Oracle® Communications WebRTC Session Controller

Concepts

Release 7.0

E40976-01

November 2013

ORACLE®

Oracle Communications WebRTC Session Controller Concepts, Release 7.0

E40976-01

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

If this is software or related documentation that is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, the following notice is applicable:

U.S. GOVERNMENT END USERS: Oracle programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, delivered to U.S. Government end users are "commercial computer software" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, use, duplication, disclosure, modification, and adaptation of the programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, shall be subject to license terms and license restrictions applicable to the programs. No other rights are granted to the U.S. Government.

This software or hardware is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications that may create a risk of personal injury. If you use this software or hardware in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure its safe use. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software or hardware in dangerous applications.

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Xeon are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Opteron, the AMD logo, and the AMD Opteron logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group.

This software or hardware and documentation may provide access to or information on content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services.

Contents

Preface v

Audience v
Documentation Accessibility v

1 Overview of WebRTC Session Controller

About WebRTC Session Controller 1-1
WebRTC Session Controller Components 1-2
About the Web Signaling Protocols 1-3
About Client Connectivity 1-3
About Policy Integration 1-3
About the WebRTC Session Controller API 1-3
About Extending WebRTC Session Controller Functionality 1-4
About Developing WebRTC Enabled Applications 1-4
WebRTC Session Controller Architecture 1-4
WebRTC Session Controller Scalability 1-4
WebRTC Session Controller High Availability 1-5
About Security and Authentication 1-5
About the WebRTC Session Controller Console 1-5
Supported Media Handling Standards 1-6

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.

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 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 that support it, as well as SIP and public switched telephone network (PSTN) phones. WebRTC Session Controller uses WebRTC (Web Real Time Communication), an API being standardized by the World Wide Web Consortium (W3C). WebRTC enables web browsers to directly share video, audio, and data.

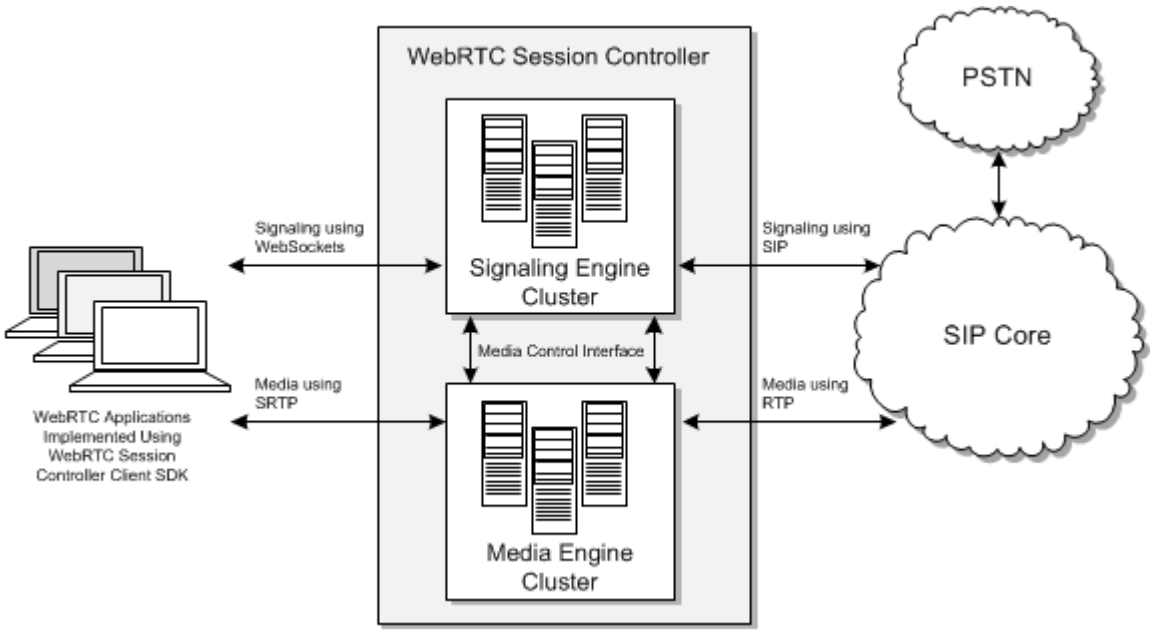
WebRTC Session Controller is packaged and installed together with Oracle Converged Application Server. WebRTC Session Controller has its own Administration Server GUI separate from the Oracle Converged Application Server GUI.

WebRTC Session Controller is a gateway server, usually placed at the edge of the network to facilitate integration between WebRTC browser clients with the IP Multimedia Subsystem (IMS). It reuses the Oracle WebLogic Server infrastructure already provided by Converged Application Server, including WebLogic Server domains and servers.

In the WebRTC Session Controller architecture, web applications communicate with WebRTC Session Controller using an Oracle-produced protocol (JsonRTC) based on the JSON data format and using 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 the WebRTC applications using WebSockets. The WebRTC Session Controller Media Engine cluster routes media from the SIP core using Real-time Transport Protocol (RTP) to the 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 hosting legacy telecom equipment.

Figure 1–1 WebRTC Session Controller Topology



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 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 your IMS core. It does this by translating JSON messages to SIP and back.

- WebRTC Session Controller Media Engine

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

- 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 the 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 Web Application Developer's Guide.*

About Client Connectivity

WebRTC Session Controller increases web communications reliability through application rehydration, which is the process of recovering disconnected sessions and re-establishing 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 handling events in the application environment, see the *Oracle Communications WebRTC Session Controller Web Application Developer'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 the subscriber is 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 from inside WebRTC Session Controller itself.

About the WebRTC Session Controller API

WebRTC Session Controller includes a JavaScript API library for extending the WebRTC Session Controller functionality and for enabling your WebRTC based applications to communicate with WebRTC Session Controller.

For information about the WebRTC Session Controller JavaScript API library, see *Oracle Communications WebRTC Session Controller JavaScript API Reference.*

About Extending WebRTC Session Controller Functionality

You can extend the JsonRTC protocol by customizing existing methods and parameters, as well as adding new ones. Extending the JsonRTC protocol allows you to add application specific actions. Applications take actions 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 to trigger various actions based on the message state.

For more information about extending WebRTC Session Controller functionality, see the *Oracle Communications 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.

The WebRTC Session Controller JavaScript API library works 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**, and so on) that provide the functions for including communication features into the WebRTC application. The WebRTC Session Controller JavaScript 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 Chrome and Firefox) on any platform (for example, desktop, mobile, and native platforms such as Android).

For more information about the WebRTC Session Controller JavaScript API library, see *Oracle Communications WebRTC Session Controller JavaScript API Reference.* For more information about using the WebRTC Session Controller JavaScript API library to develop WebRTC enabled applications, see *Oracle Communications WebRTC Session Controller Web Application Developer's Guide*.

WebRTC Session Controller Architecture

A WebRTC Session Controller cluster consists of multiple WebRTC Session Controller 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 run on the same machine or be located on different machines. You can increase a cluster's capacity by adding additional server instances to the cluster on an existing machine, or by adding additional machines to the cluster to host one or more server instances. Each server instance in a cluster must run the same version of WebRTC Session Controller. Clustered WebRTC Session Controller server instances behave similarly to non-clustered instances, except that they provide failover and load balancing. The WebRTC Session Controller cluster is fronted by a load balancer for both HTTP and SIP traffic.

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 impact to clients and end users. 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 cluster application components by deploying them on multiple server instances in the 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. You can increase the reliability and availability of your applications by using multiple servers in a dedicated cluster, as well as multiple SIP data tier servers (replicas) in a dedicated SIP data tier cluster.

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 has additional security features in place as well.

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 end points are making calls through the signaling services, their identities are authenticated using IDP (that supports OAuth, Facebook Connect etc). 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 *Oracle Communications WebRTC Session Controller Security Guide.*

About the WebRTC Session Controller Console

The WebRTC Session Controller administration 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 IMS core. You can use the console to add additional functionality that your implementation requires at any point in the translation processing.

You use the WebRTC Session Controller console for configuring Signaling Engine properties and WebRTC Session Controller Media Engine nodes, and for managing WebRTC applications, packages, criteria, and translation scripts. You use this console to create criteria, the Groovy language based scripts that select messages to translate

between JSON and SIP and perform the translation. Criteria serve as a platform for you to extend translation processing with functionality that your implementation requires. Signaling Engine configuration includes time limit parameters for SIP sessions and WebSocket connections. Media Engine entries represent media hosts that you use with WebRTC Session Controller.

See *WebRTC Session Controller Extension Developer's Guide* for more information about using the WebRTC Session Controller console to manage managing applications, packages, criteria, and translation scripts. 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 of all audio codecs (Opus, G.711, and iSAC) and video codecs (VP8) that would be used in a WebRTC application. WebRTC Session Controller supports codec reordering, removal, and insertion. Audio transcoding of the following codecs is supported: AMR, AMR-WB, G711a/u, G722-1, G723, G726-16, G726-24, G726-32, G726-40, G728, G729, GSM, ILBC, SILK, and Speex.

- **Interworking function**

  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.