Oracle® Communications WebRTC Session Controller

Security Guide

Release 7.1

E55124-01

March 2015

ORACLE®

Oracle Communications WebRTC Session Controller Security Guide, Release 7.1

E55124-01

Contents

Preface v

Audience v
Related Documents v
Documentation Accessibility v

1 WebRTC Session Controller Security Overview

Basic Security Considerations 1-1
Overview of WebRTC Session Controller Security 1-1
Understanding the WebRTC Session Controller Environment 1-1

2 Performing a Secure WebRTC Session Controller Installation

Installing WebRTC Session Controller Securely 2-1
About Access to Files Created During Installation 2-2
About Password Policies 2-2
Post-Installation Configuration 2-2
Setting Up User Accounts to Lock and Expire 2-2
Enabling SSL for LDAP Authentication Providers 2-2

3 Implementing WebRTC Session Controller Security

About WebRTC Session Controller Security 3-1
Default and Optional Security Settings 3-1
Enabling TLS (SSL) 3-2
Handling Wildcard SSL Certificates 3-2
Client to WebRTC Session Controller Authentication 3-2
Form-Based Authentication 3-2
Basic Authentication 3-2
HTTP Authentication 3-3
Digest Authentication 3-3
OAuth Providers 3-3
Logging in with an OAuth Token 3-4
Logging into a REST Provider with a Token 3-4
Two-way SSL Authentication 3-4
Guest Access 3-5
Redirecting to a Different URL after Authentication 3-5
Internal Security 3-5

Securing Coherence 3-5
Securing Ports 3-6
Signaling and Media DoS Protection 3-6
WebRTC Session Controller to SIP Security 3-7
Securing SIP 3-7
Handling Challenges from the IMS Core 3-7

Preface

This document describes security features and configuration for Oracle Communications WebRTC Session Controller.

Audience

This document is intended for administrators who configure security for WebRTC Session Controller.

Related Documents

For more information, see the following documents:

- *Oracle Communications WebRTC Session Controller Concepts*
- *Oracle Communications WebRTC Session Controller Configuration API Reference*
- *Oracle Fusion Middleware Securing Oracle WebLogic Server* in the Oracle WebLogic Server 12c documentation
- *Oracle Fusion Middleware Application Security Guide* in the Oracle WebLogic Server 12c documentation

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

WebRTC Session Controller Security Overview

This chapter describes the Oracle Communications WebRTC Session Controller security features.

Basic Security Considerations

The following principles are fundamental to using any application securely:

- **Keep software up to date**. This includes the latest product release and any patches that apply to it.
- **Limit privileges as much as possible**. Users should be given only the access necessary to perform their work. User privileges should be reviewed periodically to determine relevance to current work requirements.
- **Monitor system activity**. Establish who should access which system components, and how often, and monitor those components.
- **Install software securely**. For example, use firewalls, secure protocols using TLS (SSL) and secure passwords.
- **Learn about and use the WebRTC Session Controller security.** See "Implementing WebRTC Session Controller Security" for more information.
- **Keep up to date on security information**. Oracle regularly issues security-related patch updates and security alerts. You must install all security patches as soon as possible. See the “Critical Patch Updates and Security Alerts” website:

  `http://www.oracle.com/technetwork/topics/security/alerts-086861.html`

Overview of WebRTC Session Controller Security

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.

This guide describes the security features of the WebRTC Session Controller. For WebLogic Server information, including information about implementing application security, see the Oracle WebLogic Server 11*g* documentation.

Understanding the WebRTC Session Controller Environment

When planning your WebRTC Session Controller implementation, consider the following:

- Which resources need to be protected?

  - You need to protect customer data, such as IP addresses.
  - You need to protect internal data, such as proprietary source code.
  - You need to protect system components from being disabled by external attacks or intentional system overloads.

- Who are you protecting data from?

  For example, you need to protect your subscribers' data from other subscribers, but someone in your organization might need to access that data to manage it. You can analyze your workflows to determine who needs access to the data; for example, it is possible that a system administrator can manage your system components without needing to access the system data.

- What will happen if protections on a strategic resource fails?

  In some cases, a fault in your security scheme is nothing more than an inconvenience. In other cases, a fault might cause great damage to you or your customers. Understanding the security ramifications of each resource will help you protect it properly.

2

Performing a Secure WebRTC Session Controller Installation

This chapter presents planning information for your Oracle Communications WebRTC Session Controller system and describes recommended deployment topologies that enhance security.

For more information about installing WebRTC Session Controller, see *Oracle Communications WebRTC Session Controller Installation Guide.*

Installing WebRTC Session Controller Securely

When installing WebRTC Session Controller, do the following when you create the WebLogic Server domain for WebRTC Session Controller:

- Disable all non-SSL ports to secure all communication between components, and JCA and JMS collection, over SSL ports.
- Make sure that SSL ports are being used on the administration server and all managed servers.
- If installing WebRTC Session Controller on a cluster of servers, configure the cluster addresses to use SSL ports.
- When creating a cluster, the following file should be copied from the administration server to the following location on each server in the cluster:

  WSC_HOME/**user_projects/domains/***DOMAIN_NAME*/**security/SerializedSystemIni.dat**
- After you have created the WebLogic Server domain for WebRTC Session Controller, start the administration server. Then, use t3s to start the managed servers:

  ```
  startManagerServer.sh ManagedServer_1 t3s://host_name
  ```

  where *ManagedServer_1* is the name of the first managed server, and *host_name* is the host name of the administration server.
- Using the WebLogic Administration Console, configure certificate identity and trust store to use SSL. Do not use the default, demonstration certificate that comes with WebLogic Server. See the WebLogic Server security and system administration documentation for more information.

About Access to Files Created During Installation

Access to files created during the installation is limited. The user who performs the installation will have write access to those files created during installation.

About Password Policies

Oracle recommends having strong password policies for WebRTC Session Controller. Consider enforcing the following password policies:

- Passwords should have a minimum of eight characters.
- Passwords must contain at least one digit, one capital letter, and one special character.
- The user name must not be part of the password.

Stricter rules can be set for the authentication provider using the WebLogic Administration Console. For details on authentication providers and their configuration, refer to the discussion on securing Oracle WebLogic Server in the WebLogic Server documentation.

See *Oracle Communications WebRTC Session Controller System Administrator's Guide* for information about changing and setting WebRTC Session Controller passwords.

Post-Installation Configuration

This section explains security configurations to complete after WebRTC Session Controller is installed.

Setting Up User Accounts to Lock and Expire

Create WebRTC Session Controller user accounts and configure them to lock after a certain number of failed login attempts, and to expire after a certain period of idle time.

See *Oracle Communications WebRTC Session Controller System Administrator's Guide* for information about changing and setting WebRTC Session Controller passwords.

Enabling SSL for LDAP Authentication Providers

For secure communication between WebLogic Server and an external LDAP, enable SSL on both the external LDAP authentication provider and the corresponding WebLogic Security Provider. SSL on the WebLogic security provider is enabled from the WebLogic Administration Console.

For information about secure communication between WebLogic Server and an external LDAP authentication provider, see *Oracle Fusion Middleware Securing Oracle WebLogic Server*.

3

Implementing WebRTC Session Controller Security

This chapter describes the specific security mechanisms provided by Oracle Communications WebRTC Session Controller.

About WebRTC Session Controller Security

WebRTC Session Controller is built upon the framework of WebLogic Server. In addition to the authentication methods WebLogic Server provides (HTTP basic, form-based, and client certificate), WebRTC Session Controller supports the following:

- Identity Asserter: This will validate tokens from OAuth providers (such as Facebook and Google). For information about configuring an OAuth access provider, see "About Provisioning WebRTC Session Controller OAuth Access" in the *WebRTC Session Controller System Administrator's Guide*.
- HTTP authentication provider: HTTP authentication validates the supplied user name and password against an HTTP endpoint such as a REST endpoint. For information about configuring an HTTP authentication provider, see "About Provisioning WebRTC Session Controller HTTP Access" in the *WebRTC Session Controller System Administrator's Guide*.

For information on configuring WebLogic guest access, see "About Provisioning WebRTC Session Controller Guest Access" in the *WebRTC Session Controller System Administrator's Guide*.

Authentication of the browser application occurs when the WebRTC Session Controller JavaScript library establishes a WebSocket connection with the server. Depending on the configuration, the server enforces an appropriate HTTP authentication (for example, form-based or OAuth). Once the authentication is successful, WebRTC Session Controller can obtain account credentials from the SIP network and communicate with the SIP network using the account credentials.

WebRTC Session Controller obtains credentials for authentication by establishing web to SIP identity mapping. WebRTC Session Controller can retrieve IMS credentials (public and private identities or passwords) for a user who has completed web authentication challenges directed at WebRTC Session Controller by the IMS network. Stored credentials may be needed in another node in the WebRTC Session Controller cluster as part of mapping JSON to SIP, for example.

Default and Optional Security Settings

By default, WebRTC Session Controller configuration settings prompt users for a user name and password, using the basic authentication method. JavaScript resources are

not protected. The application can be made less or more secure by enabling guest access or necessitating secure access through a TLS (SSL) port respectively.

Enabling TLS (SSL)

Web browsers can connect to WebRTC Session Controller over an HTTP port or an HTTP with a TLS (SSL) port. The benefits of using an HTTPS port versus an HTTP port are two-fold. With TLS (SSL) connections:

- All communication on the network between the web browser and the server is encrypted. Because it is encrypted, sensitive information will never be in clear text.
- As a minimum authentication requirement, the server is required to present a digital certificate to the web browser client to prove its identity.

You can enable TLS (SSL) by using the WebLogic Server Administration Console. See the discussion on configuring TLS in the chapter about configuring diameter client nodes and relay agents in *Oracle WebLogic Server SIP Container Administrator's Guide.*

Handling Wildcard SSL Certificates

Facebook and Google OAuth URLs present wildcard certificates. By default, WebRTC Session Controller does not allow these certificates. Therefore, it is necessary to override the default SSL settings so that wildcard certificates can be handled. For more information, see the discussion on handling wildcard SSL certificates in *Oracle Communications WebRTC Session Controller System Administrator's Guide.*

Client to WebRTC Session Controller Authentication

To secure client to WebRTC Session Controller interactions, you can use one of the following methods described in this section.

Form-Based Authentication

Form-based authentication uses a custom login and error windows that you create. A client requests access to a protected resource. If the client is unauthenticated, the server redirects the client to a login page. The client then submits the login form to the server. If the login succeeds, the server redirects the client to the resource. If the login fails, the client is redirected to an error page.

The benefit of form-based login is that you have complete control over login and error screens so that you can design them to meet the requirements of your application or enterprise policy. However, WebRTC Session Controller does not offer form-based authentication by default. To enable form-based authentication, create a web application using form-based authentication methods. Deploy that web application on the WebLogic server. This allows users who authenticate with the WebLogic server to already be authenticated when they access the client application interface.

Form-based authentication is not particularly secure. In form-based authentication, the content of the user dialog box is sent as plain text, and the target server is not authenticated. This form of authentication can expose your user names and passwords unless all connections are over SSL. If someone can intercept the transmission, the user name and password information can easily be decoded.

Basic Authentication

Basic authentication is provided by default. Basic authentication uses HTTP headers to transmit the user name and password to WebRTC Session Controller. Basic

authentication is not recommended for production systems unless you can ensure that all connections between clients and the WebLogic SIP server instance are secure.

With basic authentication, a client requests access to a protected resource. The web server displays a login screen that requests the user name and password. The client then submits the user name and password to the server. The server validates the credentials and, if successful, returns the requested resource.

HTTP basic authentication is not particularly secure. Basic authentication sends user names and passwords over the Internet as text that is uu-encoded (Unix-to-Unix encoded) but not encrypted. This form of authentication, which uses Base64 encoding, can expose your user names and passwords unless all connections are over SSL. If someone can intercept the transmission, the user name and password information can easily be decoded.

HTTP Authentication

HTTP authentication validates the supplied user name and password against an HTTP endpoint (for example, a REST endpoint). The HTTP authentication provider will also support fetching the user's SIP identity information from the remote end point.

By default, WebRTC Session Controller has an HTTP authentication provider, but it needs to be added to the list of security providers in WebRTC Session Controller. The HTTP authentication provider is invoked whenever a user submits a user name and password to log in to WebRTC Session Controller (using the basic authentication dialog or a form-based login page). This provider sends a request to a configured web service end point with the user name and password in the basic authentication header. If the response from the HTTP end point is 200/OK, the authentication is considered successful. Any other response code indicates that the authentication failed.

For information about configuring HTTP authentication, see *Oracle Communications WebRTC Session Controller System Administrator's Guide.*

Digest Authentication

Digest authentication is not supported in WebRTC Session Controller. You can implement your own digest authentication provider by using a separate web application to authenticate users. After the login process is complete, requests can be made to the application that manages WebSocket connections. For more information, see the discussion on configuring digest authentication in *Oracle Fusion Middleware Securing Web Services and Managing Policies with Oracle Web Services Manager, Release 12c.*

OAuth Providers

OAuth 2.0 enables a third party application (such as WebRTC Session Controller) to obtain limited access to protected resources (such as the end user's email address, Facebook friends' list stored in resource servers such as Google and Facebook) with the end user's consent. An access token is given after a request for access is made, and is used to access the protected resources hosted by WebRTC Session Controller.

WebRTC Session Controller provides two components to help customers integrate their login mechanism with OAuth providers. These components are the WebRTC Session Controller Servlet Authenticator, and the WebRTC Session Controller OAuth Identity Asserter. Both these modules are installed by default, but they need to be added to the list of security providers in WebRTC Session Controller.

For information about configuring signaling engine parameters such as client ID, client secret, and OAuth server URL, see *Oracle Communications WebRTC Session Controller System Administrator's Guide.*

Logging in with an OAuth Token

A client application can implicitly authorize a user and provide an OAuth token. The client application can use the OAuth token to log in to WSC by passing the token in a query string to the configured OAuth login URL, for example: /login/google or /login/facebook.

WSC expects the OAuth token to be in the query string with a parameter name of oauth_token. The following example shows a sample of such a login URL, with carriage returns added for readability:

```
http://sasantha-e6420.us.abcxyz.com:7001/login/google?oauth_token=ya29.1.AADtN_W5_
Ir6Wb-Mcbhng0SRlRMpUukhumnuuYsLWniooOTe50Y9i1b77GpBEesgVA&final_redirect_
uri=http://sasantha-e6420.us.abcxyz.com:7001/wscsample/loginRedirect.html&wsc_app_
uri=/ws/webrtc/facebook
```

Note: The domain specified by `redirect_uri` must match a domain specified in the allowed domains configuration

Logging into a REST Provider with a Token

A REST security provider can login a user based on a token that is sent as an HTTP request parameter with the login request. The REST provider sends the parameter to a configured REST end point URL. The user is authenticated upon receipt of a success response and the HTTP response body, which is typically a JSON message, is stored as a **credential** in the authenticated **subject**. The group name that is associated with the REST provider configuration is stored as a **principal** in the subject. The subject is represented by the Java class `javax.security.auth.Subject` and the principal is an implementation of the interface `java.security.Principal`.

The following client login request, for example, includes a REST token, with carriage returns added for readability:

```
http://www.example.com/login?RestAccessAuthToken=myToken&wsc_app_
uri=/ws/webrtc/restauth&redirect_uri=http://www.example.com/call.html
```

Note: The domain specified by `redirect_uri` must match a domain specified in the allowed domains configuration

The client must put the RestAccessAuthToken parameter in the HTTP request with a valid value that the server will accept. WscRestAuthenticator provides the default token name RestAccessAuthToken; however, you can configure your own token. The REST server validates the token provided.

Two-way SSL Authentication

Two-way SSL is a more secure method of authentication than either basic or form-based authentication. It uses HTTP over SSL, in which the server and the client authenticate one another using public key certificates. SSL provides data encryption, server authentication, message integrity, and optional client authentication for a TCP/IP connection. You can think of a public key certificate as the digital equivalent

of a passport. It is issued by a trusted organization, which is called a certificate authority (CA), and provides identification for the bearer.

If you specify two-way SSL (client certificate) authentication, the web server will authenticate the client using the client's X.509 certificate, a public key certificate that conforms to a standard that is defined by X.509 Public Key Infrastructure (PKI). Before running an application that uses SSL, you must configure SSL support on the server and set up the public key certificate. For more information about configuring SSL, see *Oracle WebLogic Server 12c* documentation.

Guest Access

Anonymous guest access can be granted to any application in WebRTC Session Controller. When a user initiates guest access, a WebLogic Server servlet authentication filter inspects the request before the authentication providers are invoked. If the incoming request matches a WebRTC Session Controller application URL pattern (which is configured for insecure access), and if there are no other authorization headers in the request, then the servlet adds an authorization header. The request goes through the provider chain and the authentication grants the user guest access.

For more information about configuring guest access, see *Oracle Communications WebRTC Session Controller System Administrator's Guide.*

Redirecting to a Different URL after Authentication

When a user attempts to log in using a request URI that matches the pattern `/login/<any>`, you can redirect the browser to a different URL using Groovy. This enables you to perform a two-stage authentication as illustrated by the following scenario:

- The user logs in using any configured security provider such as an OAuth provider or a REST provider or a default WebLogic provider.
- After authenticating the user, the security provider sends the user a one-time access code through email or a text message (SMS). Generally, the security provider does this for a first-time user or a user who is using a device for the first time.
- The user is authenticated but must enter the one-time access code to login to WebRTC Session Controller. At this point, you must redirect the user to a secondary URL to enter the access code.

For information on enabling redirection after authentication, see "About Post Authentication Redirection" in the *WebRTC Session Controller System Administrator's Guide*.

Internal Security

You can strengthen internal security by securing Oracle Coherence and ports.

Securing Coherence

WebRTC Session Controller and its nodes use Oracle Coherence internally and enable the Coherence security framework by default. The security framework is enabled by checking the **Security Framework Enabled** checkbox through the WebLogic console. If you *do not* want the Coherence security framework enabled, see "Enabling the Oracle

Coherence Security Framework" in *Securing Oracle Coherence* and *uncheck* the **Security Framework Enabled** checkbox.

Coherence security includes securing both cluster members and extend clients. You enable security as required, based on your application or cluster implementation and your organization's security concerns and security tolerances.

For a brief discussion of each security feature, see *Oracle Coherence Security Guide.*

Securing Ports

Configure firewalls to restrict access internally. Oracle recommends that port 7001 on the managed servers be disabled and 7002 over SSL be used instead. Enabling 7002 can be done during domain installation; however you must remove the non-SSL port by using the WebOracle Communications WebRTC Session ControllerLogic Server Administration Console. For information about configuring port 7002, see *Oracle Communications WebRTC Session Controller Configuration API Reference*.

Table 3–1 lists the default ports, their names, and security considerations:

Table 3–1 WebRTC Session Controller Default Ports

Value	Description	Security
7001	The administration HTTP port	Allow this port external access for the managed servers, but not for the administration server. It is recommended that you disable this port and use an SSL port instead.
8088	The Coherence port	Restrict access from outside of the WSC network for this port.
5060	The SIP port	Allow access to the IMS network from this port.
5061	The SIPS port	Allow access to the IMS network from this port.
4057	WebRTC Session Controller Media Engine HTTP callback port	Restrict access to this port except between WebRTC Session Controller Media Engine and WebRTC Session Controller Signaling Engine.
7002	SSL port for admin/http/t3	7002 over SSL is recommended over 7001. Enable 7002 during domain installation. Remove the non-SSL port by using the WebLogic Server Administration Console.

Signaling and Media DoS Protection

WebRTC Session Controller offers denial of service (DoS) protection (against message floods, malformed requests, and so on) at signaling and media levels.

Media DoS protection includes the following:

- Pools of media ports are provisioned by port ranges on each IP interface (thus giving operators full control over the available port range).
- All media ports are closed when inactive.
- When a signaling (SIP or web-associated) session requires media ports, they are allocated from the appropriate pool.
- The allocated media ports are owned by and dedicated to the signaling session.
- Media ports attach to a specific remote peer address and are closed to other IP addresses.

- When Secure Real-Time Transport Protocol (SRTP) is used (for example, with WebRTC media), additional authentication steps are used to verify the authentication trailer for each SRTP packet.
- When released, media ports are returned to the pool and are monitored to ensure that the media ports are fully quiesced (any inactive media ports that continue to receive media traffic are put into quarantine until fully quiesced.)
- Media can be monitored to ensure that the actual received codec matches the signaled codec.
- Real-time Transport Protocol (RTP) sequence integrity can be checked and invalid packets can be discarded.

  On the signaling side, WebRTC Session Controller DoS protection applies to all types of WebRTC signaling methods. Among other measures, you can set the limits for maximum incoming message size, complete message timeout period, and number of file descriptors to help prevent denial-of-service attacks. The separation of signaling and media contributes to the overall security because a DoS attack on the signaling side does not affect the media side, and vice versa.

For more information about DoS protection, maximum incoming message size, and complete message timeout period, see the discussion on reducing the potential for denial of service attacks in *Oracle WebLogic Server 12c* documentation.

WebRTC Session Controller to SIP Security

Other ways of improving security for WebRTC Session Controller include securing SIP and handling challenges from the IMS Core.

Securing SIP

WebRTC Session Controller offers secure SIP (SIPS) connections, using TLS to secure signalling. WebRTC Session Controller also uses two-way SSL to verify the digital certificate supplied by the client. You must ensure that a SIPS transport (SSL) has been configured in order to use client-certificate authentication. For more information about configuring SSL, see *Oracle Database Advanced Security Administrator's Guide.*

Handling Challenges from the IMS Core

When you receive a challenge from the IMS core, you can handle it in the client (on the home page) because challenges are propagated to the client side. WebRTC Session Controller propagates the challenge to the client side only if there is no private identity and private password in the security context. No challenges will be made to WebRTC Session Controller if you set up the IMS network so that WebRTC Session Controller is seen as a trusted entity.

You write your own WebRTC Session Controller authentication module that handles the web user authentication. This authentication module can be a Groovy script that sets up a security context to be used when receiving challenges. The Groovy script fetches the IP Multimedia Public Identity (IMPU) or IP Multimedia Private Identity (IMPI) and the secret key information, which is needed to fill in the SecurityContext object, and adds that information to the public credential set of the security context subject. When a user logs in to the web application page, this Groovy script is invoked and is passed the authenticated subject. The authentication provider populates credential information (web ID, SIP public ID, SIP private ID, SIP private password, and so on) into the authenticated subject. This security context is used towards the IMS

network. The Groovy script is used to build the security context information from the authenticated subject. It then fetches the IMPU, IMPI, and secret key information.

Alternatively, you can configure the P-Asserted-Identity header in the groovy script. For example, if the login identity of the web application user is configured as the subscriber's IMPU, then the IMPU can be used in the P-Asserted-Identity header.