E-mail Server Administration Guide

Setting up Gmail mailboxes for OAuth 2.0 authorization

4/13/2026

Setting up Gmail mailboxes for OAuth 2.0 authorization

For basic authentication, Genesys E-mail Server can access Gmail mailboxes using the IMAP, POP3, and SMTP protocols.

Starting with version 8.5.201.05, E-mail Server supports the OAuth 2.0 authorization access to Gmail mailboxes with the IMAP and SMTP protocols. Starting with version 8.5.202.02, OAuth 2.0 support is extended to POP3 and SMTP protocols.

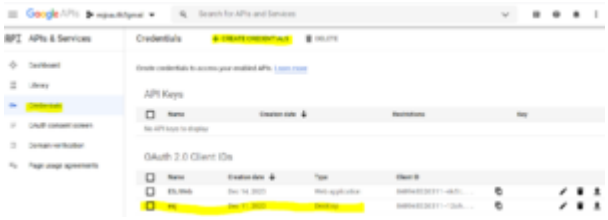
To set up Gmail mailboxes using the OAuth 2.0 authorization access:

1. Create a Google application.
2. Configure Genesys E-mail Server.

Creating a Google application

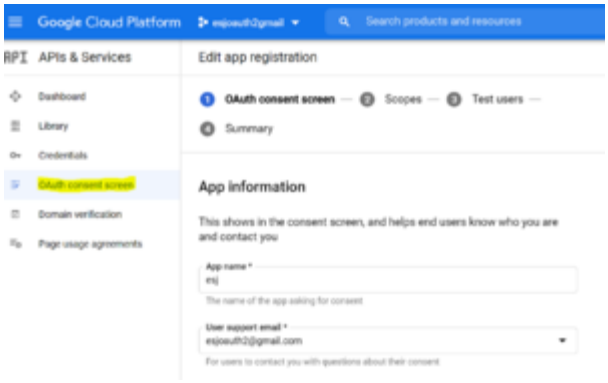
For OAuth 2.0 authorization access to Gmail mailboxes with IMAP, POP3, and SMTP protocols, create a Google application in the Google platform. (In our example, the **esj** account is created.)

1. Follow this Google documentation to configure the application. The main configuration points are included in this procedure.
2. Select **Desktop** as an application type. E-mail Server uses "Manual copy/paste" as the redirect method.



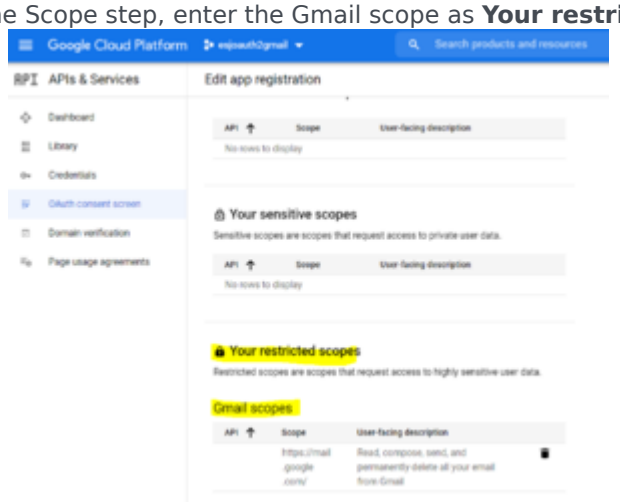
(Click to expand)

3. Download the **client_id** and **client_secret** by clicking the Download arrow of your Google application. These values are required to get the access token and to configure Genesys E-mail Server.
4. On the OAuth consent screen, add your **App information** (App name, User support email). For example:



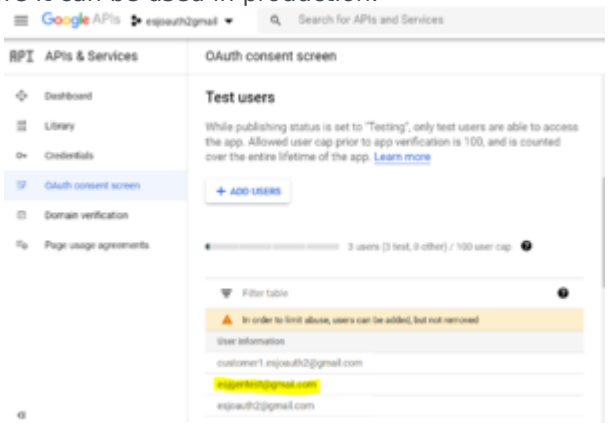
(Click to expand)

5. In the Scope step, enter the Gmail scope as **Your restricted scopes**.



(Click to expand)

6. (Optional) Add test users. This step is for the testing phase. You can add an existing or new mailbox that the E-mail Server can access as a user. The application must be published after the testing phase before it can be used in production.



(Click to expand)

7. Get a refresh token manually. Follow the steps as described in this Google documentation to get the OAuth 2.0 refresh token:
   - **Step 1**: Generate a code verifier and challenge. (Note: A refresh token can be acquired without this

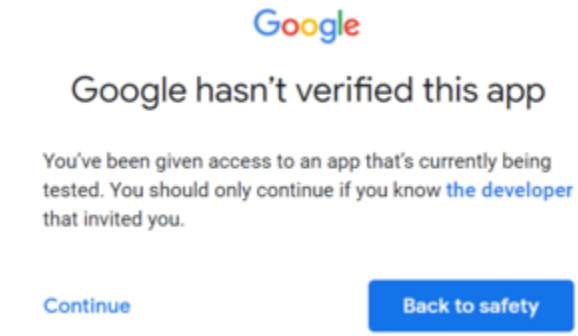
step.)

- **Step 2**: Send a request to Google's OAuth 2.0 server. In a web browser, enter the following as a URL, replacing *<your client id>* with your application client ID:

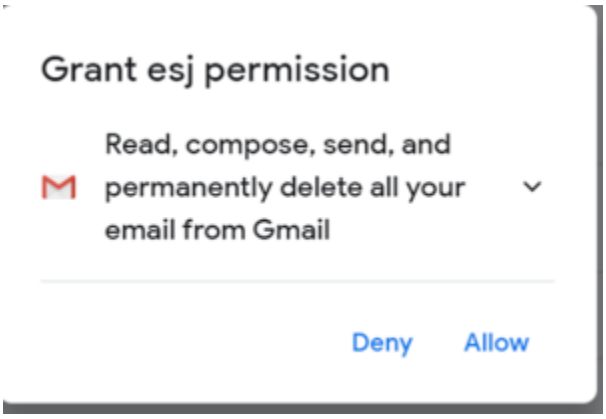
  ```
  https://accounts.google.com/o/oauth2/
  auth?scope=https://mail.google.com/&redirect_uri=urn:ietf:wg:oauth:2.0:oob&response_type=code&clien
  client id>
  ```

  Note that **redirect_uri** and **response_type** values cannot be changed.

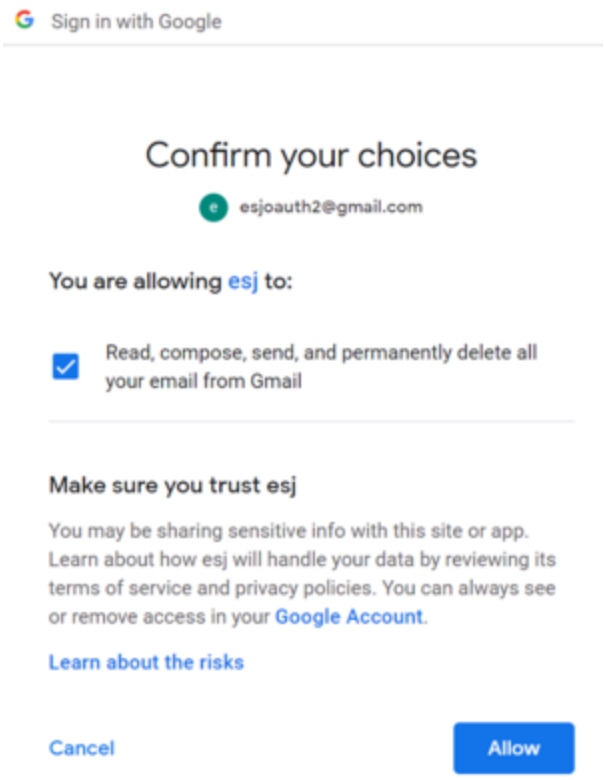
- **Step 3**: Google prompts the user for consent. You will be prompted to sign in with a mailbox if it was not signed in. In the test phase, if you have multiple mailboxes, only the mailboxes that have been added as Test Users can access the application. This may change after the application is published. After signing in with mailbox credentials, there will be an alert in the test phase. Click **Continue**:



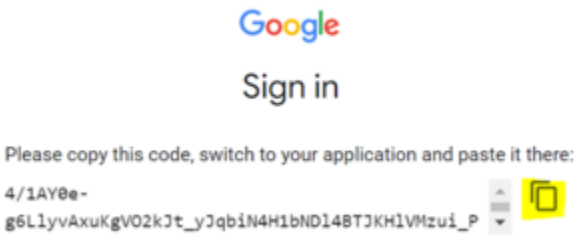
Click **Allow**:



Click **Allow**:



The Authorization Code is displayed. Copy the Code by clicking **Copy Icon**:



- **Step 4**: Exchange authorization code for refresh and access tokens.
  The Authorization Code acquired in Step 3 can be used to exchange for OAuth 2.0 access and refresh tokens within 10 minutes (after you received the authorization code) by means of the following command:

  ```
  curl -d "code=<your authorization
  code>&grant_type=authorization_code&redirect_uri=urn:ietf:wg:oauth:2.0:oob&client_id=<your
  client_id>&client_secret=<your client secret>" -X POST
  https://oauth2.googleapis.com/token
  ```

  Replace *<your authorization code>*, *<your client_id>*, and *<your client_secret>* with the actual values of your application. Keep the rest as is.

  Here is an example of the response:

  ```
  {
  ```

"access_token": "ya29.a0AfH6S7ztJBShUiXC-z7P_-**********",

"expires_in": 3599,

"refresh_token": "1//06LxPX1ZCgYIARAAGAYSNwF**********",

"scope": "https://mail.google.com/",

"token_type": "Bearer"

}

The Bearer **access_token**, which can be used to access the mailbox in IMAP/POP3/SMTP, expires every 3600 seconds. The **refresh_token** can be used to get a new Bearer token. After the application is published, the refresh code will only expire under the conditions listed in this Google document.

Configuring Genesys E-mail Server

To configure E-mail Server:

- Set the JavaMail property **mail.<type>.auth.mechanisms** (where *<type>* can be **imap**, **pop3**, and **smtp**) to X0AUTH2. (To disable OAuth 2.0, remove the JavaMail property.)
- Configure options in the **smtp-client** section.
- Configure options in the **pop-client** section.
- Configure options in the **secret-<*secretName*>** section.

Configuring the **smtp-client** section

Configure the following configuration options:

- client-id—Specify the Client ID of the Google application.
- password—Specify the refresh_token of the SMTP account.
- secret—Specify the secretName of the secret-*<secretName>* section to associate with the Google application secret.
- tenant-authority—Specify the valid Google authority server, which is a case-insensitive string that contains "google".

Configuring the **pop-client** section

Configure the following configuration options:

- client-id—Specify the Client ID of the Google application.
- password—Specify the refresh_token of the Gmail mailbox.

- secret—Specify the secretName of the secret-<*secretName*> section to associate with the Google application secret.
- tenant-authority—Specify the valid Google authority server, which is a case-insensitive string that contains "google".

Configuring the **secret-<*secretName*>** section

- password—Specify the client_secret value of the registered Google account.

Limitations

- During the test phase, a refresh token expires in 7 days.
- During the test phase, the refresh token may stop working if the Gmail mailbox is not used for a few days.