Oracle® Communications WebRTC Session Controller

Concepts

Release 7.2

E69508-01

May 2016

ORACLE®

Oracle Communications WebRTC Session Controller Concepts, Release 7.2

E69508-01

Contents

Preface

This document provides a technical overview of Oracle Communications WebRTC Session Controller, including its features, architecture, deployment, and supported configurations.

Audience

This document is intended for anyone who needs to learn about WebRTC Session Controller, especially those who configure and maintain it.

Related Documents

For more information, see the following documents in the Oracle Communications WebRTC Session Controller documentation set:

- *Oracle Communications WebRTC Session Controller Extension Developer's Guide*
- *Oracle Communications WebRTC Session Controller System Administrator's Guide*
- *Oracle Communications WebRTC Session Controller Security Guide*
- *Oracle Communications WebRTC Session Controller Application Developer's Guide*
- *Oracle Communications WebRTC Session Controller Configuration API Reference*
- *Oracle Communications WebRTC Session Controller JavaScript API Reference*
- *Oracle Communications WebRTC Session Controller Android API Reference*
- *Oracle Communications WebRTC Session Controller iOS API Reference*

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc.

Access to Oracle Support

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info or visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs if you are hearing impaired.

1

Overview of WebRTC Session Controller

This chapter provides a technical overview of Oracle Communications WebRTC Session Controller, including its main features and topology.

About WebRTC Session Controller

WebRTC Session Controller enables real-time communication between web browsers and native mobile applications that support it and the following endpoints:

- Another web browser that supports WebRTC Session Controller
- A Session Initiation Protocol (SIP) phone, which uses Internet Protocol (IP)
- A public switched telephone network (PTSN) phone, such as a home phone or a cell phone

WebRTC Session Controller uses WebRTC (Web Real-Time Communication), an API being standardized by the World Wide Web Consortium (W3C). WebRTC enables web browsers and native mobile applications to directly share video, audio, and data.

WebRTC Session Controller provides developers and system administrators with an integrated application platform for creating and deploying converged WebRTC applications. Using the WebRTC Session Controller API, web application developers can incorporate functions such as video chats, audio communications, and data transfers into their web applications. WebRTC Session Controller manages the signaling functions of setting up and tearing down sessions.

WebRTC Session Controller is packaged and installed together with Oracle Communications Converged Application Server. WebRTC Session Controller has its own administration GUI, separate from the WebLogic Server GUI.

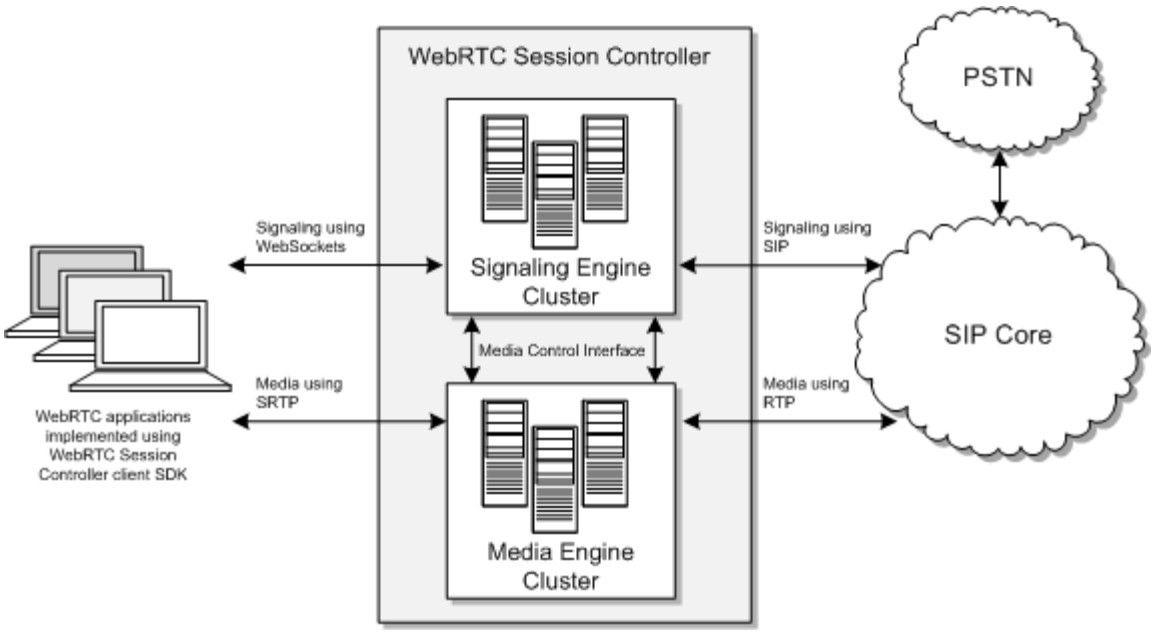
WebRTC Session Controller is a gateway server, usually placed at the edge of the network, that facilitates integration between WebRTC browsers, mobile applications and the enterprise SIP core. It reuses the Oracle WebLogic Server infrastructure already provided by Converged Application Server, including WebLogic Server domains and servers.

Web applications communicate with WebRTC Session Controller using a WebRTC Session Controller protocol (JSONRTC) based on the JavaScript Object Notation (JSON) data format. Communication occurs over WebSocket connections. WebRTC Session Controller translates the JSONRTC protocol to a telecom network protocol such as SIP.

Figure 1–1 shows a typical WebRTC Session Controller installation. The WebRTC Session Controller Signaling Engine cluster routes SIP messages from the SIP core to WebRTC applications using the WebSocket protocol. The WebRTC Session Controller Media Engine cluster routes Real-Time Transport Protocol (RTP) media messages from

the SIP core to WebRTC applications using Secure Real-time Transport Protocol (SRTP). The WebRTC applications are implemented using the JavaScript API library. The SIP core can also communicate with PSTN networks that host legacy telecom equipment.

Figure 1–1 WebRTC Session Controller Topology



WebRTC Session Controller Components

WebRTC Session Controller consists of the following components:

- **WebRTC Session Controller Signaling Engine**

  Signaling Engine sets up and tears down calls between WebRTC enabled web browsers and native mobile applications and your SIP core. It does this by translating messages from JSON to SIP and from SIP to JSON.

- **WebRTC Session Controller Media Engine**

  Media Engine manages the media stream of the multimedia calls that Signaling Engine sets up.

- **WebRTC Session Controller Media Engine administration console**

  You use the Media Engine administration console to configure the various Media Engine objects and options. There is both a command line interface as well as a web-based administration interface.

- **WebRTC Session Controller applications**

  For each WebRTC enabled application that you create, you set up a corresponding WebRTC Session Controller application. You configure the WebRTC Session Controller client application to translate the messages that your WebRTC enabled application uses from the JSON data format to SIP and from SIP to JSON.

- **WebRTC Session Controller Console**

You use the WebRTC Session Controller console to configure Signaling Engine and Media Engine and to create WebRTC Session Controller applications.

- **WebLogic Server Administration Console**

  WebRTC Session Controller extends the WebLogic Server Administration Console with an interface for configuring WebRTC Session Controller domains, clusters, servers, and the environment.

About the Web Signaling Protocols

WebRTC Session Controller translates messages between the browser-based client applications and network elements. Browser-based client applications control the features of the services running on the network by sending and receiving messages to and from WebRTC Session Controller over the WebSocket protocol. WebRTC Session Controller supports JSONRTC messages over WebSocket connections for communication with the WebRTC based clients.

WebRTC Session Controller supports a signaling protocol, mapping the signaling message and routing the messages based on routing profiles. WebRTC Session Controller also manages the protocol message state and ensures state replication for reliable message exchange.

For more information about web signaling protocols, see *WebRTC Session Controller Extension Developer's Guide* and *WebRTC Session Controller Application Developer's Guide.*

About Client Connectivity

WebRTC Session Controller increases web communication reliability through application rehydration, which is the process of recovering disconnected sessions and reestablishing connections when mobile or internet connectivity is interrupted. When sessions are reconnected they are populated with the previous session's information, such as customer name, balances, and so on.

In the browser environment, a user may reload the web page at any time. When the web page is reloaded, the session is rehydrated and the session state is recreated and synchronized. The session connectivity is also maintained across networks (for example, Ethernet to Wi-fi and Wi-fi to 3G) and across WebRTC-supporting devices (for example, desktops, mobile phones, and TVs).

For more information about creating Web applications, see *WebRTC Session Controller Application Developer's Guide.*

About Multitenancy Support

WebRTC Session Controller supports multi-tenant Software as a service (SaaS) applications. An in-premise installation of WebRTC Session Controller enables multiple departments of a customer to use WebRTC Session Controller for their own specific applications. The main benefits of multi-tenancy in WebRTC Session Controller are increased density, tenant isolation, and simplified configuration and management.

A tenant represents a configuration scope for a customer or a department that is authenticated to use its services. Multiple tenants can access a single WebRTC Session Controller installation or it can be configured as a single tenant installation. A tenant is associated with a tenant key that allows you to manage and track all usage of the installation by that customer or department. Every user account that is authenticated to access WebRTC Session Controller is associated with a tenant.

For more information on WebRTC Session Controller multitenancy, see "Configuring WebRTC Session Controller" in *WebRTC Session Controller System Administrator's Guide*.

About Statistics and Resource Limits

WebRTC Session Controller tracks a variety of statistics for use in license monitoring, performance monitoring, and controlling resource limits. In addition, a wide variety of statistics counters are available for more detailed system monitoring. For more information, see "Monitoring Statistics and Resource Limits" in *WebRTC Session Controller System Administrator's Guide*.

About Policy Integration

WebRTC Session Controller interacts with your Policy Control and Charging Rules Function (PCRF) to affect multimedia calls both before and after the media session transpires. You can use these Diameter Rx messages to confirm that subscribers are entitled to the services they are requesting, update their profile, or take other actions that your implementation requires. These actions can be requested by your Policy Control Enforcement Function (PCEF) or by WebRTC Session Controller itself.

About the WebRTC Session Controller APIs

WebRTC Session Controller includes the following API libraries:

- **JavaScript**: The JavaScript API library lets you create WebRTC-enabled Web applications that communicate using WebRTC Session Controller. Using the JavaScript API, you can create applications providing the following functionality:
  - Bidirectional voice and video calling
  - Chat and Data transfer
  - Message Session Relay Protocol (MSRP) chat based on the Rich Client Services (RCS) specification
  - File transfer based on the RCS specification
  - Simple messaging
  - Message notification
  - Endpoint capabilities exchange based on the RCS specification

  For more information, see *WebRTC Session Controller JavaScript API Reference* and *WebRTC Session Controller Application Developer's Guide*.
- **Adobe Flash JavaScript Extension**: WebRTC Session Controller provides a JavaScript extension library that lets you create applications which will support WebRTC audio and video calls in Microsoft Internet Explorer. For more information, see *WebRTC Session Controller JavaScript API Reference*, and "Creating WebRTC Session Controller Applications Compatible with Internet Explorer" in *WebRTC Session Controller Application Developer's Guide*.
- **Android**: The Android API library lets you create WebRTC-enabled applications that run on Android devices. Using this library, you can create applications with bidirectional audio and video calling features, including automatic session reconnection and call upgrade/downgrade capability, and messaging/data channel support. For more information, see *WebRTC Session Controller Android API Reference* and "Developing WebRTC-enabled Android Applications" in *WebRTC Session Controller Application Developer's Guide*.

- **iOS**: The iOS API library lets you create WebRTC-enabled applications that run on iOS devices. Using this library, you can create applications with bidirectional audio and video calling features, including automatic session reconnection and call upgrade/downgrade capability, and messaging/data channel support. For more information, see *WebRTC Session Controller iOS API Reference* and "Developing WebRTC-enabled iOS Applications" in *WebRTC Session Controller Application Developer's Guide*.

About Extending WebRTC Session Controller Functionality

You can extend the JSONRTC protocol by customizing existing API methods and parameters, as well as adding new ones. Extending the JSONRTC protocol enables you to add application-specific actions. Applications take action based on the state of a message. Whether a state triggers an action depends on certain conditions that must be met. You can modify the default conditions or add your own conditions. In addition, WebRTC Session Controller applications support Representational State Transfer (REST) callout functionality to provide easy integration with other network devices such as third party load balancers and media servers.

For more information about extending WebRTC Session Controller functionality, see *WebRTC Session Controller Extension Developer's Guide.*

About Developing WebRTC Enabled Applications

Developers can use the WebRTC Session Controller JavaScript API library to embed real-time communication functions into the WebRTC applications that they create. In addition, they can use the iOS and Android API libraries to provide real-time communications support in their iOS and Android applications.

The WebRTC Session Controller API libraries work with the WebRTC API to manage handshakes, establish and broker connections, and tear down sessions and subsessions over WebSocket, among other tasks. The library provides a set of API packages (such as **call**, **chat**, **notification**, **file transfer**, and so on, depending upon the platform) that provide specific real-time communication functions. The WebRTC Session Controller APIs are extensible and map to the underlying JSONRTC based protocol.

The WebRTC Session Controller JavaScript library is fully integrated with browsers supporting the WebRTC API (currently Google Chrome, Mozilla Firefox, and Opera) on any platform (for example, desktop, mobile, and native platforms such as Google Android). Likewise the iOS and Android API libraries are fully integrated with their respective platforms, providing a seamless WebRTC experience.

Additionally, the WebRTC Session Controller Media Engine provides Network Address Translation (NAT) traversal using Interactive Connectivity Establishment (ICE) which itself uses a Traversal Using Relays around NAT (TURN) implementation, allowing Web clients on private networks behind firewalls to successfully connect to one another. The WebRTC Session Controller APIs also support enabling Trickle-ICE negotiation, allowing a web client to exchange host ICE candidates incrementally to improve application performance.

For more information about using the WebRTC Session Controller API libraries to develop WebRTC enabled applications, see *WebRTC Session Controller Application Developer's Guide*.

WebRTC Session Controller Architecture

A WebRTC Session Controller cluster consists of multiple server instances running simultaneously and working together to provide reliability. From the client's perspective, a cluster is a single WebRTC Session Controller instance. The server instances that constitute a cluster can be located on the same machine or on different machines. You can increase a cluster's capacity by adding more server instances to the cluster on an existing machine, or by adding more machines to the cluster to host one or more server instances, as long as each server instance is running the same version of WebRTC Session Controller. Clustered server instances behave similarly to non-clustered instances, except that they provide fail over support.

WebRTC Session Controller Scalability

The capacity of an application deployed on a WebLogic Server cluster can be increased dynamically to meet demand. You can add server instances to a cluster without interrupting services; the application continues to run without impacting clients and subscribers. Because signaling and media messaging are separated, they can be scaled separately.

WebRTC Session Controller High Availability

WebRTC Session Controller Signaling Engine uses high-availability architecture to manage concurrent sessions. In a WebLogic Server cluster, application processing can continue when a server instance fails. You can increase the reliability and availability of your applications by deploying application components on multiple server instances in a dedicated cluster, as well as deploying multiple SIP data tier servers (replicas) in a dedicated SIP data tier cluster. If a server instance on which a component is running fails, another server instance on which that component is deployed can continue application processing.

For more information about high availability, see *WebRTC Session Controller System Administrator's Guide.*

About the Lightweight Proxy Registrar

The Lightweight Proxy Registrar performs certain functions provided by the SIP Proxy Registrar to reduce overall processing cost. A proxy server is primarily a router that acts on behalf of a client by sending requests to an entity that is closer to the requests' destination. A registrar stores information that it receives from the REGISTER requests.

The Lightweight Proxy Registrar includes the following components:

- Lightweight Registrar
- Lightweight Proxy
- Location Service
- Custom Application Router

For more information, see the topic about the Lightweight Proxy Registrar in *WebRTC Session Controller System Administrator's Guide*.

About Security and Authentication

WebRTC Session Controller relies on and benefits from the security features of the underlying WebLogic Server platform, including security realms, security monitoring features, and more. WebRTC Session Controller also has additional security features.

Unlike most conventional real-time systems (for example, SIP-based soft phones), WebRTC communications are directly controlled by web servers. In addition to the authentication methods WebLogic Server provides (HTTP basic, form-based, and client-certificate), WebRTC Session Controller supports an OAuth Identity Asserter, a HTTP authentication provider, and two-way SSL.

Security for WebRTC communications requires that the communicating endpoints be able to authenticate each other. While these endpoints are making calls through the signaling services, their identities are authenticated using Identity Provider (IdP), which supports OAuth, Facebook Connect and so on. WebRTC Session Controller can validate the caller's identity that is in the request to access the WebRTC-enabled client application. The identity may be a generic URI or an email address. WebRTC Session Controller maps the identity to the form identity (telco or IMS identity) of the outbound call.

WebRTC Session Controller also offers denial-of-service (DoS) protection (against message floods, malformed requests, and more) at the signaling and media levels.

For more information about authentication and security, see *WebRTC Session Controller Security Guide*.

About the WebRTC Session Controller Console

The WebRTC Session Controller console is a web browser-based, graphical user interface that you use to manage the WebRTC Session Controller environment.

You use the WebRTC Session Controller console to:

- Create, manage, and use the components that translate messages from SIP to JSON and JSON to SIP, which are required to build up and tear down multimedia calls between your client application and your SIP core
- Add additional functionality that your implementation requires at any point in the translation process
- Configure Signaling Engine properties, including time limit parameters for SIP sessions and WebSocket connections
- Configure Media Engine nodes, including media hosts that you use with WebRTC Session Controller
- Manage WebRTC applications, packages, criteria, and translation scripts

See *WebRTC Session Controller Extension Developer's Guide* for more information about using the WebRTC Session Controller console to manage translate messages. See *WebRTC Session Controller System Administrator's Guide* for more information about using the WebRTC Session Controller console to configure Signaling Engine and Media Engine.

Supported Media Handling Standards

WebRTC Session Controller provides interfaces that conform with the 3GPP Policy and charging control architecture. Signaling and media policy rules are enforced for bandwidth management, media channel allocation, and quality of service (QoS).

WebRTC Session Controller supports the following media handling standards and specifications:

- **Network Address Translation (NAT) traversal**

  WebRTC Session Controller supports Interactive Connectivity Establishment (ICE) (RFC 5245), Session Traversal Utilities for NAT (STUN) (RFC 5389), and Traversal Using Relays around NAT (TURN) (RFC 5766).

- **Media encryption and decryption**

  WebRTC Session Controller interoperates with clients by implementing media transport mechanisms defined in RFC 3711 (SRTP), RFC 4568 (SDP), SRTP/DTLS, and RFC 5124 (RTP SAVPF profile).

- **Codecs**

  WebRTC Session Controller supports the transit, reordering, removal, and insertion of all audio codecs (Opus, G.711) and video codecs (VP8/9, H.264) that would be used in a WebRTC application. Audio transcoding of the following codecs is supported on the WSC Media Engine: AMR, AMR-WB, G711a/u, G722-1, G723, G726-16, G726-24, G726-32, G726-40, G728, G729, GSM, ILBC, Opus, SILK, and Speex. Transcoding of other audio and video codecs can be performed on the WSC via third-party media resource function (MRF) elements.

- **Interworking function**s

  WebRTC Session Controller supports the interworking function that WebRTC applications use to communicate with deployed SIP/RTP-based voice-over-IP and video-over-IP devices, and PSTN devices in the signaling and media domains. WebRTC Session Controller supports DTLS/SRTP termination, ICE termination, and RTP/RTCP stream multiplexing.