Oracle® Communications WebRTC Session Controller

Security Guide

Release 7.2

E69516-01

May 2016

ORACLE®

Oracle Communications WebRTC Session Controller Security Guide, Release 7.2

E69516-01

Contents

4 Deploying WebRTC Session Controller in a Demilitarized Zone

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 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

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 *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 *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 *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 *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.

Note: If authentication is successful, and if the response body returned by the remote HTTP endpoint is valid JSON formatted data, WebRTC Session Controller normalizes the JSON data as a Java Map and embeds this normalized data as credential information in the authenticated subject. That credential information is accessible in the groovy layer, enabling you to use it to build a credential map for the SIP Register request.

For information about configuring HTTP authentication, see *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 *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 *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 *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.

4

Deploying WebRTC Session Controller in a Demilitarized Zone

This chapter explains how to deploy Oracle Communications WebRTC Session Controller in an semi-secured environment. This chapter refers to this type of deployment as a *demilitarized zone* (DMZ) *deployment*.

Overview and Recommended Configurations

A WebRTC Session Controller DMZ deployment should include multiple networks configured for access on separate networks cards. Access points on each host shield back-end systems such as administration servers, media servers, and the Session Initiation Protocol (SIP) core from DMZ/Internet traffic. In particular, consider hosting WebRTC Session Controller administration servers on a separate network to isolate administration traffic from application traffic.

If your WebRTC Session Controller installation must be deployed in the DMZ, Oracle recommends that you use one of the multi-tier WebRTC Session Controller implementations shown in Figure 4–1, or Figure 4–2 to protect its components. These implementations take advantage of technologies that WebRTC Session Controller uses to protect itself:

- A design that incorporates network layer access control so you can give individual WebRTC Session Controller components the level of protection they require. This modular design enables you to restrict access to WebRTC Session Controller from the Internet using firewalls, and restrict access within WebRTC Session Controller using WebLogic connection filters.
- The ability to require Secure Sockets Layer (SSL) communication between WebRTC Session Controller components.
- Using operating system hardening to protect specific sensitive files and programs.

Note: Figure 4–1 and Figure 4–2 are only intended to show a high level overview of possible WebRTC Session Controller networking configurations.

For explicit routing details, see the following sections:

- Securing Traffic Between the Internet and WebRTC Session Controller
- Securing Traffic between the WebRTC Session Controller and the SIP Core

---

Figure 4–1 shows the most exposed WebRTC Session Controller components, the WebRTC clients, outside firewall protection in the Internet, with the traffic being filtered by a firewall before being passed through to WebRTC Session Controller. The WebRTC Session Controller Signaling Engine (SE) and Media Engine (ME) instances are deployed in the DMZ behind a firewall as well as a suitable load balancing device for the SE instances.

Note: To provide an additional layer of security, the SE administration server can be a separate server deployed behind an additional firewall on its own separate VLAN (not pictured). Traffic to the SE administration server endpoint (default port 7001/7002) should be disallowed from the Internet.

Suitable load balancing devices include the Apache Software Foundation HTTP web server using **mod_wl**, the F5 Networks 5 load balancer, or the Oracle HTTP Web Server using **mod_wl_ohs**. Oracle recommends that you obtain and install a component with proxy capability to limit traffic between the firewall WebRTC Session Controller as well.

The SIP core itself sits behind a firewall on yet another separate VLAN and traffic from WebRTC Session Controller is routed through a SIP-aware load balancer.

Note: The load balancer serving the SE instances, if SIP-aware, can also be used to balance traffic to the SIP core.

See "Securing WebRTC Session Controller Components in the DMZ" for the list of tasks required to implement this deployment.

Figure 4–1 WebRTC Session Controller DMZ Deployment

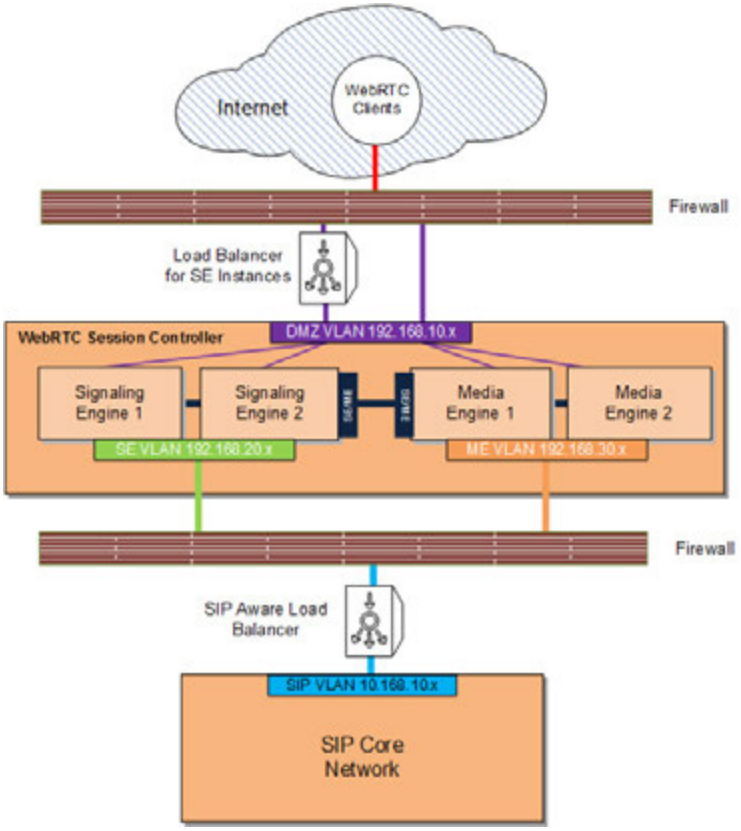
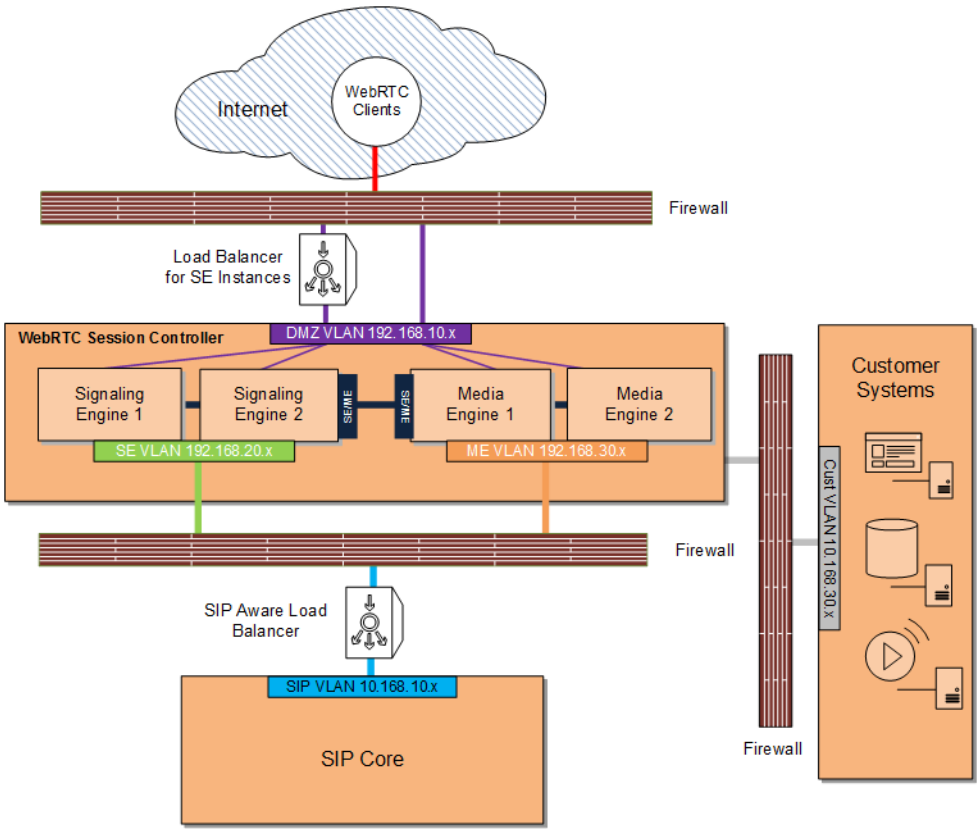


Figure 4–2 shows a WebRTC Session Controller DMZ deployment with an additional firewall separating customer systems such as database servers, media servers, and others. In this deployment, the customer systems are on a separate private VLAN behind a firewall which can only connect to WebRTC Session Controller.

Note: For an additional layer of protection, you can use a VPN tunnel as an access gateway to the customer systems as well (not pictured).

Figure 4–2 WebRTC Session Controller Interfaced with Customer Systems



WebRTC Session Controller Network Sources, Destinations, Protocols, and Ports Reference

Table 4–1 lists the various sources, destinations, protocols and ports which may be used in a WebRTC Session Controller DMZ deployment.

Note: Not all of these source/destination combinations and devices are described in the diagrams and descriptions used in this chapter.

Table 4–1 SE and ME Ports, Source, Destination, and Protocols

Source	Destination	Level 4 Protocol	Level 5-7 Protocol	Default Dest. Ports
Internet Browsers	SE Access IP	TCP	HTTP(S): HTML, JavaScript, WebSockets	7001 for HTTP, and 7002 and/or 443 for HTTPS/WSS
Internet Browsers	ME Access IP	UDP	STUN[1]/TURN[2], DTLS[3]-SRT[C]P[4]	3478 for STUN/TURN over UDP; Configurable range for DTLS-SRT[C]P media, for example, 20000 to 25000[5]
Internet Browsers	ME Access IP	TCP	TURN	3478 for plain TCP, 5349 for TLS
SE Core IP	SBC[6] Signaling IP	UDP and TCP	SIP	5060, 5061
Signaling Core IP	SE Core IP	UDP and TCP	SIP	5060, 5061

Table 4–1 (Cont.) SE and ME Ports, Source, Destination, and Protocols

Source	Destination	Level 4 Protocol	Level 5-7 Protocol	Default Dest. Ports
SE Management Nodes	SE Management IP	TCP	HTTP(S), SSH	7000/7001, 22
ME Management Nodes	ME Management IP	TCP	HTTP(S), SSH	80/443, 22
SE Internal IP	ME Internal IP	TCP	SOAP/HTTP Load Factor	8080
ME Internal IP	SE Internal IP	TCP	SOAP/HTTP Event Callbacks	4057
ME Access IP	Internet Browsers	UDP	STUN, DTLS-SRT[C]P	3478 for STUN over UDP, Configurable range for DTLS-SRT[C]P media, for example, 20000 to 25000
ME Core IP	SBC Media IP	UDP	RTP, RTCP	Configurable range, for example, 20000 to 25000
SBC Media IP	ME Core IP	UDP	RTP, RTCP	Configurable range, for example, 20000 to 25000

[1] Simple Traversal of User Datagram Protocol (UDP) through Network Address Translators (NATs)

[2] Traversal Using Relays around NAT

[3] Datagram Transport Layer Security

[4] The Secure Real-time Transport Protocol

[5] The number of media ports to open, in this case, the default 5000, depends upon your requirements for the number of concurrent media sessions.

[6] Session Border Controller

Securing WebRTC Session Controller Components in the DMZ

Complete these tasks to implement a DMZ deployment as shown in Figure 4–1:

- Securing Traffic Between the Internet and WebRTC Session Controller
- Securing Traffic between the WebRTC Session Controller and the SIP Core

Complete all of the tasks in this section and add a firewall and optional VPN between your customer systems and the WebRTC Session Controller to implement a DMZ deployment as shown in Figure 4–2.

Securing Traffic Between the Internet and WebRTC Session Controller

You secure traffic between the Internet and WebRTC Session Controller by:

- Configuring a firewall between WebRTC Session Controller and the Internet. See "Configure a Firewall to Protect WebRTC Session Controller" for details.
- Configuring WebRTC Session Controller ME specific security measures. See "ME Specific Security Tasks" for details.
- Hardening the underlying operating system components for all SE and ME engines. See "General Hardening Instructions for SE and ME Installations" for details.
- Make sure all of your SE and ME instances are fully patched.
- Optionally, for SE engines, configuring connection filters. See "Configuring Connection Filters for SE Components Instead of a Firewall" for details.

The following sections provides details.

Configure a Firewall to Protect WebRTC Session Controller

Configure a firewall as described in Table 4–2. This example uses the sample components and IP addresses/ports shown in Figure 4–1.

Note: The IP addresses and ports in Table 4–2 are only examples. Yours will be different.

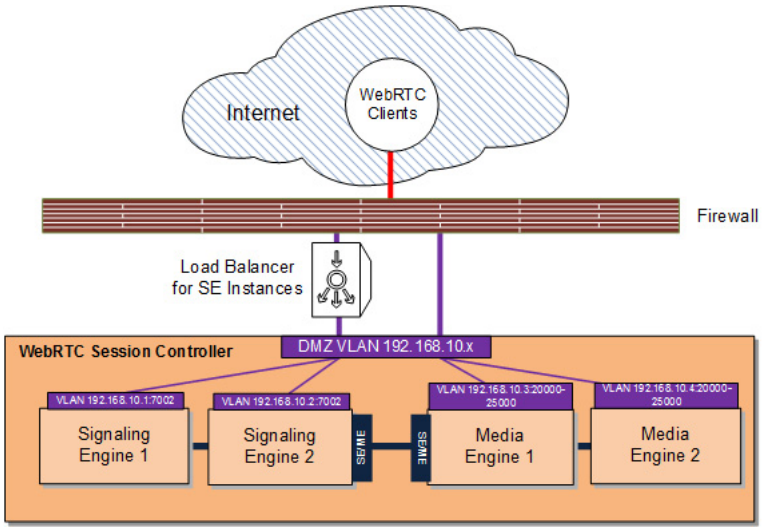
Table 4–2 Configuring a Firewall Between the Internet and WebRTC Session Controller Access Tiers

Specifically Allow Traffic From:	To The Component/IP Address:Port	Notes
WebRTC Clients	Signaling Engine 1 (192.168.10.1:7002)	Allow WebRTC API traffic to SE 1 via a load balancer.
WebRTC Clients	Signaling Engine 2 (192.168.10.2:8002)	Allow WebRTC API traffic to SE 2 via a load balancer.
WebRTC Clients	Media Engine 1 (192.168.10.3:20000-25000)	Allow WebRTC API traffic (DTLS-SRT[C]P) to ME 1 with the appropriate number of media ports open for your requirements, in this case 20000 to 25000.[1]
WebRTC Clients	Media Engine 2 (192.168.10.4:20000-25000)	Allow WebRTC API traffic (DTLS-SRT[C]P) to ME 2 with the appropriate number of media ports open for your requirements, in this case 20000 to 25000.
WebRTC Clients	Media Engine 1 and 2 (192.168.10.3:3478/5349, and 192.168.10.4:3478/5349)	Allow WebRTC API traffic to ME using TURN with ME acting as a TURN server. These ports are not in the diagram.
Signaling Engine 1 and 2	Media Engine 1 and 2 (192.168.10.5:8080, and 192.168.10.6:8080)	SOAP and the HTTP load factor application. These IP addresses are not in the diagram.
Media Engine 1 and 2	Signaling Engine 1 and 2 (192.168.10.7:4057, and 192.168.10.8:4057)	SOAP and HTTP event callbacks. These IP addresses are not in the diagram.

[1] The number of media ports to open, in this case, the default 5000, depends upon your requirements for the number of concurrent media sessions.

Figure 4–3 illustrates the configuration in Table 4–2.

Figure 4–3 Firewall Configuration: Internet to WebRTC Session Controller



Securing Traffic between the WebRTC Session Controller and the SIP Core

To secure traffic between WebRTC Session Controller and the SIP Core:

- Obtain and configure a firewall between the WebRTC Session Controller and the SIP Core so that it allows only RTCP traffic between the SIP Core and the ME engines.

  See "Configuring a Firewall Between WebRTC Session Controller and the SIP Core" for more information.

- Follow the instructions for hardening SE administration servers in "Securing the Signaling Engine Administration Server".
- Secure Node Manager access to SE engines as described in "Securing Node Manager Access to SE".
- Make sure all of your ME and SE servers are fully patched.

Configuring a Firewall Between WebRTC Session Controller and the SIP Core

This example uses the sample components and IP addresses/ports shown in Figure 4–1.

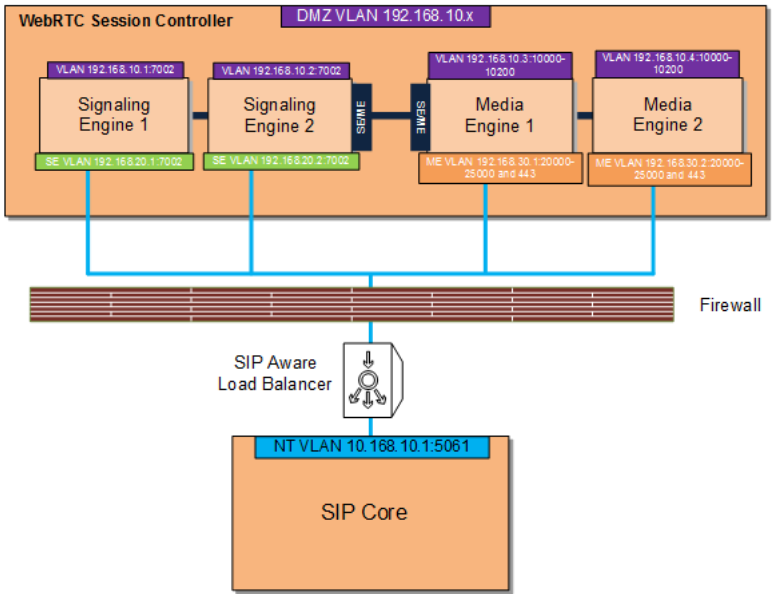
Note: The IP addresses and ports in Table 4–3 are only examples. Yours will be different.

Table 4–3 Configuring a Firewall Between WebRTC Session Controller and the SIP Core

Specifically Allow Traffic From:	To The Component/IP Address:Port	Notes
Signaling Engine 1	SIP Core (10.168.10.1:5061)	SIP traffic flows from SE 1to the SIP core and back through a SIP-aware load balancer.
Signaling Engine 2	SIP Core (10.168.10.1:5061), SE Admin Server (10.168.20.1:7002)	SIP traffic flows from SE 1to the SIP core and back through a SIP-aware load balancer. Access to the SE Admin server hosted on SE 1.
Media Engine 1	SIP Core (10.168.10.1:5061)	RTP and RTCP traffic flows from ME 1 to the SIP core and back.
Media Engine 2	SIP Core (10.168.10.1:5061)	RTP and RTCP traffic flows from ME 2 to the SIP core and back.
SIP Core	SE 1 (192.168.20.1:5060/5061), SE 2 (192.168.20.2:5060/5061), ME 1 (192.168.30.1:20000-25000), ME 2 (192.168.30.2:20000-25000)	N/A

Figure 4–4 illustrates the configuration in Table 4–3.

Figure 4–4 Firewall Configuration: WebRTC Session Controller to the SIP Core



General Hardening Instructions for SE and ME Installations

Keep these operating system level security considerations in mind:

- Confirm that the SE and ME binaries are owned by the WebRTC Session Controller installation user.

Note: File permissions and ownership are set correctly by the WebRTC Session Controller installer, but you should verify that they have not been modified before deployment.

- For SE, lock down access to everything except:
  - Read/write access to the file system below the WebLogic domain directory
  - Access to the Java Virtual Machine (JVM)
  - Access to the RMI, and HTTP/HTTPS ports.
- Periodically audit the operating system file system file to notify administrators of unauthorized system binary changes.

Oracle Linux Security Hardening Information

For detailed hardening instructions pertinent to Oracle Linux, see the following sections in the *Oracle Linux Security Guide*:

- *Pre-Installation Tasks* which includes information on physical security, BIOS passwords and other system level considerations.
- *Installing Oracle Linux* which includes information on configuring shadow passwords and hashing, disk partition encryption, software selection and network time services.
- *Implementing Oracle Linux Security* which includes information on topics including:
  - *Configuring and Using Data Encryption*
  - *Configuring and Using Access Control Lists*
  - *Configuring and Using SELinux*
  - *Configuring and Using Auditing*
  - *Configuring and Using System Logging*
  - *Configuring and Using Process Accounting*
  - *Configuring Access to Network Services*
  - *Configuring and Using Chroot Jails*

SE Specific Security Tasks

This section provides details on security tasks that are specific to WebRTC Session Controller Signaling Engine.

Securing the Signaling Engine Administration Server

Securing the administration server involves these tasks:

- Encrypting the RMI traffic between the SE engines and the SE administration server. See "Encrypt SE RMI Traffic Between SE and the SE Administration Server" for details.
- Configuring the WebRTC Session Controller SE and ME administration servers to use a nonstandard port so that you can use customized firewall rules as well as encryption (HTTPS, IIOPS, or T3S). See "Securing the SE Administration Network Channel" for details.

- Changing the SE administration server context path from the default of **/console** to something else, for example, **/adminconsole**. See "Configuring the SE Admin Server to Use a Non-standard Context Path" for details.
- Configuring WebRTC Session Controller to only allow SSL traffic to the administration server. See "Restricting the SE Administration Server to SSL" for details.

Encrypt SE RMI Traffic Between SE and the SE Administration Server

To encrypt RMI traffic between SE and the SE administration server, for each SE server:

1. Open the Administration Console for your domain.
2. Click the **Lock & Edit** button.
3. Expand the **Environments** node in the Domain Structure pane and click the **Servers** node.
4. Click the **Configuration** tab in the Summary of Servers pane and click the name of the server in the Servers table that you want to configure.
5. Check **SSL Listen Port Enabled**.

> **Note:** Weblogic server uses the default JKS file store (Demo Identity and Demo Trust) for SSL configuration. However, you could specify a custom trust Keystore. See the *Oracle WebLogic Server 12c: Configuring Managed Servers* document for details.

6. Enter a numeric port number in the **SSL Listen Port** edit box.
7. Click **Save** to save your configuration changes.
8. Expand the **Environments** node in the Domain Structure pane if it is not already expanded and click **Clusters**.
9. Click the Signaling Engine cluster and then click the **General** tab.
10. Replace the port in the **Cluster Address** edit box with the SSL port you configured in step 6.
11. Click the **Configuration** tab, then the **Replication** tab.
12. Check **Secure Replication Enabled**.
13. Repeat steps 9 through 12 for the remaining SE servers.
14. Click **Save** to save your configuration changes.
15. Click **Activate Changes** to apply your changes to the SE servers.
16. To enable a secure channel for Java Message Service (JMS) add the **-Dweblogic.DefaultProtocol=t3s** flag to **JAVA_OPTIONS** in the *middleware_home*/**bin/setDomainEnv.sh** script:

```
JAVA_OPTIONS="${JAVA_OPTIONS} -Dweblogic.DefaultProtocol=t3s"
export JAVA_OPTIONS
```

17. Change the **ADMIN_URL** item in the *domain_home*/**bin/startManagedWeblogic.sh** script to **https://***IP_address*:*port_number*
18. Restart each Signaling Engine server with this command:

```
startManagedManaged.sh server_name https://IP_address:port_number
```

Note: Make sure you enable SSL on your administration server as well to ensure that SSL is used throughout the cluster.

Network traffic between SEs and the SE administration server is now encrypted.

Securing the SE Administration Network Channel

In a DMZ deployment you should configure Signaling Engine servers to use custom ports as well as HTTPS, and certificates if required.

To secure the network channels:

1. Set your Signaling Engine environment:

   ```
   cd ~/domain_home/bin
   . ./setDomainEnv.sh
   ```

   where *domain_home* is the path to the domain's home directory.

2. Start WLST:

   ```
   java weblogic.WLST
   ```

3. Connect to the server using the *username* and *password* you configured during installation:

   ```
   connect('username','password','t3://myserver:port_number')
   ```

4. Switch to the server:

   ```
   cd('/Servers/myserverendpoint')
   ```

5. Create a new network channel:

   ```
   cmo.createNetworkAccessPoint('MyChannelName')
   ```

6. Switch to the new network channel:

   ```
   cd('/Servers/MyDomain/NetworkAccessPoints/MyChannelName')
   ```

7. Configure the network channel port:

   ```
   cmo.setProtocol('https')
   cmo.setListenPort(nnnn)
   cmo.setEnabled(true)
   cmo.setHttpEnabledForThisProtocol(true)
   cmo.setTunnelingEnabled(false)
   cmo.setOutboundEnabled(false)
   ```

8. Configure certificate usage and HTTPs for the network channel:

   ```
   cmo.setClientCertificateEnforced(true)
   cmo.setTwoWaySSLEnabled(true)
   ```

IMPORTANT: The default channel settings remain stored in ServerMBean and SSLMBean, and are used if necessary to provide connections to a server instance. Make sure you *explicitly* specify the secure channel as the communications path between access and network tiers.

Configuring the SE Admin Server to Use a Non-standard Context Path

Oracle recommends changing the administration server context path from the default of **/console** to something like **/adminportal**.

To change the SE admin context path:

1. Set your SE environment:

   ```
   cd ~/domain_home/bin
   . ./setDomainEnv.sh
   ```

   where *domain_home* is the path to the domain's home directory.

2. Start the WebLogic Scripting Tool (WLST):

   ```
   java weblogic.WLST
   ```

3. Connect to the administration server for your WebRTC Session Controller domain using the *username* and *password* you configured during installation:

   ```
   connect('username','password','t3://myadminserver:port_number')
   ```

4. Switch to the administration server:

   ```
   cd('/Servers/AdminServer')
   ```

5. Create a new network channel:

   ```
   cmo.createNetworkAccessPoint('MyAdminServer')
   ```

6. Switch to the new network channel:

   ```
   cd('/Servers/AdminServer/NetworkAccessPoints/MyAdminServer')
   ```

7. Set the network channel's protocol to **admin**:

   ```
   cmo.setProtocol('admin')
   ```

8. Configure the administration server port:

   ```
   cmo.setListenPort(nnnn)
   cmo.setEnabled(true)
   cmo.setHttpEnabledForThisProtocol(true)
   cmo.setTunnelingEnabled(false)
   cmo.setOutboundEnabled(false)
   ```

9. Enable bidirectional SSL:

   ```
   cmo.setTwoWaySSLEnabled(true)
   ```

10. Disable client side certificates if they are not in use:

    ```
    cmo.setClientCertificateEnforced(false)
    ```

11. Set the listen addresses for the new administration channel:

    ```
    cmo.setPublicAddress('nnn.nnn.nnn.nnn')
    cmo.setListenAddress('nnn.nnn.nnn.nnn')
    ```

Restricting the SE Administration Server to SSL

To configure the SE administration server to allow only SSL:

1. Open the Administration Console for your domain.
2. Click **Lock & Edit**.

3. Expand the **Environments** node in the **Domain Structure** pane and click the **Servers** node.
4. Click **AdminServer(Admin)**.
5. Click **Configuration** in the **Summary of Servers** pane and click the name of the server in the Servers table to configure.
6. Click **General**.
7. Check **SSL Listen Port Enabled**.

> **Note:** Weblogic server uses the default JKS file store (Demo Identity and Demo Trust) for SSL configuration. However, you should specify a custom trust Keystore. See the *Oracle WebLogic Server 12c: Configuring Managed Servers* document for details.

8. Enter a numeric port number in the **SSL Listen Port** edit box.
9. Uncheck the **Listen Port Enabled** box.
10. Click **Save**.
11. Click **Activate Changes** to apply your changes to the engine servers.

Securing Node Manager Access to SE

You can control WebRTC Session Controller by using Oracle WebLogic Node Manager (Node Manager) features. Node Manager relies on a one-way SSL connection for security. See "Configuring Java-based Node Manager Security" and "Using SSL with Java-Based Node Manager" in *Fusion Middleware Node Manager Administrator's Guide for Oracle WebLogic Server 12c* for details.

Configuring Connection Filters for SE Components Instead of a Firewall

In cases where WebRTC Session Controller SE components are not separated by firewalls, for instance between an SE server and an SE administration server, you can use WebLogic connection filters to provide network layer access control and block unwanted intrusions.

To configure a WebLogic connection filter:

1. Set your SE environment:

```
cd ~/domain_home/bin
. ./setDomainEnv.sh
```

   where *domain_home* is the path to the domain's home directory.

2. Start WLST:

```
java weblogic.WLST
```

3. Connect to the server using the *username* and *password* you configured during installation:

```
connect('username','password','t3://myserver:port_number')
```

4. Switch to the domain security MBean:

```
cd('/SecurityConfiguration/'+domainName)
```

5. Enable a connection filter:

```
cmo.setConnectionLoggerEnabled(true)
```

6. Define the connection filter implementation:

```
cmo.setConnectionFilter('weblogic.security.net.ConnectionFilterImpl')
```

> **Note:** The example above uses the default connection filter implementation. For information on creating custom connection filters see "Developing Custom Filters" in *Fusion Middleware Programming Security for Oracle WebLogic Server*.

7. Configure the rules as a string array:

```
set('ConnectionFilterRules',jarray.array
([String('myserver ip_address port allow t3s https'),
String('ip_address/subnet_mask ip_address port allow'),
String('ip_address ip_address port deny t3 http')],
String))
```

ME Specific Security Tasks

After installing ME instances:

- Configure user accounts and restrict access to the ME instances as described in "Configuring Permissions, Users, and Authorization" in *Oracle Communications WebRTC Session Controller System Administrator's Guide*.
- Configure Secure Real-time Transport Protocol (SRTP) sessions as described in "Configuring Secure Media (SRTP) Sessions" in *Oracle Communications WebRTC Session Controller Installation Guide*.