Authentication and Authorization

Authentication

Sonus WebRTC solution supports various user authentication methods. Web application developer can choose an authentication method based on or their application and use cases. Following are the authentication methods which a web developer can choose from:

- **LDAP authentication** - authenticates users with an enterprise's Active Directory (AD).
- **OAuth2.0** - authenticates users with social networks like Google, Facebook, LinkedIn or authenticate users with custom OAuth2.0 server.
- **Web Application Auth Tokens** - authenticates users with web applications AuthTokens that provides access to WebRTC service.
- **Generate wrtcAuthToken** - authenticate users using WebRTC gateway (GW) REST API.
- **No Authentication** - allows users to access WRTC service without any authentication.

Following are the steps involved in authenticating users:

1. User authentication is performed by any one of the above authentication servers.
2. On successful authentication, the WRTC GW generates wrtcAuth token and includes the same in HTTP response.
3. After receiving the wrtcAuth token, the web application through SDK creates a websocket with WRTC.
4. After a websocket is established, user logs in to register and receive incoming calls.

LDAP Authentication

In this authentication the WRTC application is integrated with an enterprise authentication management system such as Active Directory (AD) using LDAP. A user is authenticated based on the username and password entered. This approach can be easily integrated with an enterprise authentication management system. The following figure depicts the LDAP authentication call flow:

**Figure 1: LDAP Authentication Call Flow**

For more information on LDAP provisioning refer Step 6 of Configuring WRTC Without Using EMS. For Authentication API refer authenticate().

OAuth Authentication

In this authentication the WRTC application lists the trusted authentication brokers. A WRTC user must have an account with the listed authentication brokers and login using the account credentials. The authentication broker validates the user credentials entered and sends a security access token to the WRTC. On receiving the access token, WRTC access the user information from authentication brokers and allow the user to login to the Web application.
WRTC supports the following authentication brokers:

- Google
- Facebook
- LinkedIn
- REST based

OAuth 2.0 based authentication procedure for WRTC devices is implemented to support WebRTC endpoints such as browser endpoints; mobile devices such as Android and iOS.

The following figure details a sample call flow for OAuth 2.0 based authentication:

**Figure 2: Google Authentication Call Flow**

On successful authentication, the WRTC assigns a `wrtcAuthToken` which expires after a configured time interval, for example, 1 through 30 minutes. The WebRTC framework uses HTML5 WebSocket to establish the signaling channel between the browser and WRTC GW. WRTC GW functions as a WebSocket server and the web browser behaves as a WebSocket client. After a WebSocket connection is established, the JavaScript client sends SJCP `LOGIN` request for establishing a reliable connection over WebSocket. WRTC responds with `LOGIN` response to the WebRTC user with `connection_id` and the authenticated user profile.

### Google Authentication

User authentication is required in WRTC as the web application server stores information and maintains the user state. When a user opts to login to WRTC application using Google authentication, WRTC initiates the OAuth authorization process and redirects the user to the Google's authorization server along with the required scope of access for the user's account. Generally, the scope specifies actions that the WRTC can perform on behalf of the user, such as retrieving, updating, inserting, or deleting data. On user's consent, Google returns a token to WRTC. WRTC validates this token that lets it authorize all the requests.

A refresh token retrieves new access token when the previously generated token expires.

For more information on Google OAuth account creation, refer to [Google Developer Account Creation](#). For OAuth authentication API, refer to [authenticate()](#).

For more information on provisioning OAuth server without EMS, refer to [Step 6 of Configuring WRTC Without Using EMS](#) or with EMS, refer to [WRTC Cluster Configuration](#).

### Facebook Authentication
When a user opts to login to WRTC application using Facebook authentication, WRTC initiates a OAuth authorization process and redirects the user to the Facebook's authorization server along with the required scope of access for the user's account. Generally, the scope specifies actions that the WRTC can perform on behalf of the user, such as retrieving, updating, inserting, or deleting data. On user's consent, Facebook returns a token to WRTC. WRTC validates this token that lets it authorize all the requests.

A refresh token retrieves new access token when the previously generated token expires.

For more information on Facebook OAuth service provisioning refer Facebook Developer Account Creation. For OAuth authentication APIs refer authenticate().

For more information on provisioning OAuth server without EMS refer Step 6 of Configuring WRTC Without Using EMS or with EMS refer WRTC Cluster Configuration.

LinkedIn Authentication

When user selects Linked-In for authentication, WRTC initiates the OAuth 2.0 authorization process and redirects the user to the Linked-In's authorization server, along with the required scope of access for the user's account. Generally, the scope specifies the actions that WRTC must perform on behalf of the user, such as retrieving, updating, inserting, or deleting data. On user's consent, Linked-In returns a token to WRTC. WRTC then validates this token that lets it authorize all the requests. A refresh token retrieves new access token when the previously generated token expires.

The following figure lists the permissions required for WRTC to complete the OAuth2 authentication flow. When creating an application on LinkedIn which the Enterprise or Service Provider will use to authenticate WRTC users, the OAuth User Agreement section must have the options checked, as shown in the following figure. Generally, WRTC configures these values.

Figure 3: OAuth User Agreement

<table>
<thead>
<tr>
<th>Default Scope</th>
<th>r_emailaddress</th>
<th>r_network</th>
<th>rw_groups</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>r_contactinfo</td>
<td>rw_company_admin</td>
<td>aw_messages</td>
</tr>
<tr>
<td></td>
<td>rw_mis</td>
<td>r_basicprofile</td>
<td>r_fullprofile</td>
</tr>
</tbody>
</table>

Selecting both r_basicprofile and r_fullprofile is redundant, r_basicprofile will be selected if neither r_basicprofile nor r_fullprofile is checked.

For more information on LinkedIn OAuth service provisioning refer LinkedIn Developer Account Creation. For OAuth authentication APIs refer authenticate().

For more information on provisioning OAuth server without EMS, refer Step 6 of Configuring WRTC Without Using EMS or with EMS refer WRTC Cluster Configuration.

Web Application AuthTokens

In this authentication method, web application assigns a auth_token to a user, which is sent to the WRTC server for validation. WRTC GW validates this auth_token with developer's authentication server over REST interface and on successful authentication, WRTC assigns wrtcAuthToken to the user to establish WebSocket connection with WRTC GW.

⚠️ This authentication is configurable only through editing startup.json file, and it is not configurable through EMS.

The following figure depicts the application:

Figure 4: Application Based Authentication
For more information on webAppAuthToken provisioning refer Step 8 of Configuring WRTC Without Using EMS. For Authentication API refer webAppTokenAuthenticate().

Generate wrtcAuthToken (Third party Authentication)

For the enterprise who wish to use their own authentication can use Generate wrtcAuthToken or Third paty authentication type. In this authentication type, enterprise should have their own ADMIN webservice and validate their users. On successful authentication, the web server uses the REST API of WRTC GW to request wrtcAuthToken on behalf of the user. For security, developer's web server authenticates itself using admin_ID and password of an enterprise admin.

The following figure depicts the call flow in generating wrtcAuthToken:

Figure 5: Third-party Authentication Call Flow
For more information on wrtcAuthToken provisioning refer Configuring WRTC Without Using EMS. For Authentication API refer authenticate ()

No Authentication

If required, web developer can choose not to authenticate users and generate a wrtcAuthToken without any authentication. Which is then used to access the WebRTC service through WRTC GW. For unauthenticated users, web developers can allocate temporary IDs for example, phone numbers from the pool of numbers configured on their enterprise or assigning the users as anonymous. This authentication method is implemented for Click-to-connect use cases, such as Mayday button, contact center help, and so on, where the user is always expected to initiate a call.

⚠️ When users are assigned with anonymous (no identity is assigned), they can only make audio, video, or IM with other WebRTC users.

For authentication API, refer authenticate ().