Law audio format

    - G.729 audio on SIP/RTP side
    - H.264 profiles up to and including Profile 3.1 on SIP/RTP side
    - VP8 video
    - Real-Time media transcoding whenever required
    - RTCP AVPF (partially supported)
    - telephone-events
- Support of HTTP over IPv4 and IPv6 interfaces in the same instance
- Integration with Genesys Management Framework
- SNMP for alarms, traps and MIBs
- Platforms:
    - Windows Server 2008 32- and 64-bit
    - RedHat Linux 6.0 64-bit
    - VMWare ESXi 5
- The WebRTC Gateway is built with OpenSSL library version 1.0.1g. This version of OpenSSL is not affected by the TLS heartbeat read overrun issue.
- The WebRTC Gateway supports receiving INFO data in www-form-urlencoded format from the browser in the middle of a call, and forwarding it to the SIP Server using the SIP INFO method.
- The WebRTC Gateway accepts + as a valid first character of a DN.
- The WebRTC Gateway includes support for Cross-Origin Resource Sharing (CORS).
- JavaScript libraries for integration with web applications, communications, and attached data transfer
- The WebRTC JSAPI provides a configuration parameter to specify the time (in milliseconds) to wait for an answer from the peer side after making an offer. If that timeout expires and the offer is still not answered, then the JSAPI sends the `onPeerNoanswer` event to the client application.
  The minimum valid value for this timeout is `18000` (18 seconds) and the default value is `60000` (60 seconds).
- The WebRTC JSAPI provides a mechanism for mid-session data transfer for the following scenarios:
    - Between two peers
    - From a peer to the SIP Server as mapped user data

Product Overview

Genesys WebRTC Service provides web browsers with Real-Time Communications (RTC) capabilities via simple JavaScript APIs. This means you can write rich, real-time multimedia applications—such as video chat—on the web, without requiring plug-ins, downloads, or installs. Genesys WebRTC Service is able to do this by using the Web Real-Time Communications technologies that have been developed by the World Wide Web Consortium and the Internet Engineering Task Force.

The key component of Genesys WebRTC Service is the Genesys WebRTC Gateway. The Gateway converts WebRTC communications to SIP, allowing for the seamless integration of WebRTC-capable devices into a SIP-based infrastructure—in particular, an architecture that uses Genesys SIP Server. From that perspective, the WebRTC Gateway plays the role of a Media Gateway / Session Border Controller.

All calls that have been initiated or received by a browser, including browser-to-browser calls, are bridged through the Gateway. This is done to ensure that every call is handled by SIP Server as a SIP call, so that all of the core Genesys features—including routing, treatments, and IVR—can be provided by SIP Server.

For more information about the Genesys WebRTC Service, see the Architecture page in the WebRTC Developer's Guide.

Codec Support

The Genesys WebRTC Gateway has the following codec support:

Codec	Codec Type	Supported Interfaces/ Protocols
G.711	Audio	SIP, WebRTC
G.722	Audio	SIP, WebRTC
G.729	Audio	SIP
telephone-event	Audio/DTMF	SIP, WebRTC
VP8	Video	SIP, WebRTC
H264	Video	SIP

Ports and Protocols

Important
All of the following default port values can be changed using configuration options.

Source Protocol(s)	Destination	Default Port(s)	Transport
Browser	WebRTC Gateway	8086 *	TCP/TLS (HTTP or HTTPS)
Browser	TURN Server	443	TCP, UDP (TURN)
TURN Server or Browser	WebRTC Gateway	36000-36999	UDP (DTLS, SRTP)
WebRTC Gateway	TURN Server	49152-65535	UDP (DTLS, SRTP)
SIP Server	WebRTC Gateway	5066	TCP, UDP (SIP, SIPS)
WebRTC Gateway	SIP Server	5060	TCP, UDP (SIP, SIPS)
WebRTC Gateway	SIP Endpoint	9000-9999	UDP (RTP, SRTP)
SIP Endpoint	WebRTC Gateway	9000-9999	UDP (RTP, SRTP)
MCP	WebRTC Gateway	9000-9999	UDP (RTP, SRTP)
WebRTC Gateway	MCP	20000-45000	UDP (RTP, SRTP)
WebRTC Gateway	Config Server	2020	TCP
***** Port 443 is recommended for HTTPS			

Deploying WebRTC Service

Installing and Deploying the Genesys WebRTC Service

Objective	Related procedures and actions
1. Prepare your deployment.	Review the tasks on the Preparing your Deployment page.
2. Install a test version of the Genesys WebRTC Service.	Review the steps outlined in the task table on the Installation Procedures page. **Note:** Genesys recommends that you first deploy the Genesys WebRTC Service in a lab environment. After successfully deploying the Genesys WebRTC Service, create and test a WebRTC application before you switch to production.
3. Deploy to production.	Once the Genesys WebRTC Service is working correctly in your lab environment, you can deploy it to your production environment by following the steps outlined in the task table on the Installation Procedures page. See Deployment Models for more information on your deployment options.

Environment

The following lists the recommended system configurations for the WebRTC gateway in lab and production environments.

- CPU: Intel(R) Xeon(R) CPU E5-2680 v2 @ 2.80GHz or greater
- In Lab/POC:
  - 1 vCPU and 4 GB RAM recommended
  - Disk space allocation minimum 35GB
- In production:
  - 1-2 vCPU(s) and 4-8 GB RAM recommended
  - Disk space allocation minimum 60GB

Check System Performance

Sysbench is a common benchmarking utility that you can use to test your server performance on Linux. Since it is one of the most common benchmarking utilities, you can easily find and compare performance results online.

Installing Sysbench

You can install Sysbench via yum. To install Sysbench 0.4.12 on Redhat/CentOS 6.x, you can `wget` the latest `remi` and `epel` repositories, install them, and then install Sysbench.

```
wget http://dl.fedoraproject.org/pub/epel/6/x86_64/epel-release-6-8.noarch.rpm
wget http://rpms.famillecollet.com/enterprise/remi-release-6.rpm
rpm -Uvh remi-release-6*.rpm epel-release-6*.rpm

yum install sysbench
```

Sysbench CPU Test

To run a single-threaded CPU test, run the following command. Typically with a value of `20000`, a single-threaded test can take anywhere from 20 seconds to a minute or more. The shorter the time, the better the result, so faster CPUs will take less time to run the test.

```
sysbench --test=cpu --cpu-max-prime=20000 --num-threads=1 run
```

Important

For the Genesys WebRTC Gateway deployment, we recommend that the total amount of time for the single-threaded CPU test be within one minute.

You can use the script below to run multi-threaded Sysbench CPU tests. Multi-threaded CPUs will not show their true performance unless you run the test with the correct amount of threads. If you have an 8-core CPU, you must run an 8-thread CPU test to get the true performance number.

```
for each in 1 2 4 8; do sysbench --test=cpu --cpu-max-prime=20000 --num-threads=$each run;
done
```

Sysbench Memory Test

To test the RAM performance, run the following memory operations speed test.

```
sysbench --test=memory --memory-block-size=1M --memory-total-size=100G run
```

Important

For the Genesys WebRTC Gateway deployment, we recommend that the performance result be 4000 ops/sec and higher.

Sysbench Disk IO Test

To prepare test files, run the following command. Typically, prepare a test file with a size that is about two times more than the amount of RAM the server has. For example, if the server has 4GB RAM, use a 8GB file size.

```
sysbench --test=fileio --file-total-size=8G prepare
```

Then, follow by a basic Sysbench random read/write test:

```
sysbench --test=fileio --file-total-size=8G --file-test-mode=rndrw --init-rng=on --max-
time=180 --max-requests=0 run
```

Finally, do cleanup:

```
sysbench --test=fileio --file-total-size=8G cleanup
```

Important

For the Genesys WebRTC Gateway deployment, we recommend that the performance result be 1000 iops/sec and above.

Prepare your Deployment

Purpose
To list the items to consider before you install and set up the Genesys WebRTC Service.

Complete the following tasks before installing the Genesys WebRTC Service:

- Review the Interoperability Considerations and ensure that the required components have been installed.
- If you don't have one already, set up a Web Server to host and serve the Genesys WebRTC JavaScript API/SDK files and the demo application over HTTPS and/or HTTP.

Interoperability Considerations

The Genesys WebRTC Gateway can be used with several types of component, including Genesys servers and many of the leading industry-standard web browsers. This topic provides information on which of these are supported.

Interoperable Components

- Genesys SIP Server
- Genesys Universal Routing Server (URS)
- Genesys Orchestration Server (ORS)
- Genesys Voice Platform (GVP) Media Control Platform (MCP)
- Google Chrome Browser (desktop and mobile)
- Mozilla Firefox Browser (desktop and mobile)
- Opera Web Browser (desktop and mobile)
- Genesys SIP Endpoint SDK
- Genesys Workspace Desktop Edition (WDE) SIP Endpoint
- Genesys Workspace Web Edition (WWE)
- Third-party SIP soft-phones (Bria and LinPhone)
- Genesys Epi Phone

Supported Genesys Product Versions

The following table indicates which versions of the listed Genesys components are supported for use with the WebRTC Gateway.

Component	8.0	8.1	8.5
Management Framework	Yes	Yes. Preferred Configuration Server version is 8.1.3 or above.	—
Genesys Administrator	No	Yes	—
SIP Server	No	Yes. Preferred SIP Server version is 8.1.1 or above. **Note**: *not verified with SIP cluster*.	—
GVP	No	Yes. Preferred GVP version is 8.1.7.	Yes. Works with GVP 8.5.0. However, GVP 8.5.1 or above is the

Component	8.0	8.1	8.5
			preferred version.
URS	No	Yes. Preferred URS version is 8.1.4 or above.	—
ORS	—	Yes. Preferred ORS version is 8.1.4 or above.	—
Workspace Desktop Edition SIP Endpoint	—	—	Yes. Preferred version is 8.5.1 or above.
Workspace Web Edition	—	—	Yes. Preferred version is 8.5.1 or above.

Supported Endpoints

Important

It is recommended that the latest stable version of a supported browser is used. However, in some cases, it may be required to use an older version due to some bug or incompatible change in the latest version. See the WebRTC Product Alerts and Release Notes for known issues with specific browser versions.

Web Endpoint	Version
Chrome	54 and above
Firefox	49 and above
Opera	41 and above

SIP Endpoint	Version
Bria	4.1 and above
Genesys SIP Endpoint SDK	8.5.2 and above

Supported Call Scenarios

Peer-to-Peer

Peer-to-peer sessions can be either symmetric or asymmetric:

- **Symmetric Session**—Each peer offers the same media types—such as audio to audio or video to video.
- **Asymmetric Session**—Different media types—such as audio to video or video to audio—are offered by

the peers.

Browser-to-Browser	• Symmetric session • Asymmetric session
Browser-to-SIP (or vice versa)	• Symmetric session • Asymmetric session
Browser-to-MCP	• Video calls • Audio calls (both sides must be audio-only) • Announcement • VoiceXML dialog • ASR input support • DTMF support (Chrome only—Firefox does not currently support sending DTMF tones)

With Genesys Routing

Direct Agent Routing	• Call to Route Point • No treatment play • Agent is always available
Agent Routing with Treatment	• Call to Route Point • Agent is not Ready, MCP plays Video (audio-only is also supported) • Agent becomes Ready • Treatment stops • Call routed to Agent
Routed to MCP for Transfer	• Call to Route Point • Routed to MCP • MCP plays Video (audio-only is also supported) • VoiceXML transfer • Bridge

	• Media redirect

IVR

- MCP is involved
- VoiceXML dialog
  - Conditional logic possible thru Attached Data
  - For example, profile information can be passed as data and the VXML app can play the appropriate message or transfer to the appropriate agent based on that information
  - ASR input supported
  - DTMF input supported—**for Chrome only**
- VoiceXML transfer
  - Bridge
  - Media redirect

Other

Call Recording	• Recording audio only • Call from browser to Route Point • Routed to MCP • MCP plays music and records
Hold/Retrieve	• Call routed to Agent (SIP) • Agent puts the caller on Hold • Agent later Retrieves the call • Works fine with SIP Endpoint SDK and Bria • Music and Video on Hold scenario works as well (with MCP) **Note:** Firefox does not support renegotiation. For more information, see Genesys WebRTC Developer's Guide.
Call upgrade/downgrade scenario	• Update the call in progress by starting or stopping the sending of local audio or video • Limited testing done and a demo added with new media session every time a call upgrade or downgrade happens, as Firefox does not

	support renegotiation

Third-Party Call Control (3PCC)

For third-party call control (3pcc), set the configuration options as shown in the following table.

`TServer` section of your SIP Server application object	• `sip-hold-rfc3264`—Enable SIP Hold per RFC 3264. Must be set to `true`.
`TServer` section of your SIP Switch WebRTC Extension DN objects	• `dual-dialog-enabled`—Enable Dual Dialog Mode. Must be set to `false`. • `refer-enabled`—Enable REFER support. Must be set to `false`. • `sip-cti-control`—Enable SIP CTI Control. Must have an empty value when Broadsoft extensions are not used for WebRTC. However, when Broadsoft extensions are used, set this parameter to `talk,hold`.

Reporting Considerations

Genesys WebRTC interactions are treated internally as SIP interactions with SIP Server. Because of this, WebRTC calls will normally be reported as SIP calls when Genesys Info Mart or Genesys Interactive Insights are configured in the deployment.

In order to allow customers to report on these interactions independently, the WebRTC Gateway includes the `reporting-service-type` configuration option in the `[rsmp]` section. The default value for this parameter is `WebRTC`. Once this parameter has been set, the WebRTC Gateway adds a custom SIP header called `X-genesys-service_type`—which contains this value—to each SIP call it makes.

On the SIP Server side, the following two parameters must be set in the `[TServer]` section:

- `userdata-map-all-calls=true`
- `userdata-map-trans-prefix=X-genesys-`

When these parameters have been set, if a SIP call comes to SIP Server from the WebRTC Gateway, it passes the `service_type` obtained from the request to ICON as user data. The standard Genesys Info Mart configuration includes integration of the `service_type` Key-Value pair as Genesys Info Mart's `interaction_descriptor.service_type` field.

SIP Endpoint Configuration

Component	Specific/Preferred Configuration
Bria	• Disable H264 • Enable VP8 **Note:** Bria has issues with H.264 (support for packetization mode, setting H.264 profile level, video quality, and so on), so VP8 is preferred for use with Bria.
Genesys SIP Endpoint SDK	• Supports H.264 (from version 8.5.1), and VP8. Set VP8 priority to 1 (highest). • `auto_accept_video = 1`
Genesys Workspace SIP Endpoint	Supports audio and video

SIP Endpoint Known Issues

- There are compatibility issues with various types of SIP Endpoint, as discussed in the section on SIP Endpoint configuration

Media Codec Considerations

For deployment models that support agents on SIP endpoints, additional measures must be taken to ensure that a WebRTC-enabled browser can interoperate with the SIP endpoint's media codecs. If the SIP endpoint does not support audio or video codecs required by WebRTC browsers, the deployment will require real-time media transcoding. The WebRTC Service has an automatic media transcoder to ensure the media path can be bridged between the browser and a SIP endpoint. It is important to note that media transcoding, especially video transcoding, can be a process-intensive task, so it is recommended that the SIP endpoint be configured with codecs that are compatible with the codecs that have been implemented on the WebRTC side.

Please refer to, Codec Support for a list of codecs that are supported by the Genesys WebRTC Service.

For calls established between two WebRTC browsers, the codecs will always be compatible since the WebRTC specification requires compatible codecs.

Genesys Media Server may also be involved as part of the conversation with a WebRTC browser, in order to serve media such as music or video on hold. Genesys Media Server may serve media files that use any of its supported media formats. If necessary, the media files will be automatically transcoded to a codec that is compatible with the WebRTC browser. As a way to optimize processing, Media Server supports automatic caching of transcoded media, which means that a media file only gets transcoded once. Subsequent calls to the same Media Server instance can reuse the cache without needing to transcode again.

Third-Party Software

The following components are mandatory for running the Genesys WebRTC Service:

Platform	Component Name
Windows	Microsoft Visual Studio 2005 SP1 ATL Security Update
Windows	Microsoft Visual Studio 2008 SP1 ATL Security Update
Linux	GLib 2.14+

Installation Procedures

Purpose
List the Installation tasks for the Genesys WebRTC Service.

Note: *As mentioned below, you must create a WebRTC Gateway application object before you can run and test the Genesys WebRTC Service.*

Tasks	Procedures
1. Create a WebRTC Gateway application object in Genesys Administrator or Genesys Administrator Extensions	• If you are using version 8.1.3 or later of Configuration Server, import the `WebRTC_Gateway_850.apd` template. • If you are using an earlier version of Configuration Server, import the `WebRTC_Gateway_850_generic.apd` template.
2. Install the Genesys WebRTC Service	Complete one of the following procedures, according to your operating system: • Installing the Genesys WebRTC Service on Windows • Installing the Genesys WebRTC Service on Linux
3. Configure the necessary options in the rsmp section of your WebRTC Gateway application object	Review the Security Considerations and Platform configuration sections and set the required options.

Validate the Installation

- After the WebRTC gateway has been installed, configured and started, you can run a basic gateway health check by going to the following URL in a browser:

  ```
  http[s]://<WebRTC_Gateway_FQDN>:<HTTP_Port>/ping
  ```

  You should get a response with the text "up".
  If you want to test this locally from the gateway machine on Linux, you can run:

  ```
  wget --no-check-certificate -q -O - http[s]://localhost:<HTTP_Port>/ping
  ```

- After the WebRTC JavaScript API (JSAPI) has also been installed on a Web server, you can test the WebRTC operation using the demo1 application that comes with the JSAPI, by making a loop-back call or a call with another demo instance or a SIP endpoint. For running demo1, go to the following URL:

  ```
  https://<WebServer_FQDN>/webrtc/demo/demo1/audio_video.html
  ```

  Here it is assumed that the JSAPI files are installed in the Web server under sub-directory "webrtc".
  Follow the steps in the page using the information there to make a test call and perform some operations with the call, such as hold and resume.

The audio_video.html may need to be modified to set the gateway URI (http[s]://<address>:<port>) and possibly other JSAPI configuration parameters, especially STUN or TURN server information, although most of these parameters can be input during STEP 2 of the demo. You may also want to add the configured DN values to be used in STEP 4 (for registering with the SIP Server configured in the gateway) and STEP 5 (for calling a DN, which another endpoint is registered with) in the HTML page itself for easy use.

You would need to have SSL certificates installed for the gateway (as described in the Platform Configuration section) as well as the Web server; or for testing purposes, you could manually allow the security exceptions in the browser for the Web server and the gateway. The caller app/demo1 can be registered with the gateway anonymously using the ID "Anon", and can make a loop-back call by calling the ID "self" to test the signaling and media connections with the gateway. If you want to make a call to another endpoint, however, you will need to have a SIP Server configured in the gateway, and have the caller endpoint register with the SIP Server using an extension DN, which the caller can use to make the call via the gateway.

Create a WebRTC Gateway application object

Before you proceed with the installation of Genesys WebRTC Service on Windows or Linux machines, you must create a WebRTC Gateway application object in Genesys Administrator (GA) or Genesys Administrator Extension (GAX).

The following procedure helps you to create a WebRTC Gateway application object in Genesys Administrator:

1. Open Genesys Administrator and navigate to **PROVISIONING** > **Environment** > **Application Templates**.
   You will see a list of application templates that were already imported into GA.
2. Click **New** in the **Application Templates** task bar.
   The **New Application Template** screen appears.
3. On the **Configuration** tab, in the **Type** drop-down list, select **WebRTC Gateway**.
4. Click **Import Metadata**.
   In the pop-up window that appears, you will see a list of metadata templates that were already imported into GA.
5. If you do not see any metadata template, click **Add** to import one.
6. On the **Browse** window, navigate to the **CD** > **Templates** folder and import the **WebRTC_Gateway_852.xml** file.
   The metadata information is now added to the Application Template.
7. In the **Name** field, type a name for the Application Template and click **Save & Close**.
8. Click **Applications** on the left tree-view pane.
   You will see a list of applications that were already created in GA.
9. Click **New** on the **Applications** task bar.
   The **New Application** screen appears.
10. On the **Configuration** tab, in the **Application Template** field, browse and select the newly created WebRTC Gateway application template.
    The application template information is now added to the application.
11. In the **Name** field, type a name for the application.
12. In the **Server Info** section, specify the following values:
    1. In the **Host** field, browse and select the host where you would like to install the WebRTC Gateway application.
    2. In the **Listening Ports** field, click **Add** and configure the port.
       The **Port Info** pop-up window appears.
    3. On the **Port Info** pop-up window > **General** tab, specify the desired port value in the **Port** field and click **Ok**.
    4. In the **Working Directory** field, type a period (.)

    5. In the **Command Line** field, type a period (.)

13. Click **Save and Close**.

The WebRTC Gateway application object is now created and available for installation. You can now proceed with further installation procedures.

Next Steps

- Install the WebRTC package on Windows
- Install the WebRTC package on Linux

Install on Windows

Installing the Genesys WebRTC Service on Windows

Purpose
To install the deployment files for the Genesys WebRTC Service on a Windows-based web server.

> Important
>
> Before running the installer, make sure that you have created an application object for WebRTC Gateway in Genesys Administrator or Genesys Administrator Extension.

Note: *After running each of the Windows installers, inspect the directory tree of your system to make sure that the files have been installed in the location that you intended.*

Installing the WebRTC Gateway

Start

1. On your desktop, open the the Genesys WebRTC Service CD/DVD or the Genesys WebRTC Service IP.
2. Open the `Web Real-Time Communications\Windows` folder and double-click the `Setup.exe` file.
   You might be asked to reboot your system to delete or rename certain system files before the Installation Wizard runs.
   The Genesys Installation Wizard launches and the `Welcome` panel is displayed.
3. On the `Welcome` panel, do one of the following:
   - Click `Next` to begin the installation procedure.
   - Click `Cancel` to exit the Genesys Installation Wizard.
4. Follow the prompts to complete the installation.
   - Click `Cancel` to exit the Genesys Installation Wizard.
5. Click `Finish` to exit the Genesys Installation Wizard.

End

Installing the WebRTC JavaScript API

The Genesys WebRTC JavaScript API library is shipped in a file called `grtc.js`. There is also a directory of sample applications that demonstrate how to use the API. The following instructions explain how to locate these files.

Start

1. On your desktop, open the the Genesys WebRTC Service CD/DVD or the Genesys WebRTC Service IP.
2. Navigate to the `Web Real-Time CommunicationsJSAPI/Windows/src/jsapi/classes/` directory, which contains the Genesys WebRTC JavaScript API library in a file called `grtc.js`.
3. Include the `grtc.js` file in your web application in order to use the WebRTC JavaScript API.
4. Navigate to the `Web Real-Time CommunicationsJSAPI/Windows/` directory.
5. Copy the demo directory to your hard drive.
6. Examine the README file in the demo directory for information on how to run the demo apps.

End

Install on Linux

Installing the Genesys WebRTC Service on Linux

Purpose
To install the deployment files for the Genesys WebRTC Service on a Linux-based server.

Important

- Before running the installer on Red Hat, make sure that the Red Hat Compatibility Packs have been installed.
- Before running the installer, make sure that you have created an application object for WebRTC Gateway in Genesys Administrator or Genesys Administrator Extension.

Note: *After running one of the installers, inspect the directory tree of your system to make sure that the files have been installed in the location that you intended.*

Installing the WebRTC Gateway

Start

1. Open a terminal in the Genesys WebRTC Service CD/DVD or the Genesys WebRTC Service IP.
2. Navigate to the `Web Real-Time Communications/Linux` directory and run the `install.sh` file. The Genesys Installation starts.
3. Follow the prompts to complete the installation.
4. If the installation is successful, the console displays the following message:

   ```
   Installation of Genesys WebRTC Service Gateway, version 8.5.00x.yy has completed
   successfully.
   ```
5. In order to use HTTPS or SIPS for the Linux version of the WebRTC Gateway, install the Genesys Security Pack version 8.5.1.

End

Installing the WebRTC JavaScript API

The Genesys WebRTC JavaScript API library is shipped in a file called `grtc.js`. There is also a directory of sample applications that demonstrate how to use the API. The following instructions explain how to locate these files.

Start

1. Open a terminal in the Genesys WebRTC Service CD/DVD or the Genesys WebRTC Service IP.
2. Navigate to the `Web Real-Time CommunicationsJSAPI/Linux/src/jsapi/classes/` directory, which contains the Genesys WebRTC JavaScript API library in a file called `grtc.js`.
3. Include the `grtc.js` file in your web application in order to use the WebRTC JavaScript API.
4. Navigate to the `Web Real-Time CommunicationsJSAPI/Linux/` directory.
5. Copy the demo directory to your hard drive.
6. Examine the README file in the demo directory for information on how to run the demo apps.

End

Deploying WebRTC with MCP

When using the Genesys WebRTC Service and the Genesys Voice Platform Media Control Platform (MCP) on Red Hat Enterprise Linux (RHEL), Genesys recommends that MCP be run on a server that is separate from the server running the WebRTC Service, for two reasons:

- MCP performance on RHEL 5.x is considerably better than on RHEL 6.x. However, the WebRTC Gateway only works on RHEL 6.x.
- MCP consumes the majority of the resources of the server it is hosted on, even under RHEL 5.x. Because of this, Genesys recommends—regardless of whether WebRTC is running in your environment—that MCP be placed on its own, separate machine.

Security Considerations

Genesys is committed to offering products and solutions with a high level of security. Opening a real-time communication channel on the web requires secure transmission, and the Genesys WebRTC Service does not compromise the security level of the enterprise network. The following topics discuss the issues that you may wish to take into account when deploying the Genesys WebRTC Service, depending on your enterprise network architecture.

Secure Transports

With WebRTC, the browser is initiating both an HTTP connection and a media session that connects it with another WebRTC peer. Since the description of the media session is signaled over HTTP, this can be protected with the use of HTTPS, which is not enabled by default, however. If an HTTP proxy and/or a load balancer (LB) that are deployed in the same network are used, then HTTPS may not be needed between the proxy/LB and the gateway.

The media session is secured through the use of Secure Real-time Transport (SRTP), meaning all media are encrypted before being sent over the wire. The security keys for encrypting the media are exchanged in a secure manner through either the signal path (SDP security descriptions) or the media path itself (DTLS). The Genesys WebRTC Service enables DTLS-SRTP by default to ensure that all media paths with the browser are secure. Note that DTLS-SRTP is mandated by the WebRTC standard for media.

Please see Platform Configuration section on how to configure these options.

Securing the Host

The best practice for securing a host that is deployed on the web is to place the host behind a firewall. The Genesys WebRTC Service requires the firewall to open up the HTTPS port for inbound traffic with the browser, and a range of SRTP ports for inbound and outbound traffic with the TURN server (or directly with the browser, if that is acceptable, and TURN server is not to be used).

The WebRTC Service also needs to establish SIP calls to Genesys SIP Server. Genesys SIP Server and agent endpoints are typically deployed within the enterprise firewall, so in this case, the WebRTC Service only needs to access the SIP port and a range of RTP ports on the enterprise network.

The WebRTC Service also connects to the Genesys Management Framework, which is deployed within the enterprise network. This means that the Service requires access to Local Control Agent and Configuration Server.

Other ports may be allowed by the firewall for other administration purposes. It is recommended that they only be accessible either from within the enterprise network or through a secure communication channel from the Internet, such as ssh.

Please refer to Ports and Protocols for information on the port usage.

Security for the Web Application

The user does not directly access the WebRTC Service to place calls, as it is the responsibility of the web application to deliver a user interface to the browser for the user. The WebRTC Service includes a JavaScript API that provides a set of HTTP API methods by means of which the web application can access the WebRTC Service separately from the web server that hosts the web application.

Cross Origin Resource Sharing (CORS)

Since the browser Same Origin Policy prevents a web page from making an `XMLHttpRequest` to another domain, the WebRTC Service supports Cross Origin Resource Sharing (CORS) to allow the web application to interact with the WebRTC Service across domains.

For a simple request—one that uses either `GET` or `POST` and whose body is `text/plain`—the request is sent with an extra header called `Origin`. The `Origin` header contains the origin URI (scheme, domain name or address, and port, as per RFC 6454) of the requesting page so that the server can easily determine whether or not it should serve a response. An example `Origin` header might look like this:

```
Origin: http://www.genesys.com:8080
```

The WebRTC Service provides a configuration option (`domain-whitelist`) to specify the allowed list of domains, and any CORS request specifying an `Origin` that is not allowed will result in an error. However, if the list of allowed domains is set to empty, the WebRTC Service then allows interactions with all sites with CORS by setting the `Access-Control-Allow-Origin` HTTP header in the response to the value specified in the `Origin` header value from the original request.

If the server decides that the request should be allowed, it either sends an `Access-Control-Allow-Origin` header echoing back the same origin that was sent or `'*'` if it is a public resource.

For example:

```
Access-Control-Allow-Origin: http://www.genesys.com:8080
```

If this header is missing, or the value of this header does not match the value of `Origin` header, then the browser disallows the request. If all is well, then the browser processes the response.

Tools and Services

Fail2ban

To block IP addresses that repeatedly have failed login attempts, Genesys recommends installing and using Fail2ban on Linux hosts.

The Fail2ban RPMs are available through EPEL repository, which may need to be set up first. You can then install Fail2ban using YUM:

```
sudo yum –y install fail2ban
```

The default install will ban IP addresses after three failed attempts for 600 seconds.

Telnet and FTP

Telnet and FTP services have known security issues. Genesys recommends disabling these services on the hosts.

Xinetd

Genesys recommends disabling Xinetd on Linux by running the following command:

```
sudo chkconfig xinetd off
```

Platform Configuration

The following topics discuss the important configuration options that you may need to set in the rsmp section of your WebRTC gateway application object, before start using the WebRTC gateway. For further information on these options and other possible configuration options, please refer to Configuration Options Reference.

Configure the following deployment specific options first:

- `sip-proxy`—The address and port of your Genesys SIP-Server, for example: `<host-name>:<port>`.
- `sip-register`—The list of DNs configured in SIP Server for use by the WebRTC Gateway, for example: `8100-8199,9000,9002,9800-9820`. A client can use any one of these DNs to register itself with the SIP Server via the gateway for receiving calls. A 'callee' DN that is used by the client to make an outbound call to does not need to be in this list. In this case, the user will be registered by the gateway as anonymous.
- `stun-server`—The address and port of the STUN server to be used, for example: `<host-name>:<port>`. This option is not mandatory, but Genesys recommends that you configure it before using the WebRTC Gateway. This option is required when the WebRTC gateway is in a private network and it cannot find out its public address by itself.
- `http-port`—The HTTP or HTTPS port of the WebRTC Gateway used for signaling with the browser. The default value is `8086`, which may need to be changed depending on the network configuration. It's is recommended that port 443 is used for HTTPS.

Configuring HTTPS

For enabling HTTPS on the WebRTC gateway, you need an SSL certificate from a public Certificate Authority (CA), and need to set the following gateway configuration parameters to appropriate values: `enable-https`, `https-cert`, `https-cert-key` and `https-trusted-ca` (the last two are for Linux only). Note that the port used for HTTPS is still specified using `http-port`, and HTTP is not supported when HTTPS is enabled.

Important

Chrome by default requires that HTTPS is used for enabling media device access on the host machine.

If you have a Load Balancer (LB) or proxy fronting the WebRTC Gateway, you need an SSL certificate installed for the LB or proxy. In this case, HTTP, instead of HTTPS, could be used between the LB/proxy and the gateway, provided these two are in a secure network. You may still need to use HTTPS between the browser and the LB/proxy.

Enabling DTLS-SRTP

DTLS-SRTP (RFC 5763) is enabled on the web side by default. However, if you need to re-enable it for some reason, simply set the `web-enable-dtls` configuration option to `true`.

When `web-enable-dtls` is enabled, it will be signalled in an SDP offer sent by the gateway using the fingerprint attributes, though there will also be crypto attributes in SDP for SDES-SRTP (RFC 4568) support. When an offer or answer comes in with only crypto attributes, then SDES-SRTP will still be supported. When `web-enable-dtls` is set to `false`, only SDES-SRTP will be supported.

Important

Only Chromium based browsers support SDES-SRTP for historical reasons, and the WebRTC standard demands that DTLS-SRTP is used.

Enabling SRTP for the SIP Side

SRTP is not enabled on the SIP side by default. If the SIP agent and the WebRTC Gateway are in the same local/private network, you may not need to enable it. If needed, you can enable it by setting the gateway configuration parameter `sip-srtp-mode` to an appropriate vlaue (`optional` or `strict`).

Configuring Secure SIP (SIPS)

Use of SIPS is not enabled by default. If the SIP agent and the WebRTC gateway are in the same local/private network, you may not need to enable it. To enable it, you need an SSL certificate from a public Certificate Authority (CA), and need to set the following gateway configuration parameters to appropriate values: `sip-tls-port`, `sip-tls-cert`, `sip-tls-cert-key` and `sip-tls-trusted-ca` (the last two are for Linux only).

Configuring a Specific IP Address for SIP or RTP

When a different IP address than what is automatically detected by the gateway needs to be used for SIP or RTP, you can use the following options to configure this address. This is useful for AWS instances or multi-homed hosts. `sip-address`: This value will be used in SIP Via and/or Contact headers, that indicate the address to which SIP messages should be sent. `rtp-address`: This value will be used for SDP c= line, that indicates the address to which RTP should be sent.

Configuring RTP Port Ranges

You do not normally need to change the port ranges used for RTP. If required, however, you could configure these using options `sip-rtp-min-port`, `sip-rtp-max-port`, `web-rtp-min-port`, and `web-rtp-max-port`.

Configuring Media Codecs

Although you should not need to, if required, you could disable or change the priority order of the supported media codecs using options `codecs`, `sip-added-codecs`, `web-added-codecs`, `sip-disallowed-codecs`, and `web-disallowed-codecs`. Note that the codec list in an SDP created by the the gateway for a client, including the order of codecs, is always based on the offer or answer SDP from the other client, and only the codecs in `sip-added-codecs` or `web-added-codecs` (depending on whether the SDP is to be sent to the SIP or Web client) are appended to this codec list; not all the codecs from the codecs option are included in the SDP.

Disabling Anonymous Sign-in

The option `allow-anonymous-user` can be used to disable anonymous sign-ins from the WebRTC client by setting it to 'false'. When disabled, only valid DNs on the SIP server can be used to sign-in. This option has the default value of 'true'.

Configuring Cross Origin Resource Sharing (CORS)

The WebRTC gateway supports CORS to restrict HTTP[S] access from arbitrary domains. A list of allowed domains can be configured using the option domain-whitelist. See Security Considerations for more information.

SIP Server High Availability

When DNS SRV is used for SIP Server High Availability (HA), you must configure the WebRTC Gateway:

- Set `rsmp.sip-proxy` with the IP addresses or the FQDNs of the SIP Servers, separated by a comma.
- Set `rsmp.sip-proxy-srv` to the SIP Server SRV address.

The Gateway will start with the first sip-proxy address. When a request to the current address times-out, the Gateway will switch to the other address and then back. When a request arrives from SIP Server with its SRV address, it will be translated to the current active SIP Server address before sending a response.

TURN Server Deployment

The Genesys WebRTC Service has been tested with the coTURN TURN server, which is a free, high-performance open-source TURN and STUN server implementation. Genesys currently recommends v4.5.0.3 of the coTURN TURN server; however, more recent versions may exist.

If a TURN server is already deployed in your network or you are planning to deploy a TURN server other than coTURN, you can skip the following steps.

Host Requirements

- RHEL/CentOS 6.6+
- At least two vCPUs, 4GB Memory. SSD may be used, but is not mandatory.
- The most important factor is the networking performance. Make sure that your network has:
    - High packet per second (PPS) performance
    - Low network jitter (<= 30ms)
    - Low latencies (<= 150ms)

For example, if you are deploying the TURN server in an AWS instance, then Enhanced Networking is only available on instances launched with HVM AMIs. Preferably, use C3/C4/R3 instances. For details on AWS EC2 instance types, check here.

Installation and Configuration

Download and install the latest version of TURN server from this link.

You must install the TURN server on a separate server. If you want to install the TURN server in the same machine as the WebRTC Gateway, make sure there is no port conflict.

You must have super-user access to install the TURN server.

1. Download the TURN server installation package from the above link, and unzip the file into a temporary folder.
2. Run the **install.sh** file to install the TURN server:

```
# chmod +x install.sh
 # ./install.sh
```

3. The installation creates a `turnserver` user, and the TURN server runs from this account. To enable the TURN server to bind lower ports (<1024), set the following:

```
# setcap 'cap_net_bind_service=+ep' /usr/bin/turnserver
```

4. Update the parameters in the following configuration file:

   ```
   /etc/turnserver/turnserver.conf
   ```

   - **listening-port**: set to 443
   - **external-ip**: set to the `<public-ip>[/private-ip]` for this VM; for example `54.145.183.203/10.157.156.67`. The assumption is that the TURN server will be behind NAT in most of the deployments.
   - Uncomment **fingerprint**
   - Uncomment **lt-cred-mech**
   - **user**: set to `<user>:<password>`; for example `genesys:genesys`. The same user/password you need to configure for your web application.
   - **realm**: simply set it to `yourdomain.com` (for example `ec2-54-145-183-203.compute-1.amazonaws.com`)

5. Start the process:

   ```
   # service turnserver start
   ```

6. Check if the TURN server is running:

   ```
            # ps -elf | grep turnserver
   5 S 498      24882     1  0  80   0 - 138025 ep_pol 23:44 ?       00:00:00 /usr/bin/
   turnserver -o -c /etc/turnserver/turnserver.conf
   ```

7. For additional details on the TURN server parameters and configuration, you can check here.

Verifying your TURN Server

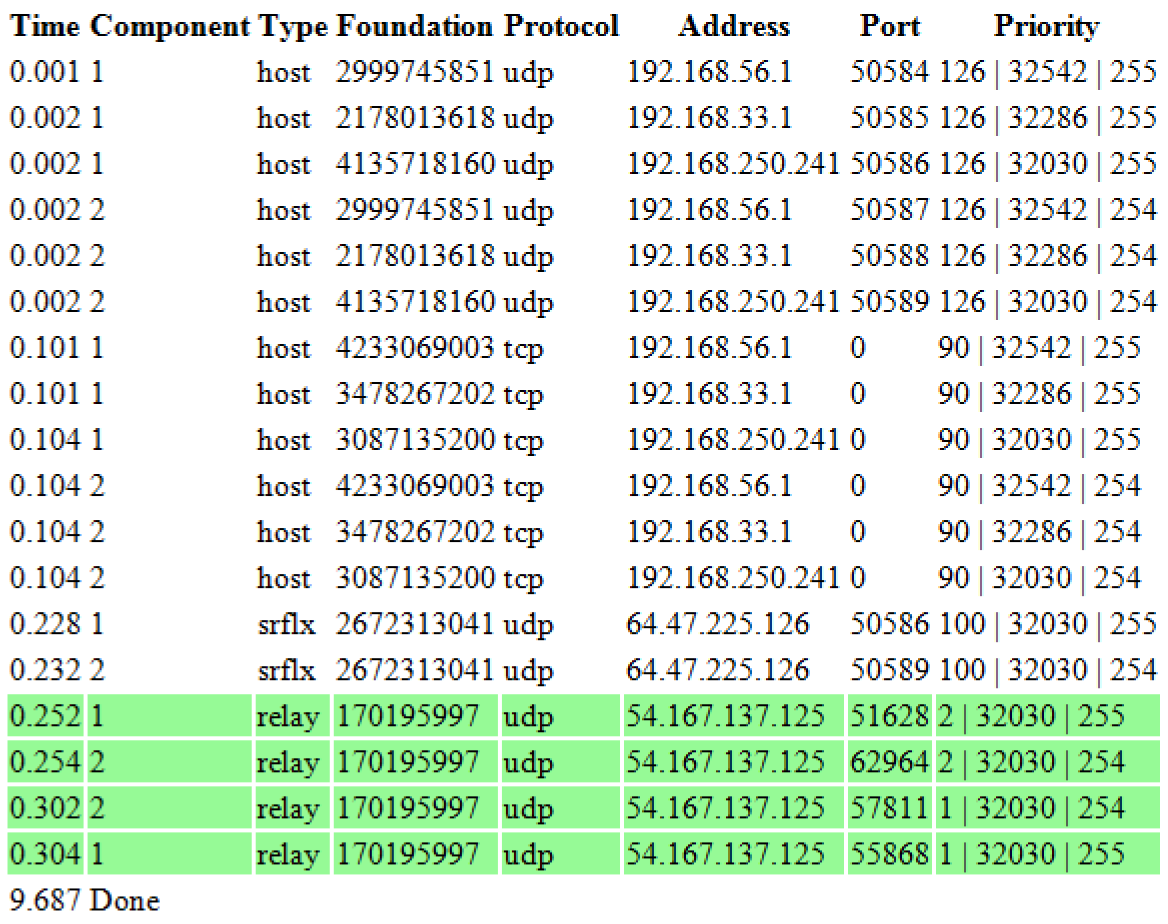
To verify whether your TURN server is running properly and is accessible, go to this demo page.

This page tests the trickle ICE functionality in a WebRTC implementation. It creates a `PeerConnection` with the specified STUN/TURN servers, and then starts candidate gathering for a session with a single audio stream. As candidates are gathered, they are displayed in a text box, along with an indication when candidate gathering is complete.

Select and remove the default STUN server that is populated in this page, and then add your TURN server(s) instead. For example, if you have installed the Genesys TURN server in `turn.example.net` and set password as `genesys` for `genesys` user, add both:

- `turn:turn.example.net:443` (and user/password as `genesys/genesys`)
- `turn:turn.example.net:443?transport=tcp` (and user/password as `genesys/genesys`)

Then press `Gather candidates` and specifically check for `relay` candidates in the result (see example below). If no relay candidates are generated, either your TURN server is not running properly or is not accessible from your network.



Time	Component	Type	Foundation	Protocol	Address	Port	Priority
0.001	1	host	2999745851	udp	192.168.56.1	50584	126 | 32542 | 255
0.002	1	host	2178013618	udp	192.168.33.1	50585	126 | 32286 | 255
0.002	1	host	4135718160	udp	192.168.250.241	50586	126 | 32030 | 255
0.002	2	host	2999745851	udp	192.168.56.1	50587	126 | 32542 | 254
0.002	2	host	2178013618	udp	192.168.33.1	50588	126 | 32286 | 254
0.002	2	host	4135718160	udp	192.168.250.241	50589	126 | 32030 | 254
0.101	1	host	4233069003	tcp	192.168.56.1	0	90 | 32542 | 255
0.101	1	host	3478267202	tcp	192.168.33.1	0	90 | 32286 | 255
0.104	1	host	3087135200	tcp	192.168.250.241	0	90 | 32030 | 255
0.104	2	host	4233069003	tcp	192.168.56.1	0	90 | 32542 | 254
0.104	2	host	3478267202	tcp	192.168.33.1	0	90 | 32286 | 254
0.104	2	host	3087135200	tcp	192.168.250.241	0	90 | 32030 | 254
0.228	1	srflx	2672313041	udp	64.47.225.126	50586	100 | 32030 | 255
0.232	2	srflx	2672313041	udp	64.47.225.126	50589	100 | 32030 | 254
0.252	1	relay	170195997	udp	54.167.137.125	51628	2 | 32030 | 255
0.254	2	relay	170195997	udp	54.167.137.125	62964	2 | 32030 | 254
0.302	2	relay	170195997	udp	54.167.137.125	57811	1 | 32030 | 254
0.304	1	relay	170195997	udp	54.167.137.125	55868	1 | 32030 | 255
9.687	Done						

Also, the type of candidates released to the application can be controlled via the `IceTransports` constraint.

Deployment Models

Before deploying the Genesys WebRTC Service to production, you should install it in a lab environment and create and test a sample application in that environment. The Genesys WebRTC Service Developer's Guide has important details about how to create your test application.

Once you have successfully created and tested your application, see the following topics for an in-depth discussion of how the Genesys WebRTC Service can be deployed in a production environment.

- NAT and ICE
    - Network Address Traversal
    - Firewalls and Restrictive NAT
- Deployment Types
    - Private Cloud
- Scalability, Availability, Failover

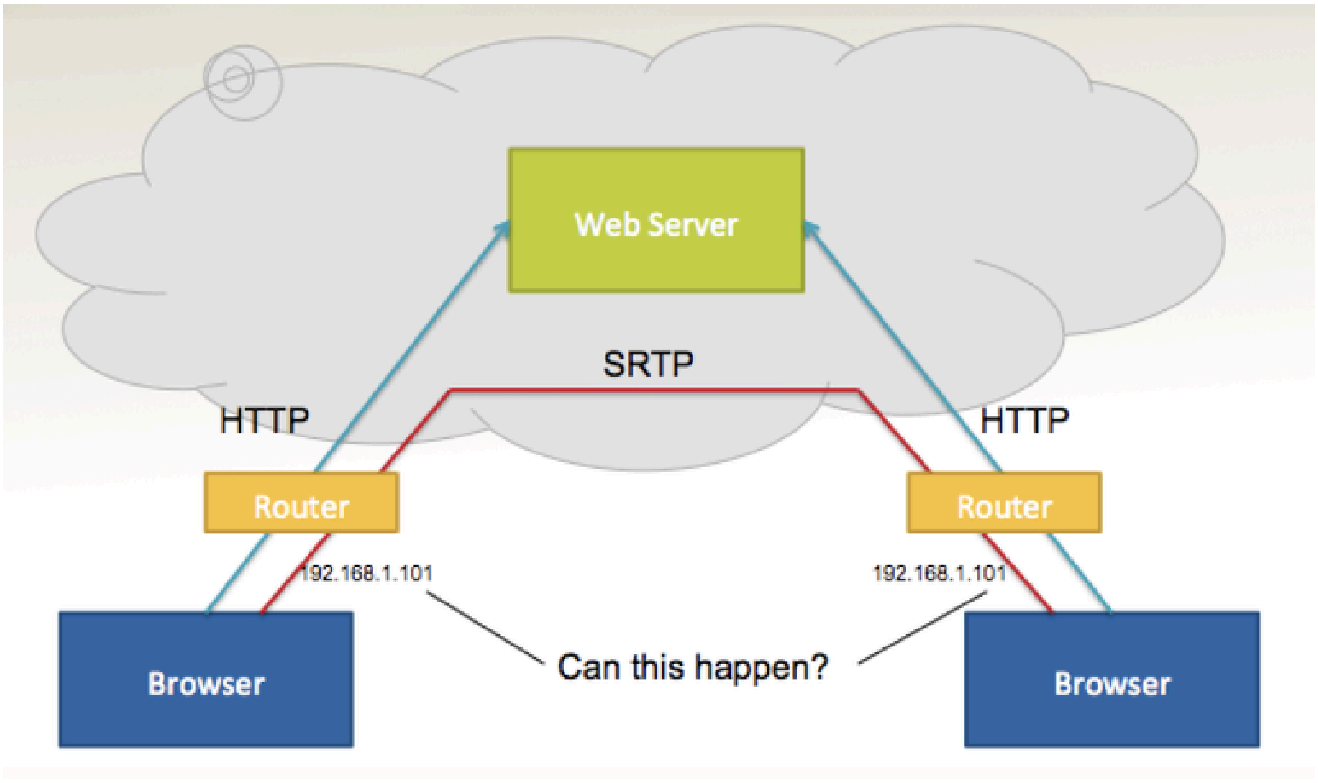
NAT and ICE

Since WebRTC sends real-time media over a UDP stream, it includes a mechanism that ensures media can be sent between the browser and the WebRTC gateway in both directions. The following topics provide some networking basics to highlight the concept behind Network Address Traversal (NAT), how this affects the Genesys deployment, and how it is addressed by Interactive Connectivity Establishment (ICE) that is employed by the WebRTC standard.

Network Address Traversal

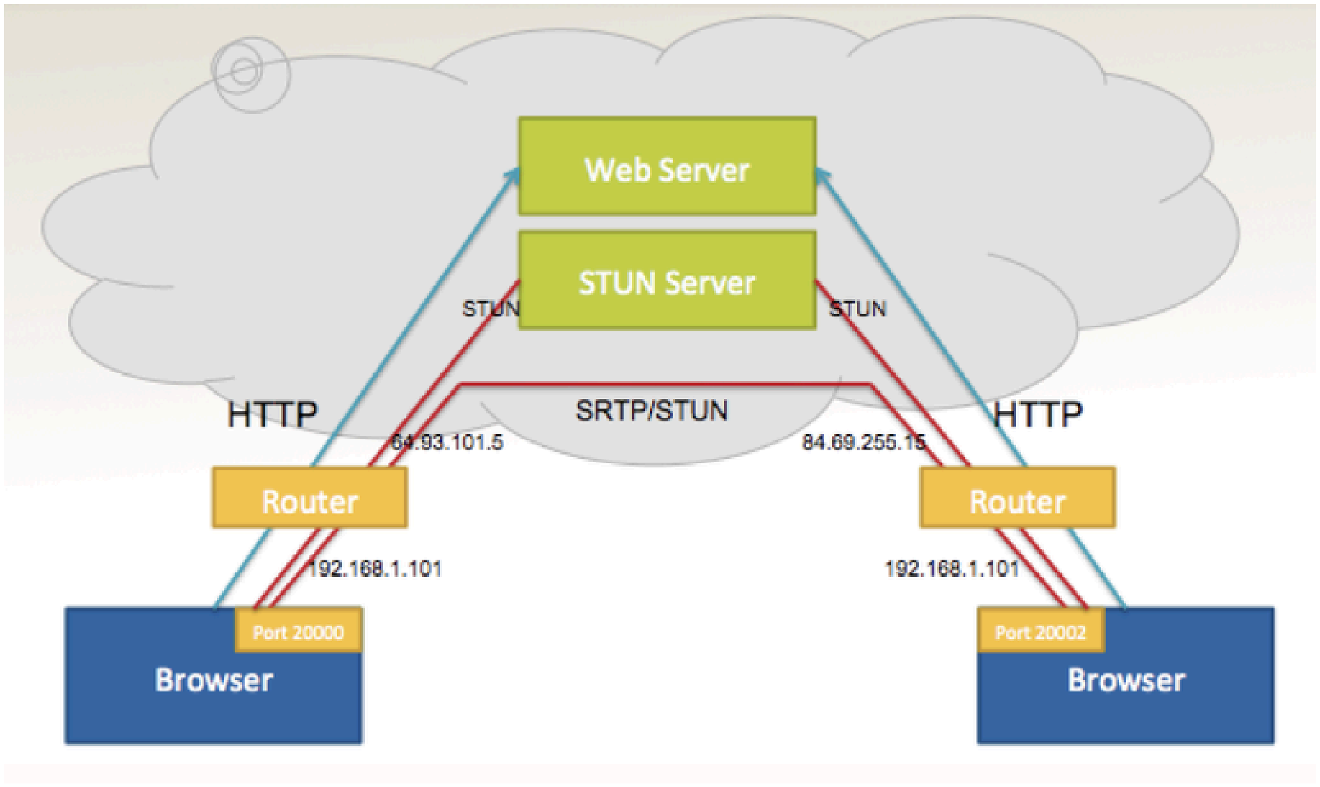
When a device, such as a computer, is working behind a home router, it acquires an IP address from that router. In order to serve multiple devices at home, the router assigns its own IP address in a private subnet but allows all devices behind the router to reach out to the Internet. In a typical case, the home computer gets an internal address, such as `192.168.1.101`. A browser accessing normal web pages will use TCP to connect from the browser client to a server, so home routers work very well in these scenarios.

With WebRTC, however, the browser needs to send a media stream directly to another browser over UDP, as shown in the following diagram:



Since the browsers on both devices will only see the IP addresses acquired from the home routers, they will not be able to connect directly to one another, since these addresses are private. The Interactive Connectivity Establishment (ICE) protocol (RFC5245) was designed to resolve this problem, which does not normally exist in a traditional VoIP environment, since all of the device endpoints are deployed within the office network and hence no address translation is required.

The following is a simplified view of how ICE deals with NAT:



ICE uses the Session Traversal Utilities for NAT (STUN) protocol as a way to discover the server-reflexive address seen by a public server. Once a browser discovers all the addresses seen by public servers, it can use ICE to negotiate the connection with the peer. ICE parameters can be carried in SDP or in some other external signaling mechanism, and JSEP allows ICE negotiation independent of SDP negotiation.

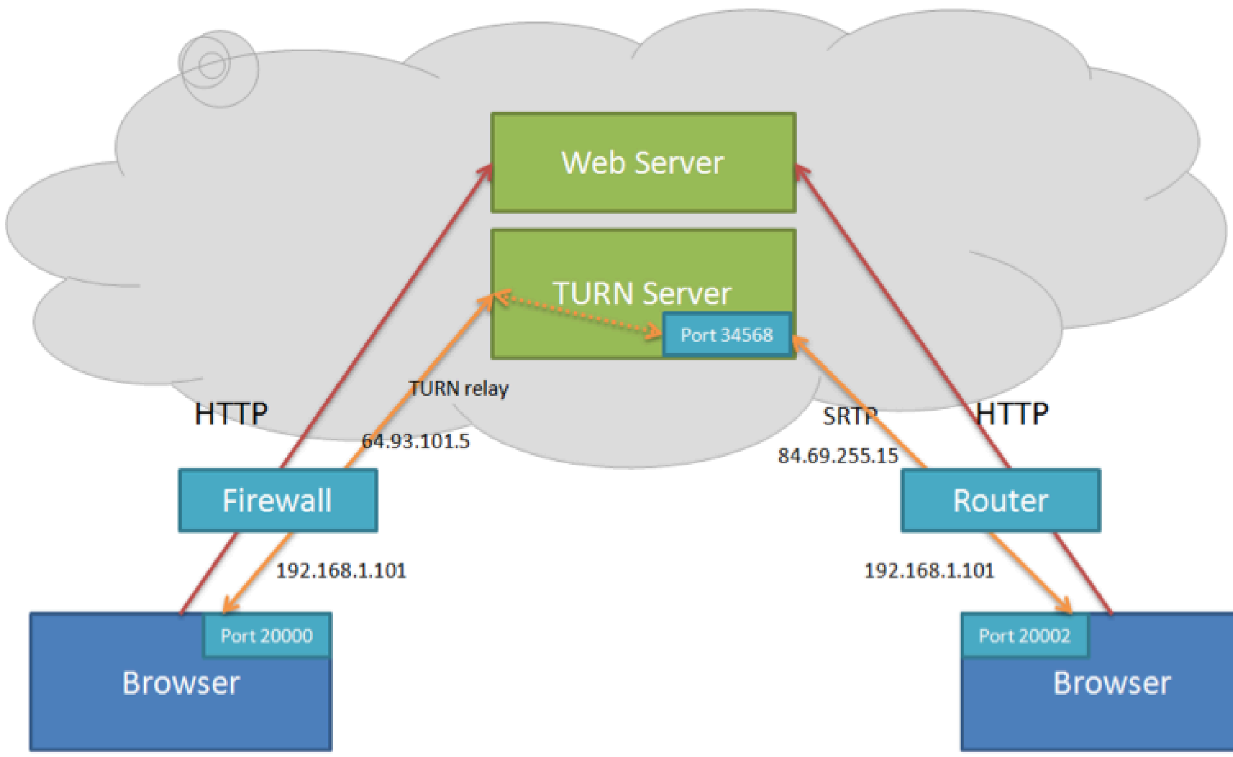
Firewalls and Restrictive NAT

In addition to dealing with NAT (Network Address Traversal), enterprise firewalls often implement restrictive access control as a means of protecting the enterprise network. A typical enterprise firewall might block all unknown ports from both inbound and outbound traffic. This means only a handful of known commonly used protocols, such as HTTP/HTTPS, ftp, and ssh, will be allowed to access the enterprise network.

Also, some routers may have a restrictive NAT translation policy that enforces address- and port-dependent mapping. This means the NAT binding can only be enabled for a specific address and port. When both endpoints are behind the same type of NAT, then both sides will be unable to find the actual binding outside of NAT.

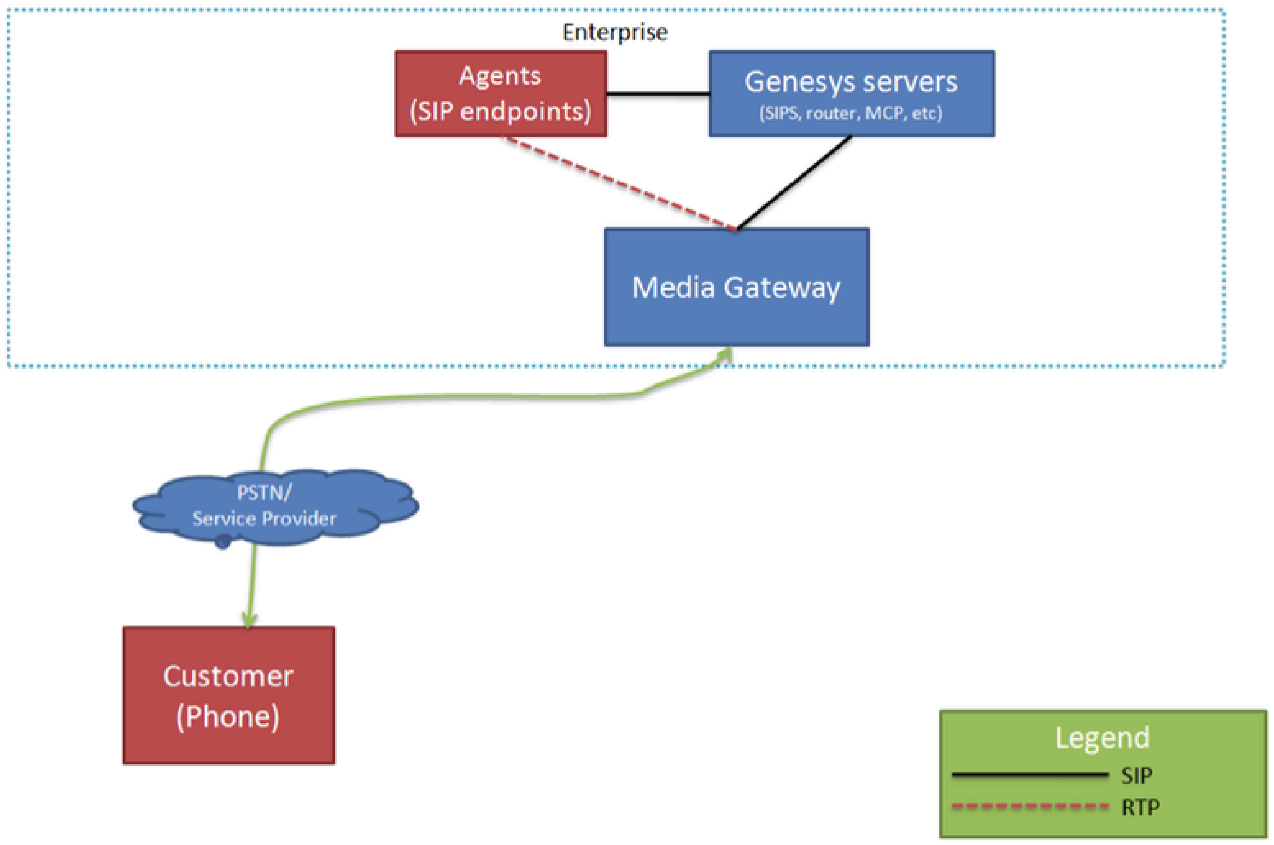
In order to work with firewalls or restrictive NAT, ICE requires an additional intermediary called a Traversal Using Relays around NAT server (TURN server). This server, which is deployed on a public address, can relay media packets for a TURN client so that they are ultimately able to reach another endpoint. The browser that wants to send and receive media is considered as a TURN client and connects to the TURN server to acquire a relay address. Using this address the browser client can use this as a candidate in ICE negotiation during SDP offer/answer negotiation.

The following diagram shows an example where the browser client on the left is a TURN client and the TURN client is behind a restrictive firewall that blocks UDP traffic but allows a connection to the TURN server. The web application that offers the use of this TURN server would provide the TURN server address and a credential in the web page. The browser uses the TURN server address and the credential to initiate a request for relay ports on its behalf. In the following example, port 34568 is assigned to the TURN client and the browser can create an SDP offer with port 34568 on the TURN server address (a public address) as a candidate.


Web Server
TURN Server
Port 34568
TURN relay
HTTP
64.93.101.5
SRTP
HTTP
84.69.255.15
Firewall
Router
192.168.1.101
192.168.1.101
Port 20000
Browser
Port 20002
Browser

Deployment Types

In a typical Genesys deployment, customers call into the contact center with a telephone and the contact center receives the inbound call from a service provider via various means. In the following diagram, the deployment assumes that the contact center is integrated with the service provider or PSTN via a media gateway or a session border controller.



When you are deploying the Genesys WebRTC Service, you must ensure that there is a media path between the browser client on one end of the call and either another browser client or a SIP client on the other end. The following topics discuss issues that can arise when the Genesys WebRTC Service is deployed within an enterprise network (or private cloud) or in a public cloud.

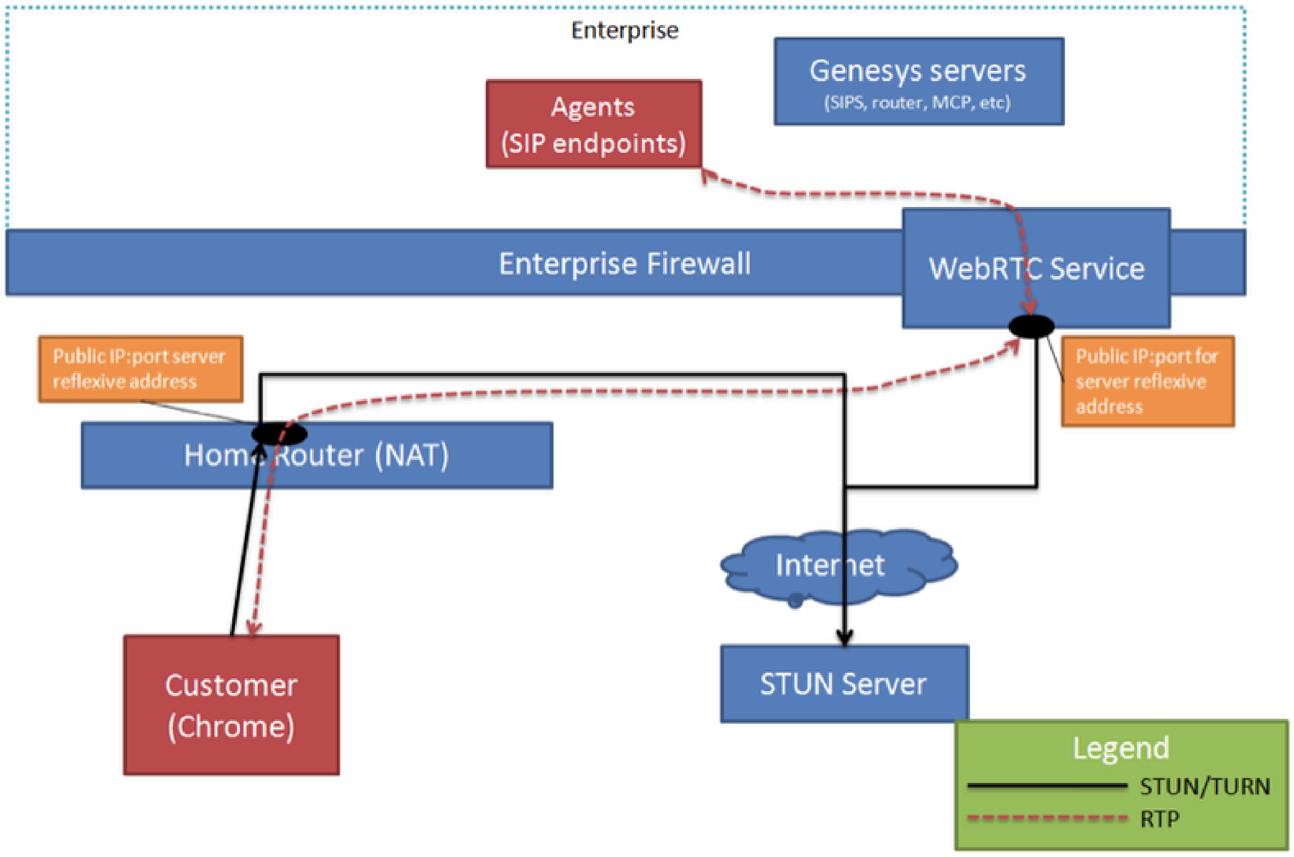
Private Cloud

In a private cloud, Genesys servers are generally deployed behind an enterprise firewall and are only able to access specific services from the Internet. The Genesys WebRTC Service is one of the servers that need to be accessible from the Internet for HTTP and SRTP. The WebRTC Service can then send SIP/RTP towards the enterprise network and is also able to bridge the media. In this type of deployment, the WebRTC Service must be deployed in the DMZ so that the component can handle inbound HTTP and SRTP traffic.

The following sections show the media path between customers and agents using various endpoints.

Customer (Browser) – Agent (SIP)

SIP agents are deployed within the enterprise. The WebRTC Service bridges media in from customers using browsers. To allow customers to connect to the WebRTC Service, the enterprise must deploy a STUN server on the public Internet so that the browser and the WebRTC Service can discover the server-reflexive address.



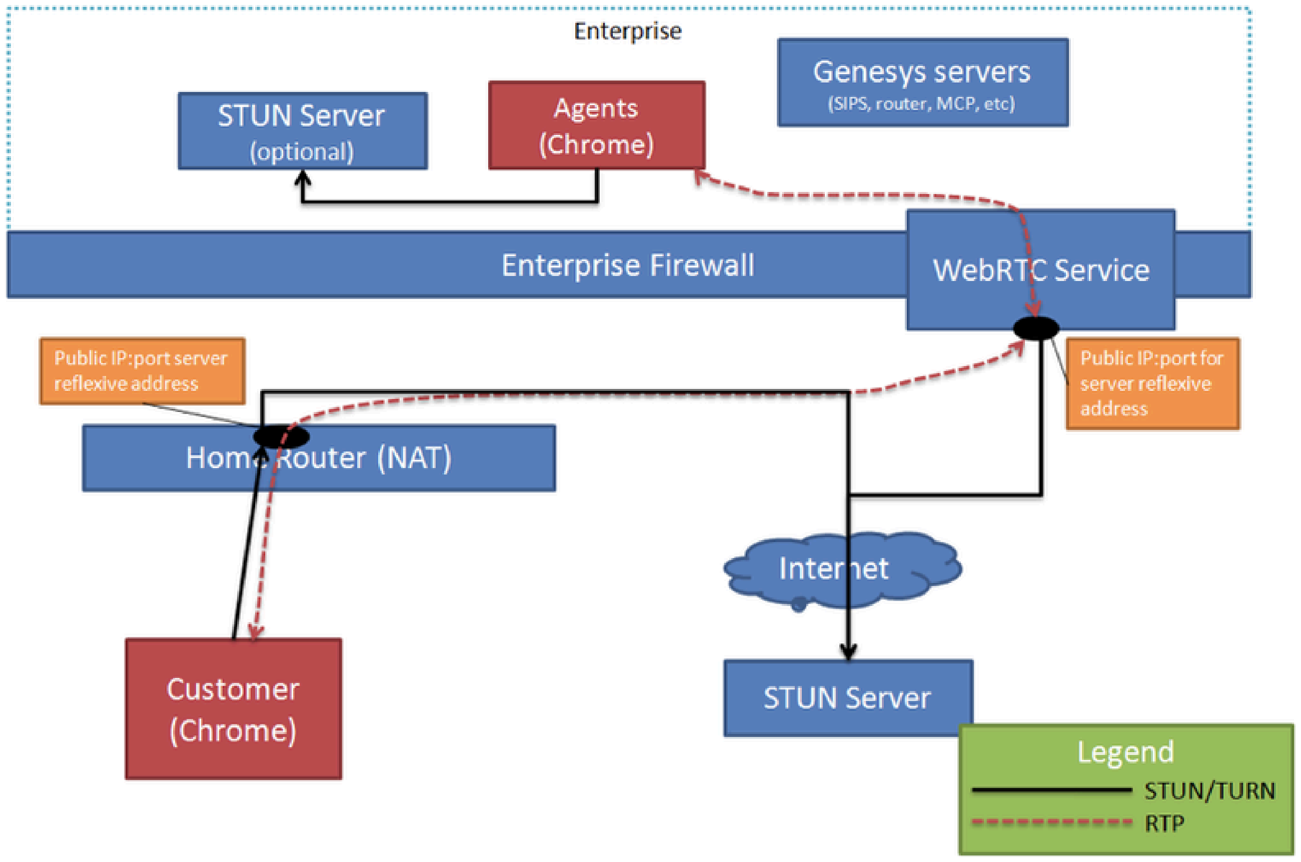
Customer (Browser) – Agent (Browser Inside the Enterprise)

Note: *A web-based agent desktop application implemented using the Genesys WebRTC JavaScript API could be used in this model. Genesys Workspace Web Edition is such a tool that supports audio-only calls, with no video support at this time.*

When agents use a browser as the endpoint to handle calls, the agent browser must be able to connect to the WebRTC Service in the DMZ. Since they are all hosted within the same enterprise network there are no restrictions about doing that. If NAT is deployed within the enterprise, then the enterprise must also deploy a STUN server within the enterprise network so that the browser and the WebRTC Service can resolve the server-reflexive address within the enterprise network.

On the customer-facing side, the media path is the same as described in the previous section; a STUN server must be deployed on the public network to allow customer browsers to resolve the server-reflexive address.

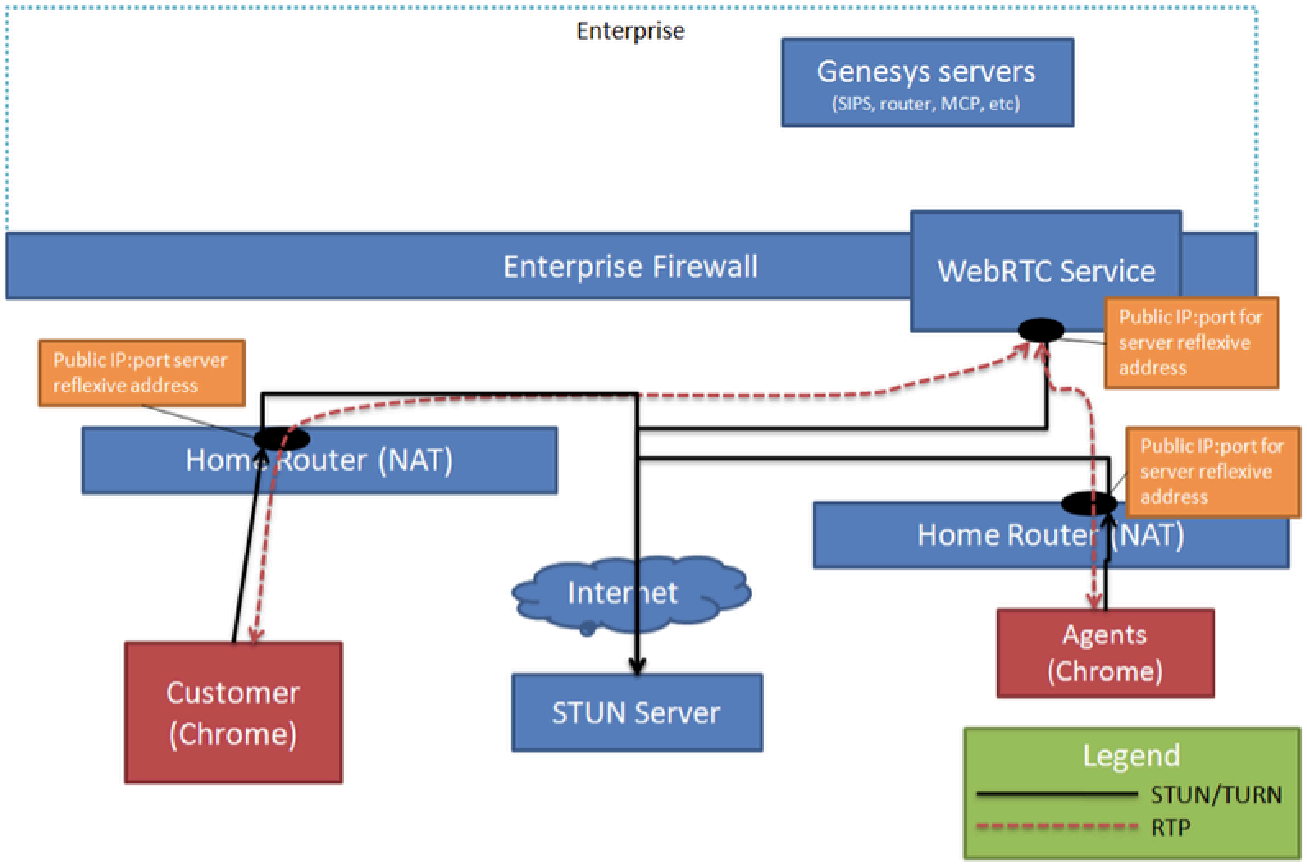
The WebRTC Service may still need to establish a media path to a Genesys server, such as a media server, for additional media services. It is therefore important to mention that the WebRTC Service is establishing the media path to the media server by means of the SIP/RTP path. Network address translation should not be configured between the WebRTC Service and the media servers, even though browser-using enterprise agents may be hosted in a subnet where NAT is enabled.



Customer (Browser) – Agent (Browser Outside the Enterprise)

Note: *A web-based agent desktop application implemented using the Genesys WebRTC JavaScript API could be used in this model. Genesys Workspace Web Edition is such a tool that supports audio-only calls, with no video support at this time.*

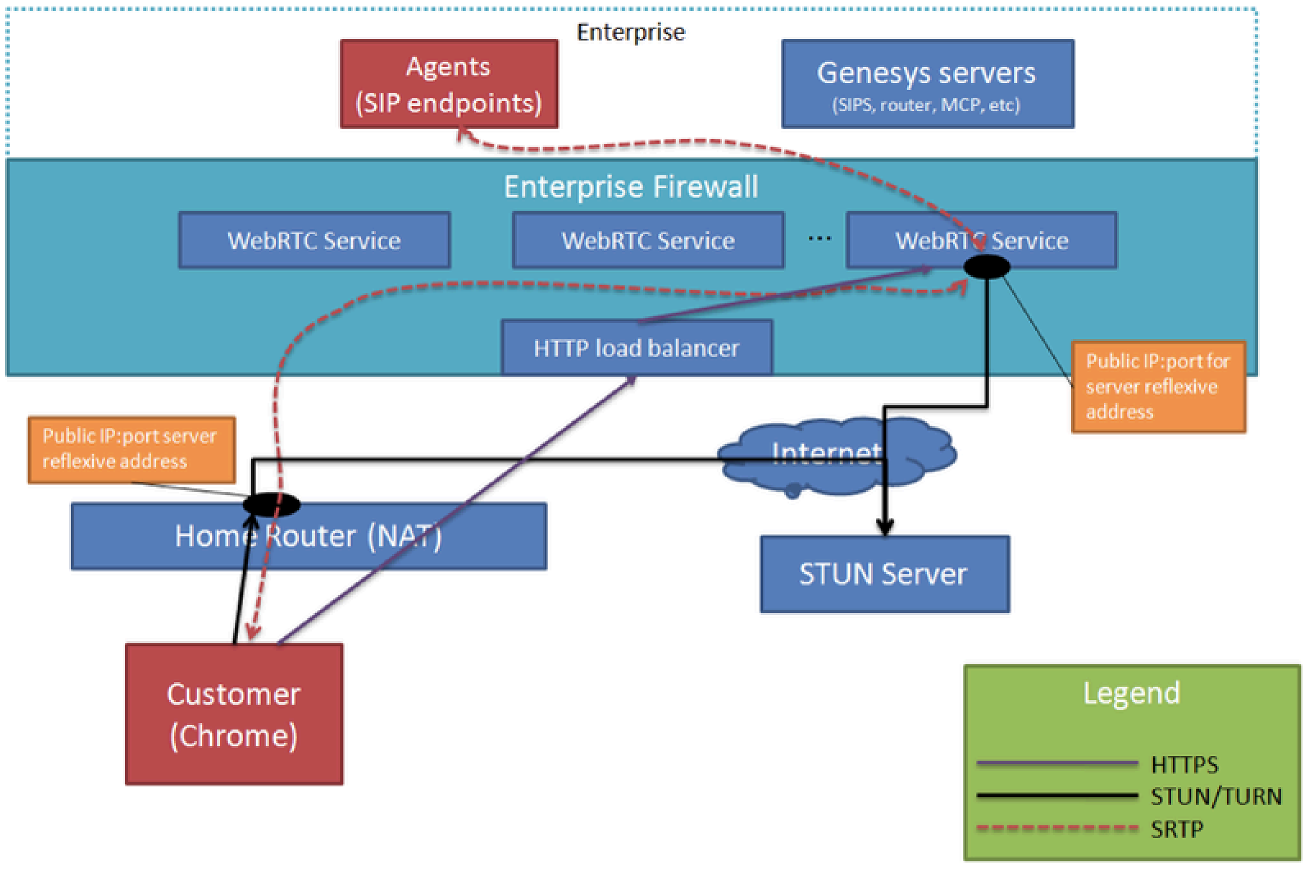
In deployments where remote agents are using a browser to handle calls, the remote agents connect their media paths to the WebRTC Service in the same manner as the customer browsers. The same public STUN server can service both customers and remote agents connecting to the WebRTC Service.



Scalability, Availability, Failover

Scalability

The Genesys WebRTC Service is scalable to multiple instances. This allows it to support a large number of simultaneous media sessions with the use of an HTTP load balancer in front of the group of WebRTC Service instances. Since the WebRTC Service handles user sessions and media sessions, it is important for the HTTP load balancer to support session persistence with the use of session cookies. This will ensure that all communication with a user is always bound to a specific instance of the WebRTC Service.



Availability

In order for the WebRTC Service to achieve maximum availability—that is, in order to ensure that it can service one hundred percent of capacity at all times—you should deploy redundant instances.

In general, if the WebRTC Service is deployed as a cluster of `N` instances and you wish to prepare for the potential failure of `M` instances, you can maintain maximum availability of the service by deploying `N + M` instances. The HTTP load balancer should be configured to detect the failure of any instances and to direct incoming sessions to the remaining instances.

Failover

Currently, the Genesys WebRTC Service does not provide continuity of the media session when a failure occurs at a Genesys WebRTC Service instance. When a failure happens during a media session, the session is considered failed.

Outside of a media session, a user who is connected to the WebRTC Service to receive inbound calls will need to establish a new session with a different instance of the WebRTC Service. When the JavaScript library detects a failure, it will attempt to re-establish the session. At this time, the HTTP load balancer will also detect the failure and direct new requests to another available instance of WebRTC Service.

Log Events Reference

20006

Level

Standard

Text

Invalid or missing value for the option: [section-name]:[option-name]

Attributes

[section name]

Name of the section where the option is specified

[option name]

Option name

Description

The WebRTC Gateway encountered an invalid configuration.

Actions

Check configuration.

20007

Level

Standard

Text

Failed to set SIP address. Check network interfaces and the [rsmp]:[sip-address] parameter.

Attributes

None

Description

The WebRTC Gateway failed to set the SIP address.

Actions

Check network addresses for the machine.

20008

Level

Standard

Text

Failed to set RTP address. Check network interfaces and the [rsmp]:[rtp-address] parameter.

Attributes

None

Description

The WebRTC Gateway failed to set the RTP address.

Actions

Check network addresses for the machine.

20009

Level

Standard

Text

libSRTP init FAILED (err=<error-code>).

Attributes

None

Description

The WebRTC Gateway failed to initialize SRTP.

Actions

Restart the Gateway.

20010

Level

Standard

Text

Failed to initialize SIP stack. Check the [rsmp]:[sip-port] parameter.

Attributes

None

Description

The WebRTC Gateway failed to initialize the SIP Stack.

Actions

Check to see whether the SIP port is being used by some other process.

20011

Level

Standard

Text

Failed to start ROAP server. Check the [rsmp]:[http-port] parameter.

Attributes

None

Description

The WebRTC Gateway failed to start the ROAP server.

Actions

Check to see whether the HTTP port is being used by some other process.

20012

Level

Standard

Text

TranscoderManager creation failed: <error message>.

Attributes

None

Description

The WebRTC Gateway failed to initialize the Transcoder Manager.

Actions

Restart the Gateway.

20013

Level

Standard

Text

The text of this message may vary.

Attributes

The attributes in this message may vary.

Description

A ROAP operation–related error has occurred.

Actions

Check the error message.

20014

Level

Standard

Text

The text of this message may vary.

Attributes

The attributes in this message may vary.

Description

A SIP operation–related error has occurred.

Actions

Check the error message.

20015

Level

Standard

Text

The text of this message may vary.

Attributes

The attributes in this message may vary.

Description

An ICE operation–related error has occurred.

Actions

Check the error message.

5060

Level

Standard

Text

WebRTC Gateway started.

Attributes

None

Description

Application started.

Actions

No action required.

Configuration Options Reference

The WebRTC configuration options are listed below. Click on an option name to display its properties.

ems

logconfig.MFSINK

logconfig.MFSINK (MF Sink Log Filter)			
Description	**Valid values**	**Default value**	**Takes effect**
Controls the log messages that are sent to the MF sink. The format is: "levels|moduleIDs|specifierIDs" (repeated if necessary). The values between the pipes can be in the format: "m-n,o,p" (for example, "0-4, 5, 6"). The wildcard character "*" can also be used to indicate all valid numbers. For example: "*|*|*" indicates that all log messages should be sent to the sink. Alternatively, "0,1|0-10|*|4|*|*" indicates that CRITICAL(0) and ERROR(1) level messages with module IDs in the range 0-10 will be sent to the sink; and all INFO(4) level messages will be sent as well.	Pipe-delimited ranges for log levels, module IDs, and specifier IDs. Ranges can be comma-separated integers or ranges of integers or "*".	*|*|*	immediately

metricsconfig.MFSINK

metricsconfig.MFSINK (MF Sink Metrics Filter)			
Description	**Valid values**	**Default value**	**Takes effect**
Specifies the metrics that are delivered to the MF Sink. "*" indicates that all metrics will be sent to the sink. Alternatively, "5-10,50-55,70,71" indicates that metrics with IDs 5, 6, 7, 8, 9, 10, 50, 51, 52, 53, 54, 55, 70, and 71 will be sent to the MF sink.	Comma-separated list of metric values or ranges. A metric value must be between 0 and 141 inclusive. The values "*" and blank are also allowed.	*	immediately

log

all

all (Output for level all)			
Description	**Valid values**	**Default value**	**Takes effect**
Specifies the outputs to which an application sends all log events. The log output types must be separated by a comma when more than one output level is configured.	• **stdout** — Log events are sent to the standard output (stdout). • **stderr** — Log events are sent to the standard error output (stderr). • **network** — Log events are sent to Message Server, which can reside anywhere on the network. Message Server stores the log events in the Log Database. Setting the all log level option to **network** enables an application to send log events of the standard, interaction, and trace levels to Message Server. Debug-level log events are neither sent to Message Server nor stored in the Log Database. • **memory** — Log events are sent to the memory output on the local disk. This is the safest output in terms of application performance. • **[filename]** — Log events are stored in a file with the specified name. If a path is not specified, the file is created in	../logs/rsmplog	immediately

	the application's working directory.		

check-point

check-point (Check point interval)			
Description	**Valid values**	**Default value**	**Takes effect**
Specifies, in hours, how often the application generates a check point log event, to divide the log into sections of equal time. By default, the application generates this log event every hour. Setting the option to 0 prevents the generation of check-point events.	0 - 24	1	immediately

compatible-output-priority

compatible-output-priority (Enable 6.X compatible log output priority)			
Description	**Valid values**	**Default value**	**Takes effect**
Specifies whether the application uses 6.x output logic.	• **true** — The log of the level specified by "Log Output Options" is sent to the specified output. • **false** — The log of the level specified by "Log Output Options" and higher levels is sent to the specified output.	false	immediately

debug

debug (Output for level debug)			
Description	**Valid values**	**Default value**	**Takes effect**
Specifies the outputs to which an application sends the log events of the debug level and higher (that is, log events of the standard, interaction, trace, and debug levels). The log	• **stdout** — Log events are sent to the standard output (stdout). • **stderr** — Log events are sent to the	../logs/rsmplog	immediately

output types must be separated by a comma when more than one output level is configured.	standard error output (stderr). • **memory** — Log events are sent to the memory output on the local disk. This is the safest output in terms of application performance. • **network** — Log events are sent to Message Server, which can reside anywhere on the network. Message Server stores the log events in the Log Database. Setting the debug log level option to **network** enables an application to send log events of the standard, interaction, and trace levels to Message Server. Debug-level log events are neither sent to Message Server nor stored in the Log Database. • **[filename]** — Log events are stored in a file with the specified name. If a path is not specified, the file is created in the application's working directory.		

expire

expire (Log Expiration)			
Description	**Valid values**	**Default value**	**Takes effect**
Determines whether log files expire. If they do, sets the measurement for determining when they expire, along with the maximum number	• **false** — No expiration; all generated segments are stored.	20	immediately

of files (segments) or days before the files are removed.	• **[number] file or [number]** — Sets the maximum number of log files to store. Specify a number from 1-100. • **[number] day** — Sets the maximum number of days before log files are deleted. Specify a number from 1-100.		

interaction

interaction (Output for level interaction)

Description	Valid values	Default value	Takes effect
Specifies the outputs to which an application sends the log events of the interaction level and higher (that is, log events of the standard and interaction levels). The log outputs must be separated by a comma when more than one output level is configured.	• **stdout** — Log events are sent to the standard output (stdout). • **stderr** — Log events are sent to the standard error output (stderr). • **network** — Log events are sent to Message Server, which can reside anywhere on the network. Message Server stores the log events in the Log Database. Setting the interaction log level option to **network** enables an application to send log events of the standard and interaction levels to Message Server. • **memory** — Log events are sent to the memory output on the local disk. This is the safest output in terms of	../logs/rsmplog	immediately

	application performance. • • **[filename]** — Log events are stored in a file with the specified name. If a path is not specified, the file is created in the application's working directory.		

keep-startup-file

keep-startup-file (Keep startup log file)

Description	Valid values	Default value	Takes effect
Specifies whether a startup segment of the log, containing the initial T-Server configuration, is to be kept. If it is, this option can be set to true or to a specific size. If set to true, the size of the initial segment will be equal to the size of the regular log segment defined by the segment option. The value of this option will be ignored if segmentation is turned off (that is, if the segment option set to false).	• **false** — No startup segment of the log is kept. • **true** — A startup segment of the log is kept. The size of the segment equals the value of the segment option. • **[number] KB** — Sets the maximum size, in kilobytes, for a startup segment of the log. • **[number] MB** — Sets the maximum size, in megabytes, for a startup segment of the log.	false	After restart

memory

memory (Memory snapshot file name)

Description	Valid values	Default value	Takes effect
Specifies the name of the file to which the application regularly prints a snapshot of the memory output, if it is configured to do this. The new snapshot overwrites the	[string] (memory file name)	(blank)	immediately

previously written data. If the application terminates abnormally, this file will contain the latest log messages. Memory output is not recommended for processors with a CPU frequency lower than 600 MHz.			

message_format

message_format (Log messages format)

Description	Valid values	Default value	Takes effect
Specifies the format of log record headers that an application uses when writing logs in the log file. Using compressed log record headers improves application performance and reduces the log file's size. With the value set to short: • Headers of the log file or the log file segment contain information about the application (such as the application name, application type, host type, and time zone), whereas single log records within the file or segment omit this information. • A log message priority is abbreviated to Std, Int, Trc, or Dbg, for standard, interaction, trace, or debug messages, respectively. • The message ID does not contain the prefix GCTI or the application type ID. A log record in the full format looks like	• **short** — An application uses compressed headers when writing log records to its log file. • **full** — An application uses complete headers when writing log records to its log file.	short	immediately

this: 2002-05-07T18:11:38.196 Standard localhost cfg_dbserver GCTI-00-05060 Application started A log record in the short format looks like this: 2002-05-07T18:15:33.952 Std 05060 Application started			

messagefile

messagefile (Message file)			
Description	**Valid values**	**Default value**	**Takes effect**
Specifies the file name for application-specific log events. The name must be valid for the operating system on which the application is running. The option value can also contain the absolute path to the application-specific *.lms file. Otherwise, an application looks for the file in its working directory.	[string].lms (message file name)	(blank)	Immediately, if an application cannot find its *.lms file at startup

print-attributes

print-attributes (Enable printing extended attributes)			
Description	**Valid values**	**Default value**	**Takes effect**
Specifies whether the application attaches extended attributes, if any exist, to log events that it sends to log output. Typically, log events of the interaction log level and audit-related log events contain extended attributes. Setting this option to true enables audit capabilities, but negatively affects performance. Genesys recommends enabling	• **true** Attaches extended attributes, if any exist, to a log event sent to log output. • **false** Does not attach extended attributes to a log event sent to log output.	false	immediately

this option for Solution Control Server and Configuration Server when using audit tracking. For other applications, refer to Genesys Combined Log Events Help to find out whether an application generates interaction-level and audit-related log events; if it does, enable the option only when testing new interaction scenarios.			

segment

segment (Log Segmentation)

Description	Valid values	Default value	Takes effect
Specifies whether there is a segmentation limit for a log file. If there is, sets the mode of measurement, along with the maximum size. If the current log segment exceeds the size set by this option, the file is closed and a new one is created.	• **false** — No segmentation is allowed. • **[number] KB or [number]** — Sets the maximum segment size, in kilobytes. The minimum segment size is 100 KB. • **[number] MB** — Sets the maximum segment size, in megabytes. • **[number] hr** — Sets the number of hours for the segment to stay open. The minimum number is 1 hour.	10000	immediately

spool

spool (Folder for the temporary network log output files)

Description	Valid values	Default value	Takes effect
Specifies the folder, including full path to it, in which an application creates temporary files related to network log	[path] (the folder, with the full path to it)	(blank)	immediately

output. If you change the option value while the application is running, the change does not affect the currently open network output.			

standard

standard (Output for level standard)			
Description	**Valid values**	**Default value**	**Takes effect**
Specifies the outputs to which an application sends the log events of the standard level. The log output types must be separated by a comma when more than one output level is configured.	• **stdout** — Log events are sent to the standard output (stdout). • **stderr** — Log events are sent to the standard error output (stderr). • **network** — Log events are sent to Message Server, which can reside anywhere on the network. Message Server stores the log events in the Log Database. Setting the standard log level option to **network** enables an application to send log events of the standard level to Message Server. • **memory** — Log events are sent to the memory output on the local disk. This is the safest output in terms of application performance. • **[filename]** — Log events are stored in a file with the specified name. If a path is not specified, the file is created in the application's	../logs/rsmplog	immediately

	working directory.		

time_convert

time_convert (Time generation for log messages)			
Description	**Valid values**	**Default value**	**Takes effect**
Specifies the system in which an application calculates the log record time when generating a log file. The time is converted from the time in seconds since the Epoch (00:00:00 UTC, January 1, 1970). • **Local Time (local)** — The time of log record generation is expressed as a local time, based on the time zone and any seasonal adjustments. Time zone information from the application's host computer is used. • **Coordinated Universal Time (utc)** — The time of log record generation is expressed as Coordinated Universal Time (UTC).	• **local** • **utc**	local	immediately

time_format

time_format (Time format for log messages)			
Description	**Valid values**	**Default value**	**Takes effect**
Specifies how to represent, in a log file, the time when an application generates log records. A log record's time field in the ISO 8601 format looks like	• **time** • **locale** • **ISO8601**	time	immediately

2001-07-24T04:58:10.123 • **HH:MM:SS.sss (time)** — The time string is formatted according to the HH:MM:SS.sss (hours, minutes, seconds, and milliseconds) format. • **According to the system's locale (locale)** The time string is formatted according to the system's locale. • **ISO 8601 format (ISO8601)** — The date in the time string is formatted according to the ISO 8601 format. Fractional seconds are given in milliseconds.			

trace

trace (Output for level trace)			
Description	**Valid values**	**Default value**	**Takes effect**
Specifies the outputs to which an application sends the log events of the trace level and higher (that is, log events of the standard, interaction, and trace levels). The log outputs must be separated by a comma when more than one output level is configured.	• **stdout** — Log events are sent to the standard output (stdout). • **stderr** — Log events are sent to the standard error output (stderr). • **network** — Log events are sent to Message Server, which can reside anywhere on the network. Message Server stores the log events in the Log Database. Setting the trace log level option to **network**	../logs/rsmplog	immediately

	enables an application to send log events of the standard, interaction, and trace levels to Message Server. • **memory** — Log events are sent to the memory output on the local disk. This is the safest output in terms of application performance. • **[filename]** — Log events are stored in a file with the specified name. If a path is not specified, the file is created in the application's working directory.		

verbose

verbose (Verbose Level)

Description	Valid values	Default value	Takes effect
Determines whether a log output is created. If it is, specifies the minimum level of log events generated. The log events levels, starting with the highest priority level, are standard, interaction, trace, and debug.	• **all** — All log events (that is, log events of the standard, trace, interaction, and debug levels) are generated. • **debug** — The same as all. • **trace** — Log events of the trace level and higher (that is, log events of the standard, interaction, and trace levels) are generated, but log events of the debug level are not generated. • **interaction** — Log events of the	debug	immediately

	interaction level and higher (that is, log events of the standard and interaction levels) are generated, but log events of the trace and debug levels are not generated. • **standard** — Log events of the standard level are generated, but log events of the interaction, trace, and debug levels are not generated. • **none** — No output is produced.		

rsmp

allow-anonymous-user

allow-anonymous-user (Allow Anonymous User)

Description	Valid values	Default value	Takes effect
Set this to true (default) to enable anonymous users to sign-in to the WebRTC Gateway. If set to false, then only registered users for SIP Server can sign-in.	true, false	true	At start or restart

allow-ipv6

allow-ipv6 (Allow IPv6)

Description	Valid values	Default value	Takes effect
Controls whether IPv6 is allowed in the WebRTC Gateway.	true, false	false	At start or restart

codecs

codecs (Supported and Default Media Codecs)

Description	Valid values	Default value	Takes effect

Codecs that are not listed here will not be used in an offer or answer. The codecs currently supported are: • pcmu (G.711 mu-Law) • pcma (G.711 A-Law) • g722 • g729 (G.729/a/b) • opus (non-transcoding case) • iSAC/16000 (non-transcoding case) • iSAC/32000 (non-transcoding case) • telephone-event • vp8 • h264	A comma separated list of supported codecs (from the Description column) within brackets. A codec's clock rate (in Hz) can optionally be specified after its name followed by a '/'. A default payload type number can be specified using the format name=<pt>, or name=(pt=<pt>). The latter format needs to be used if an fmtp is to be specified, which will be specified as fmtp=<fmtp>. A comma is used as a separator between the different values. All or part of the fmtp value can be enclosed within square brackets, where those brackets will be removed when used in an offer, and in the case of an answer, the brackets and the content will be replaced by the fmtp value from the remote offer. See the default value for an example.	(g722,pcmu,pcma,opus,g729,telephone-event=126,vp8=100,h264=(pt=108,fmtp="[profile-level-id=42000B;packetization-mode=1]"))	At start or restart

domain-whitelist

domain-whitelist (List of allowed domains)			
Description	**Valid values**	**Default value**	**Takes effect**
Specifies a comma-separated white-list of allowed domains. A wildcard character (*) will be allowed in a domain value for specifying an arbitrary scheme, sub-domain, or port, as illustrated below: • *://sub.foo.com:8080—Any URL scheme, that is, either `http` or `https`, is accepted. • https://*.foo.com:8090—Any sub-domain of `foo.com` is accepted. • http://*.foo.com:*—Any port number or sub-domain is accepted.	See description	(blank)	At start or restart

• *foo.com:*—Any scheme, port, or sub-domain is accepted. Note that if the port number is not specified, then the default HTTP port 80 is assumed. • A check will be performed against this list only if the request contains an `Origin` header. Requests without an `Origin` header are not necessarily Cross-Origin Resource Sharing (CORS) requests. If the domain value in the `Origin` header matches any of the values in the white-list, then the request will be accepted, and the `Access-Control-Allow-Origin` header in the response will be populated with the `Origin` value. • If the value of the white-list is empty (which is the default), then CORS checking will be disabled. The `Access-Control-Allow-Origin` header in the response will still be set using the `Origin` value in the request, if there is one; otherwise, the value of this header will be set to `*`. • If the white-list contains valid values, and the `Origin` header doesn't exist or match any of the white-list domains,			

then the request will be rejected with an HTTP response code of `403 Forbidden`.			

enable-https

enable-https (Enable HTTPS)			
Description	**Valid values**	**Default value**	**Takes effect**
Enables HTTPS. When set to true, the other https-* parameters need to be set correctly using the host's security certificate as well. When enabled, the WebRTC Gateway supports secure HTTPS requests only. Non-secure HTTP requests will not work. Note that HTTPS is required with the latest Chrome browser by default. If an HTTPS proxy is used between the browser/client and the gateway, and it can convert the HTTPS requests to HTTP, then HTTPS does not need to be enabled here.	true, false	false	At start or restart

enable-transcoding

enable-transcoding (Enable Transcoding)			
Description	**Valid values**	**Default value**	**Takes effect**
Transcoding of audio and/or video between the SIP and Web sides is enabled if this value is set to true. Otherwise, transcoding will be disabled. When enabled, transcoding will be activated for a media type, only when there is no common codec negotiated between the sides, or when a codec sent by one side is not supported by the other	true, false	true	At start or restart

side.			

http-port

http-port (HTTP Port)			
Description	**Valid values**	**Default value**	**Takes effect**
The HTTP or HTTPS port used for WebRTC signaling. Port 443 is recommended for HTTPS.		8086	At start or restart

https-cert

https-cert (HTTPS Certificate)			
Description	**Valid values**	**Default value**	**Takes effect**
For Windows, the thumbprint obtained from the user certificate generated for the host. For Linux, the fullpath of the host certificate file (.pem).		(blank)	At start or restart

https-cert-key

https-cert-key (HTTPS Certificate Key)			
Description	**Valid values**	**Default value**	**Takes effect**
Applicable for Linux only. The fullpath of the host private key file (.pem).		(blank)	At start or restart

https-trusted-ca

https-trusted-ca (HTTPS Certificate Authority)			
Description	**Valid values**	**Default value**	**Takes effect**
Applicable for Linux only. The fullpath of the Certificate Authority file (.pem).		(blank)	At start or restart

http-trace

http-trace (HTTP Trace)			
Description	**Valid values**	**Default value**	**Takes effect**
Traces HTTP requests and responses	true, false	false	At start or restart

reporting-service-type

reporting-service-type (WebRTC Reporting Service Type)			
Description	**Valid values**	**Default value**	**Takes effect**
SIP calls are reported out of the box when SIP Server and ICON are configured. When this parameter is set, the service_type key-value pair is sent to SIP Server and then reported to ICON. This allows the reports for the WebRTC service to be filtered based on the service type specified here. To disable the sending of a service type, set this parameter value to "none".		WebRTC	At start or restart

rtp-address

rtp-address (RTP Address)			
Description	**Valid values**	**Default value**	**Takes effect**
Allows for configuration of a specific IP address for SDP c= line. If not set, the stack will attempt to detect the IP address automatically. This is useful for AWS instances or multi-homed hosts. For example, in an AWS instance you can set this to the elastic-IP. This setting applies to the SIP side only.		(blank)	At start or restart

rtp-trace-level

rtp-trace-level (RTP Trace Level)			
Description	**Valid values**	**Default value**	**Takes effect**
The RTP trace level controls how many packets are printed into the log.	• **0** — Print "key" packets only (1st RTP/RTCP and last RTCP) to keep log small. • **1** — Print RTP/RTCP	1	At start or restart

	packets periodically, but no more than 1 pkt per second. • **2** — Print more often, and always print all errors and "bad" packets. • **3** — Print a few RTP packets per second and all RTCP and "bad" packets. • **4** — Print ALL packets - WARNING: log will be huge, may affect performance.		

sip-added-codecs

sip-added-codecs (SIP Added Codecs)

Description	Valid values	Default value	Takes effect
When transcoding is enabled, codecs from this list will be appended to the codec list for offers to a SIP endpoint, except for the codecs that are already in the original offer. Note that all codecs in an offer from one side will be added to the resulting offer to the other side.	List of codecs using the same format as codecs option. If not specified here, the pt and the fmtp values will be used from the list specified in the codecs option. Note that at least one video codec should be specified, and this codec should most likely be supported by the SIP side. Otherwise, the call may fail even if transcoding is supported. For example, if the Web side offers only VP8, and the SIP side only supports H.264, sip-added-codecs will need to contain h264. Note, however, that the answer to the Web side will contain only H.264 based on the reply from the SIP side, unless web-added-codecs contains vp8. This may not be an issue for audio, given that pcmu is supported by both sides. However, if a common audio codec is disallowed on one side, then it should be added to the other side for similar reasons as those given for video codecs.	(vp8,h264)	At start or restart

sip-address

sip-address (SIP Address)			
Description	**Valid values**	**Default value**	**Takes effect**
Allows for configuration of a specific IP address for SIP Via or Contact. If not set, the stack will attempt to detect the IP address automatically. This is useful for AWS instances or multi-homed hosts. For example, in an AWS instance you can set this to the elastic-IP.		(blank)	At start or restart

sip-disallowed-codecs

sip-disallowed-codecs (SIP Disallowed Codecs)			
Description	**Valid values**	**Default value**	**Takes effect**
Disallowed codecs for the SIP side. An offer or answer to the SIP side may not use any of these codecs.	List of codecs using the same format as the codecs option. No need to specify "pt" or "fmpt" for a codec.	(blank)	At start or restart

sip-no-avpf

sip-no-avpf (SIP Side AVPF Negotiation)			
Description	**Valid values**	**Default value**	**Takes effect**
Set this to true in order not to negotiate AVPF in SDP on the SIP side (RFC 4585). This is necessary to work with SIP endpoints that do not support AVPF. Note that regardless of the value of this option, if sip-no-rtcpfb = false, RTCP feedback messages will be forwarded to the SIP side. These settings are useful for a Chrome-to-Chrome call.	true, false	true	At start or restart

sip-no-rtcpfb

sip-no-rtcpfb (Forwarding RTCP Feedback Messages to SIP Side)			
Description	**Valid values**	**Default value**	**Takes effect**
If set to false, RTCP feedback messages sent by a WebRTC client in accordance with (RFC	true, false	false	At start or restart

4585) will be forwarded to the corresponding SIP endpoint in a call. A true value will disable this. Note that even though endpoints should ignore RTCP packets of unknown types, some may have issues with this.			

sip-port

sip-port			
Description	**Valid values**	**Default value**	**Takes effect**
Local port used for SIP signaling.		5066	At start or restart

sip-preferred-ipversion

sip-preferred-ipversion (Preferred IP version to be used for SIP)			
Description	**Valid values**	**Default value**	**Takes effect**
Preferred IP version to be used for SIP.	• **ipv4** IPv4 • **ipv6** IPv6	ipv4	At start or restart

sip-proxy

sip-proxy (SIP Proxy)			
Description	**Valid values**	**Default value**	**Takes effect**
The IP address(es) or the FQDN(s) of one or two (in the case of HA) SIP Server(s), separated by a comma. Optionally the port can be specified for each address, separated by a colon (:). If the default SIP port 5060 is used, it can be omitted. In all scenarios, Genesys SIP Servers are specified, and are used by the WebRTC Gateway as both SIP Proxy and Registrar. Only one address is used at a time, and the gateway switches to the other when a timeout is	It should be in the following format: <IP-address|FQDN>[:<port>][,<IP-address|FQDN>[:<port>]]	127.0.0.1	At start or restart

detected with a SIP request. Therefore, warm failover can be achieved by specifying two SIP Servers configured in HA mode.			

sip-proxy-srv

sip-proxy-srv			
Description	**Valid values**	**Default value**	**Takes effect**
The DNS SRV address for the HA SIP Server pair (without the port number), of which the addresses are specified using the `sip-proxy` option. When a SIP request arrives with the SRV address, the gateway will translate it to the currently active SIP Server address and use it in its response.		(blank)	At start or restart

sip-register

sip-register (List of DNs for Registration)			
Description	**Valid values**	**Default value**	**Takes effect**
A list of comma separated entries that specify DNs configured in the SIP Server. Any web client that registers with the gateway using an ID that matches any one of these DNs will be registered with the SIP Server by the gateway, so that it can receive calls via the SIP Server and the gateway. Therefore, a new deployment should set this parameter correctly before it can work with the SIP Server.	Each entry can be a single DN, a range of DNs specified with two hard coded values separated by a hyphen (-), or a range of DNs specified using wildcard characters (* and/or ?). A valid DN string contains only digits [0-9], '+' (as prefix), and/or the wildcard characters. Any leading zeros are preserved. Example: `1020,2020-2050,003000,556*,100*10,123??78,+408555`	(blank)	At start or restart

sip-rtp-max-port

sip-rtp-max-port (SIP-side Max RTP Port)			
Description	**Valid values**	**Default value**	**Takes effect**
UDP port range for SIP-		9999	At start or restart

side RTP connection.			

sip-rtp-min-port

sip-rtp-min-port (SIP-side Min RTP Port)			
Description	**Valid values**	**Default value**	**Takes effect**
UDP port range for SIP-side RTP connection.		9000	At start or restart

sip-srtp-mode

sip-srtp-mode (SIP Side SRTP Mode)			
Description	**Valid values**	**Default value**	**Takes effect**
SRTP mode that is to be used in SDP negotiation on the SIP side.	• **none** no SRTP will be used • **optional** offer two m-lines with and without SRTP (for each media), and accept either • **strict** offer secure option only with SRTP; reject any non-secure offers	none	At start or restart

sip-tls-cert

sip-tls-cert (SIP TLS Certificate)			
Description	**Valid values**	**Default value**	**Takes effect**
For Windows, the thumbprint obtained from the user certificate generated for the host. For Linux, the fullpath of the host certificate file (.pem)		(blank)	At start or restart

sip-tls-cert-key

sip-tls-cert-key (SIP TLS Certificate Key)			
Description	**Valid values**	**Default value**	**Takes effect**
Applicable for Linux only. The fullpath of the host private key file (.pem).		(blank)	At start or restart

sip-tls-port

sip-tls-port (SIP TLS Port)			
Description	**Valid values**	**Default value**	**Takes effect**
SIP TLS Port. To disable TLS transport for SIP traffic altogether, set to 0.		0	At start or restart

sip-tls-trusted-ca

sip-tls-trusted-ca (SIP TLS Certificate Authority)			
Description	**Valid values**	**Default value**	**Takes effect**
Applicable for Linux only. The fullpath of the Certificate Authority file (.pem).		(blank)	At start or restart

stun-server

stun-server (STUN Server)			
Description	**Valid values**	**Default value**	**Takes effect**
Optional STUN server specification (port may be omitted, if default STUN port 3478 is used). When the gateway is in a private network with only the host's private address visible to it, a STUN server sitting outside that network may be required by the ICE processing for gathering the host's public (server reflexive) address. Only local addresses are gathered when STUN or TURN is not configured.	<IP-address|FQDN>[:<port>], where port can be omitted, if default port 3478 is used.	(blank)	At start or restart

turn-passwd

turn-passwd (TURN Password)			
Description	**Valid values**	**Default value**	**Takes effect**
The TURN password to use for the allocation.		(blank)	At start or restart

turn-relay-type

turn-relay-type (TURN Relay Type)			
Description	**Valid values**	**Default value**	**Takes effect**

The type of relay to use. TCP(1) and UDP(0) are supported; TLS is not supported. The default is UDP. If this parameter is not defined, but the TURN server URL contains the query string "transport=tcp", then TCP will be chosen.		0	At start or restart

turn-server

turn-server (TURN Server)			
Description	**Valid values**	**Default value**	**Takes effect**
Optional TURN server specification. A TURN server is not typically required on the gateway side, but it may need to be configured on the browser/client side when there is a strict firewall between the client and the gateway that will prevent RTP/UDP traffic. Only local addresses are gathered when STUN or TURN is not configured.	<IP-address|FQDN>[:<port>], where port can be omitted, if default port 3478 is used.	(blank)	At start or restart

turn-user

turn-user (TURN User)			
Description	**Valid values**	**Default value**	**Takes effect**
The TURN username to use for the allocation.		(blank)	At start or restart

use-sid-from-url

use-sid-from-url (Use Session Id from URL)			
Description	**Valid values**	**Default value**	**Takes effect**
When this is set to true, Genesys WebRTC Gateway uses HTTP request URL parameters or value of Cookie header to match HTTP request with existing client session. If it is false, Genesys WebRTC Gateway uses value of Cookie header to match	true, false	true	After Genesys WebRTC Gateway restart

the HTTP request with existing client session. **Note:** This option works with Genesys WebRTC Java Script API from version 8.5.210.35.			

web-added-codecs

web-added-codecs (Web Added Codecs)			
Description	**Valid values**	**Default value**	**Takes effect**
When transcoding is enabled, codecs from this list will be appended to the codec list for offers to a WebRTC endpoint, except for the codecs that are already in the original offer. If G.722 is not supported on the SIP side, and yet G.722 is desired on the Web side, add "g722" to web-added-codecs for the cases where the SDP offer comes from the SIP side. Note that transcoding will take place in this case between G.722 and the audio codec used on the SIP side. The other comments for sip-added-codecs are applicable here as well.	List of codecs using the same format as `sip-added-codecs` option.	(g722,pcmu,vp8)	At start or restart

web-disable-sdes

web-disable-sdes			
Description	**Valid values**	**Default value**	**Takes effect**
When this is set to true, the crypto attribute to support SDES-SRTP (RFC 4568) will not be in initial offers to Web clients. If it is false, initial offers will have both fingerprint and crypto attributes. Note that this option does not prevent supporting SDES-SRTP when the client only offers crypto	true, false	true	At start or restart

attribute(s). However, SDES-SRTP use is discouraged, and only Chrome supports it for backward compatibility.			

web-disallowed-codecs

web-disallowed-codecs (Web Disallowed Codecs)

Description	Valid values	Default value	Takes effect
Disallowed codecs for the WebRTC side. An offer or answer to the Web side may not use any of these codecs.	List of codecs using the same format as `sip-disallowed-codecs` option.	(h264)	At start or restart

web-dtls-certificate

web-dtls-certificate (Certificate for Web-side DTLS)

Description	Valid values	Default value	Takes effect
Path of the X.509 certificate file to be used with Web-side DTLS. This file can also contain the private key for the certificate, in which case web-dtls-privatekey does not need to be set. The certificate file is mandatory for DTLS to work. The default certificate already contains the private key.		../config/x509_certificate.pem	At start or restart

web-dtls-cipherlist

web-dtls-cipherlist (Cipher List for DTLS)

Description	Valid values	Default value	Takes effect
A list of cipher strings to be used with DTLS on the Web side. For information on the format, see OpenSSL ciphers. The default cipher string should work well.		(blank)	At start or restart

web-dtls-keypassword

web-dtls-keypassword (Password for DTLS Certificate Key)

Description	Valid values	Default value	Takes effect

The password for the private key specified using web-dtls-privatekey, if used.		(blank)	At start or restart

web-dtls-privatekey

web-dtls-privatekey (Private Key for DTLS Certificate)			
Description	**Valid values**	**Default value**	**Takes effect**
Path of the private key file for the certificate specified in web-dtls-certificate. This parameter is not necessary if the certificate file also contains the private key.		(blank)	At start or restart

web-enable-dtls

web-enable-dtls (Enable DTLS on the Web-side)			
Description	**Valid values**	**Default value**	**Takes effect**
When this is set to true, DTLS-SRTP (RFC 5763) will be enabled on the Web side. When enabled, it will be signaled in an SDP offer sent by the gateway using the fingerprint attributes, though there will also be crypto attributes in SDP for SDES-SRTP (RFC 4568) support, provided that SDES-SRTP is not disabled using the option `web-disable-sdes`. When an offer or answer comes in with only crypto attributes, then SDES-SRTP will still be supported. When this is set to false, only SDES-SRTP will be supported.	true, false	true	At start or restart

web-ice-addresses

web-ice-addresses (Web ICE addresses)			
Description	**Valid values**	**Default value**	**Takes effect**
Allows configuration of a local IP address list to	A comma-separated list of valid IP addresses.		At restart

be used with ICE on the web/ROAP side. Comma is the delimiter, and each IP address could be IPv4 or IPv6 (no need for square brackets). These addresses are used by ICE to gather the host candidates.			

web-media-bundle

web-media-bundle (Media Bundle on Web-side)

Description	Valid values	Default value	Takes effect
Set this to true to enable media bundling on the ROAP side (see http://tools.ietf.org/html/draft-ietf-mmusic-sdp-bundle-negotiation-03). When enabled, it will be signalled in an SDP offer sent by the gateway, and it will be accepted from an inbound SDP offer. If both sides agree, then the same media port will be used for both audio and video. With media bundle, it is assumed that RTCP multiplexing is also supported. Disabling this bundle option is discouraged.	true, false	true	At start or restart

web-nack-enabled

web-nack-enabled (Enable Web Side NACK Sending)

Description	Valid values	Default value	Takes effect
Set this to true (default) to enable RTCP NACK (transport layer) feedback messages as per (RFC 4585). Set this to false to disable this feature. The minimum time between two NACK messages is currently restricted to one second.	true, false	true	At start or restart

web-offer-bundle-only

web-offer-bundle-only			
Description	**Valid values**	**Default value**	**Takes effect**
This option helps to work around some renegotiation issues with Firefox. When an older version of Firefox without bundle support is used (version 37 and lower), set this option to `false`.	true, false	true	At start or restart

web-pli-always

web-pli-always (Force Web Side PLI Requests)			
Description	**Valid values**	**Default value**	**Takes effect**
If this parameter is set to true and web-pli-mintime is nonzero, RTCP PLI feedback messages (RFC 4585) will be sent on a Web-side video leg at every web-pli-mintime interval, regardless of transcoding or packet losses.	true, false	true	At start or restart

web-pli-mintime

web-pli-mintime (Minimum Time Interval for Web Side PLI Requests)			
Description	**Valid values**	**Default value**	**Takes effect**
The minimum time period, in milliseconds, between two RTCP PLI feedback messages (RFC4585) that can be sent on a Web-side video leg. If this value is 0, PLI transmission is disabled. The actual time between two PLI messages depends on various things: if web-pli-always is true, one message will be sent every web-pli-mintime milliseconds. Otherwise, if transcoding is on, a message will be sent when the number of lost packets during web-pli-mintime exceed a	The parameter must be an integer.	1000	At start or restart

specific threshold.			

web-rtcp-mux

web-rtcp-mux (RTCP-Mux on the Web-side)

Description	Valid values	Default value	Takes effect
Set this to true to enable rtcp-mux on the ROAP side, as per RFC 5761. When enabled, it will be signalled in an SDP offer sent by the gateway, and it will be accepted from an inbound SDP offer. If both sides agree, then the same port will be used for both RTP and RTCP. Set this to false if rtcp-mux is not to be used. Note: If web-rtcp-mux is false, then web-media-bundle cannot be true, as it would not make sense.	true, false	true	At start or restart

web-rtp-max-port

web-rtp-max-port (ROAP-side Max RTP Port)

Description	Valid values	Default value	Takes effect
Maximum UDP port value for ICE (ROAP-side RTP connection).		36999	At start or restart

web-rtp-min-port

web-rtp-min-port (ROAP-side Min RTP Port)

Description	Valid values	Default value	Takes effect
Minimum UDP port value for ICE (ROAP-side RTP connection).		36000	At start or restart

snmp

SNMP settings.

timeout

timeout (SNMP Task Timeout)

Description	Valid values	Default value	Takes effect
The maximum amount of time that SNMP can wait for a new task. This value is specified in milliseconds.	The parameter must be an integer value greater than zero.	100	At start or restart

Hardware Sizing Information

Hardware Sizing and Performance Information

Network Sizing Guidelines

The Genesys WebRTC implementation uses the G.711 and VP8 codecs:

- The G.711 codec is used for audio and requires 64kbps of bandwidth in each direction (incoming and outgoing).
- The VP8 codec is used for video encoding. The bitrate requirement depends on the quality of the streams, starting with a minimum of 100kbps and going up to 2000kbps or more for a single-party HD call. More detail is provided in the following table, which lists bandwidth requirements in kilobits per second.

Video Resolution	Gateway		Browser	
Incoming	**Outgoing**	**Incoming**	**Outgoing**	
SD	256	256	128	128
HQ	512	512	256	256
ED	1024	1024	512	512
HD	2048	2048	1024	1024

Performance Testing Scenarios

Genesys performed load testing for the following hardware and software platforms to create the sizing guidelines for Genesys WebRTC Service 8.5.2.

Important
VGA video resolution was used for this testing.

Performance Testing Configuration

	Linux Virtual Machine	Microsoft Windows Virtual Machine
OS	Red Hat Linux Enterprise Server v6.3x86_64 Kernel 2.6.32-279	Windows Server 2008 R2 Enterprise x64
Processor Type	Intel® Xeon® CPU X5675; 2 vCPUs; hyper-threading disabled	Intel® Xeon® CPU X5675; 2 vCPUs; hyper-threading disabled
Speed	3.06 GHZ	3.06 GHZ

	Linux Virtual Machine	Microsoft Windows Virtual Machine
Memory Size (RAM)	5 GB	5 GB
Hard Disk Space	35 GB	35 GB

Important

Genesys recommends Red Hat Enterprise Linux as the preferred platform for Genesys WebRTC Service.

Performance Test Results

Performance testing was conducted using VGA video resolution.

Important

Due to a Windows memory leak and due to the fact that the Windows version of the Genesys WebRTC Service runs as a 32-bit process (running in compatibility mode for 64-bit Windows), most of the performance testing was done with Red Hat Enterprise Linux. Windows testing is still ongoing, so there are no conclusive test results at this time, but Genesys expects that your results with Windows should be slightly lower than the Linux results and recommends that you limit traffic to 110 simultaneous calls.

Linux Performance Test Results

Description	Max Concurrent Calls	CAPS
Browser-to-browser audio+video without Xcoding	120	1.65
Browser-to-browser audio without Xcoding	190	2.47
Browser-to-SIP endpoint audio+video with Xcoding	190	2.41
Browser-to-SIP endpoint audio with Xcoding	250	3.06
Browser-to-browser audio+video SRTP without Xcoding	120	1.65
Browser-to-SIP endpoint audio+video SRTP without Xcoding	140	1.85
Multiple instances of browser-to-browser audio+video without	240	3.28

Description	Max Concurrent Calls	CAPS
Xcoding (Tested with 2 instances. Since larger configurations have not been tested, Genesys recommends scaling horizontally by adding new hosts.)		