MATLAB® Web App Server™

User's Guide

MATLAB®

MathWorks®

How to Contact MathWorks

Latest news: www.mathworks.com

Sales and services: www.mathworks.com/sales_and_services

User community: www.mathworks.com/matlabcentral

Technical support: www.mathworks.com/support/contact_us

Phone: 508-647-7000

The MathWorks, Inc.
1 Apple Hill Drive
Natick, MA 01760-2098

Revision History

Contents

4

Cloud Deployment

5

Web App Packaging and Deployment

6

7

Introduction

MATLAB Web App Server Product Description

Share MATLAB apps and Simulink simulations as browser-based web apps

MATLAB Web App Server lets you host MATLAB apps and Simulink® simulations as interactive web apps. You can create apps using App Designer, package them using MATLAB Compiler™, and host them using MATLAB Web App Server. Your end-users can access and run the web apps using a browser without installing additional software.

MATLAB Web App Server supports integration with authentication standards such as OpenID Connect and LDAP so that you can control access to your web apps. You can host and share multiple apps developed using different releases of MATLAB and Simulink.

MATLAB Web App Server Differences

MathWorks® has two offerings of MATLAB Web App Server:

- A standalone MATLAB Web App Server product.
- A development version of MATLAB Web App Server included in MATLAB Compiler.

The differences between the standalone MATLAB Web App Server product and the development version of MATLAB Web App Server in MATLAB Compiler are listed in the following table.

Supported Functionality	MATLAB Web App Server Product	Development Version of MATLAB Web App Server in MATLAB Compiler
Authentication	✓	✗
Role-based access	✓	✗
Policy-based access	✓	✗
Support for multiple releases of MATLAB	✓	✗
Number of concurrent sessions	Unlimited [a]	32
Server setup and configuration	Command-line only	Graphical user-interface

a   Subject to server hardware limitations.

See Also

More About

- Product Requirements and Platform Availability
- "Install or Uninstall MATLAB Web App Server Product" on page 3-2

System Requirements and Supported Browsers

Supported Browsers and Platform Incompatibilities

Supported Browsers

Web apps are compatible with commonly used browsers such as Google Chrome™, Safari, Firefox®, Microsoft Edge®.

For browsers that update automatically, the current stable version is supported.

Web apps are also supported on the Safari browser on iPads, and the Chrome™ browser on Chromebooks.

Platform Incompatibilities

MATLAB Web App Server cannot be installed on Red Hat® Enterprise Linux® 6.

Linux systems require the `systemd` software suite.

See Also

More About
- "Install or Uninstall MATLAB Web App Server Product" on page 3-2
- "Set Up MATLAB Web App Server" on page 3-4

Installation and Setup

Install or Uninstall MATLAB Web App Server Product

Warning MATLAB Web App Server must be installed in a trusted intranet environment on dedicated hardware. The only purpose of the physical or virtual machine where the server is installed must be to host web apps that connect to the server. The server must never be exposed to the open Internet. For more information, see "MATLAB Web App Server Security" on page 4-24.

Install

Installing the MATLAB Web App Server product requires a valid software license, which you can obtain by purchasing the product or downloading a product trial. For more information, visit the MathWorks Store.

To start the installation, run the MathWorks installer and select the MATLAB Web App Server product for installation. To download the installer, visit MathWorks Downloads. For detailed instructions, see "Install Products Using Internet Connection".

The installation process creates a default installation folder based on your operating system and the release version. For instance, the default installation folder for R2022a is as follows:

Operating System	Default Installation Folder
Windows®	`C:\Program Files\MATLAB\MATLAB Web App Server\R2022a`
Linux	`/usr/local/MATLAB/MATLAB_Web_App_Server/R2022a`
macOS	`/Applications/MATLAB/MATLAB_Web_App_Server/R2022a`

Uninstall

Caution Before uninstalling the MATLAB Web App Server product from your system, you need to execute `webapps-uninstall` from the system command-line. Executing `webapps-uninstall` unregisters services and removes default user accounts used by the services. Failing to execute `webapps-uninstall` results in orphan services and user accounts and may make subsequent installations of the product unsuccessful.

To uninstall the MATLAB Web App Server product, follow the usual instructions for uninstalling MathWorks products. For more information, see "Uninstall MathWorks Products".

You can manually delete the following folders after uninstalling the product:

Operating System	Folders
Windows	**Apps Folder:** `%ProgramData%\MathWorks\webapps\R2022a\apps` **Logs Folder:** `%ProgramData%\MathWorks\webapps\R2022a\logs` **Config Folder:** `%ProgramData%\MathWorks\webapps\R2022a\config`
Linux	**Apps Folder:** `/local/MathWorks/webapps/R2022a/apps` **Logs Folder:** `/local/MathWorks/webapps/R2022a/logs` **Config Folder:** `/local/MathWorks/webapps/R2022a/config`
macOS	**Apps Folder:** `/Library/Application Support/MathWorks/webapps/R2022a/apps` **Logs Folder:** `/Library/Application Support/MathWorks/webapps/R2022a/logs` **Config Folder:** `/Library/Application Support/MathWorks/webapps/R2022a/config`

See Also

More About

- "Set Up MATLAB Web App Server" on page 3-4

Set Up MATLAB Web App Server

You can set up the server once you complete installing the MATLAB Web App Server product. For information on installing the product, see "Install or Uninstall MATLAB Web App Server Product" on page 3-2.

License Manager

You can set up the MATLAB Web App Server without a license manager. However, you will need a license manager running on your network prior to using the MATLAB Web App Server. For more information, see "Install License Manager on License Server".

Install MATLAB Runtime

1   Download the MATLAB Runtime installer from the MathWorks website or the MATLAB desktop.

Option	Steps

MathWorks Website	Select the appropriate platform and release-specific installer from:
	
	`https://www.mathworks.com/`
	`products/compiler/matlab-`
	`runtime.html`

MATLAB Desktop	At the MATLAB command prompt, type: `compiler.runtime.download`

2   Install the MATLAB Runtime using the installer. For installation instructions, see "Install and Configure MATLAB Runtime" (MATLAB Compiler).

Set Up the Server

1   After installing the MATLAB Web App Server product, navigate to the folder containing the MATLAB Web App Server command-line scripts.

Operating System	Default Location of Command-Line Scripts
Windows (*Administrator*)	`C:\Program Files\MATLAB\MATLAB Web App Server\R2022a\script`

Linux (*sudo*)	`/usr/local/MATLAB/`
	`MATLAB_Web_App_Server/R2022a/`
	`script`

macOS (*sudo*)	`/Applications/MATLAB/`
	`MATLAB_Web_App_Server/R2022a/`
	`script`

2   At the operating system command line, start the interactive setup interface by typing:

Operating System	Command
Windows *(Administrator)*	`webapps-setup`
Linux *(sudo)*	`webapps-setup`
macOS *(sudo)*	`webapps-setup`

MATLAB Web App Server registers two services:

- A service to run the server.
- A service to run the apps.

For platform-specific service names, see "Service Information, Groups, and Folder Locations" on page 3-5.

3 Enter information about the license server at the prompt. You can specify this information in one of two ways:

- Port Number@License Server Name. For example: `27000@myLicenseServer`.
- Path to the license file. For example: `C:\myLicenses\license.lic`.

If you do not have a license manager set up, you can just press **Enter** to continue with the rest of the setup process. After setting up your license manager, use `webapps-config` to connect with the license manager.

4 Specify a user account to run the server service and the apps service. To use the default accounts press **Enter**. For account names, see "Service Information, Groups, and Folder Locations" on page 3-5.

`MwWebAppServerR2022a`, the default account for running the server service, has permissions to process HTTP and HTTPS traffic and manage authentication. The account has permissions to write to the apps folder that contains web apps. It can read an SSL private key and authentication configuration associated with the server.

`MwWebAppWorkerR2022a`, the default account for running the apps service, has the permissions of a standard user on the machine. The account can read and execute web apps as well as generate logs.

Warning Do not use the same account for the server service and the app service. Using the same account makes your server vulnerable to malicious app exploits.

5 Specify the MATLAB Runtime versions you want to use by typing `y` or `n` at the prompt. The setup utility automatically finds all MATLAB Runtime versions installed on your machine as long as they are installed in the default installation location.

If you do not have MATLAB Runtime installed, follow the instructions in "Install MATLAB Runtime" on page 3-4, and configure it using `webapps-runtime`.

With the setup complete, you can start the server using `webapps-start`.

Service Information, Groups, and Folder Locations

Setting up the MATLAB Web App Server creates two services and folders for uploading apps, capturing logs, and managing the server configuration.

Service Information

Operating System	Server Service Information	Apps Service Information
Windows	**Account Name:** `MwWebAppServerR2022a` **Service Name:** `mw-webapps-R2022a`	**Account Name:** `MwWebAppWorkerR2022a` **Service Name:** `mw-webapps-launcher-R2022a`
Linux	**Account Name:** `MwWebAppsServerR2022a` **Service Name:** `mw-webapps-R2022a` **Service File:** `/etc/systemd/system/mw-webapps-R2022a.service`	**Account Name:** `MwWebAppsWorkerR2022a` **Service Name:** `mw-webapps-launcher-R2022a` **Service File:** `/etc/systemd/system/mw-webapps-launcher-R2022a.service`
macOS	**Account Name:** `MwWebAppsServerR2022a` **Service Name:** `com.mathworks.mw-webapps-R2022a` **Service File:** `/Library/LaunchDaemons/com.mathworks.mw-webapps-R2022a.plist`	**Account Name:** `MwWebAppsWorkerR2022a` **Service Name:** `com.mathworks.mw-webapps-launcher-R2022a` **Service File:** `/Library/LaunchDaemons/com.mathworks.mw-webapps-launcher-R2022a.plist`

Groups

Operating System	Group Name	Description
Windows	`MwWebAppAuthorsR2022a`	Members of this group can upload applications to the server.
Linux	`MwWebAppsAuthorsR2022a`	Members of this group can upload applications to the server.
macOS	`MwWebAppsAuthorsR2022a`	Members of this group can upload applications to the server.

Folder Locations

Operating System	Folders
Windows	**Apps Folder:** `%ProgramData%\MathWorks\webapps\R2022a\apps` **Logs Folder:** `%ProgramData%\MathWorks\webapps\R2022a\logs` **Config Folder:** `%ProgramData%\MathWorks\webapps\R2022a\config`
Linux	**Apps Folder:** `/local/MathWorks/webapps/R2022a/apps` **Logs Folder:** `/local/MathWorks/webapps/R2022a/logs` **Config Folder:** `/local/MathWorks/webapps/R2022a/config`
macOS	**Apps Folder:** `/Library/Application Support/MathWorks/webapps/R2022a/apps` **Logs Folder:** `/Library/Application Support/MathWorks/webapps/R2022a/logs` **Config Folder:** `/Library/Application Support/MathWorks/webapps/R2022a/config`

Create Folders to Organize Web Apps

You can create folders within the `apps` root folder to organize your web apps. Only folders created at the root level are displayed on the web apps home page. Subfolders are not supported and are not displayed on the home page, even if they exist. You must use the `webapps-appdir` command line interface to create folders. Otherwise, any folders you create will not have the required permissions. While creating folders within the `apps` root folder is optional, doing so lets you organize your web apps and apply policy-based access to web apps contained within the folders. For more information, see "Policy-Based Access" on page 4-14.

```
/apps
    myApp00.ctf

—— folder_A/
    myApp01.ctf
    myApp02.ctf

—— folder_B/
    myApp03.ctf
    myApp04.ctf
    myApp05.ctf

—— folder_C/
    myApp04.ctf
    myApp05.ctf
    myApp06.ctf
    myApp07.ctf
```

See Also

More About

- "Install or Uninstall MATLAB Web App Server Product" on page 3-2
- "Policy-Based Access" on page 4-14

Upgrading to MATLAB Web App Server Product

Caution You cannot run the development version of MATLAB Web App Server in MATLAB Compiler and the MATLAB Web App Server product on the same machine.

To upgrade to the MATLAB Web App Server product from the development version of MATLAB Web App Server in MATLAB Compiler, you need to:

1 Unregister the services associated with the development version of MATLAB Web App Server by clicking **Unregister** in the server application.

2 Install the MATLAB Web App Server product. For more information, see "Install or Uninstall MATLAB Web App Server Product" on page 3-2.

3 Set up the MATLAB Web App Server product. For information, see "Set Up MATLAB Web App Server" on page 3-4.

Your `apps`, `config`, and `logs`, folders are retained during the upgrade process. For more information, see "Service Information, Groups, and Folder Locations" on page 3-10.

See Also

More About

- "Install or Uninstall MATLAB Web App Server Product" on page 3-2
- "Set Up MATLAB Web App Server" on page 3-4

Service Information, Groups, and Folder Locations

Setting up the MATLAB Web App Server creates two services and folders for uploading apps, capturing logs, and managing the server configuration.

Service Info

Operating System	Server Service Information	Apps Service Information
Windows	**Account Name:** `MwWebAppServerR2022a` **Service Name:** `mw-webapps-R2022a`	**Account Name:** `MwWebAppWorkerR2022a` **Service Name:** `mw-webapps-launcher-R2022a`
Linux	**Account Name:** `MwWebAppsServerR2022a` **Service Name:** `mw-webapps-R2022a` **Service File:** `/etc/systemd/system/mw-webapps-R2022a.service`	**Account Name:** `MwWebAppsWorkerR2022a` **Service Name:** `mw-webapps-launcher-R2022a` **Service File:** `/etc/systemd/system/mw-webapps-launcher-R2022a.service`
macOS	**Account Name:** `MwWebAppsServerR2022a` **Service Name:** `com.mathworks.mw-webapps-R2022a` **Service File:** `/Library/LaunchDaemons/com.mathworks.mw-webapps-R2022a.plist`	**Account Name:** `MwWebAppsWorkerR2022a` **Service Name:** `com.mathworks.mw-webapps-launcher-R2022a` **Service File:** `/Library/LaunchDaemons/com.mathworks.mw-webapps-launcher-R2022a.plist`

Groups

Operating System	Group Name	Description
Windows	`MwWebAppAuthorsR2022a`	Members of this group can upload applications to the server.
Linux	`MwWebAppsAuthorsR2022a`	Members of this group can upload applications to the server.
macOS	`MwWebAppsAuthorsR2022a`	Members of this group can upload applications to the server.

Folder Locations

Operating System	Folders
Windows	**Apps Folder:** `%ProgramData%\MathWorks\webapps\R2022a\apps` **Logs Folder:** `%ProgramData%\MathWorks\webapps\R2022a\logs` **Config Folder:** `%ProgramData%\MathWorks\webapps\R2022a\config`
Linux	**Apps Folder:** `/local/MathWorks/webapps/R2022a/apps` **Logs Folder:** `/local/MathWorks/webapps/R2022a/logs` **Config Folder:** `/local/MathWorks/webapps/R2022a/config`
macOS	**Apps Folder:** `/Library/Application Support/MathWorks/webapps/R2022a/apps` **Logs Folder:** `/Library/Application Support/MathWorks/webapps/R2022a/logs` **Config Folder:** `/Library/Application Support/MathWorks/webapps/R2022a/config`

See Also

More About

- "Set Up MATLAB Web App Server" on page 3-4

Configure Server to Use MATLAB Runtime

If you did not configure the MATLAB Web App Server to use a version of MATLAB Runtime during the setup process, you can do so using the `webapps-runtime` command. However, before configuring the server to use a version of MATLAB Runtime, verify that you have one installed.

Note MATLAB Runtime starting from R2019b up until the most recent release is supported. You can use multiple versions of the MATLAB Runtime with the server.

Install MATLAB Runtime

1 Download the MATLAB Runtime installer from the MathWorks website or the MATLAB desktop.

Option	Steps

MathWorks Website	Select the appropriate platform and release-specific installer from:
	
	`https://www.mathworks.com/`
	`products/compiler/matlab-`
	`runtime.html`

MATLAB Desktop	At the MATLAB command prompt, type: `compiler.runtime.download`

2 Install MATLAB Runtime using the installer. For installation instructions, see "Install and Configure MATLAB Runtime" (MATLAB Compiler).

Configure the Server

To configure the server to use a version of MATLAB Runtime:

1 Navigate to the command-line scripts folder.

Operating System	Default Location of Command-Line Scripts
Windows *(Administrator)*	`C:\Program Files\MATLAB\MATLAB Web App Server\R2022a\script`

Linux *(sudo)*	`/usr/local/MATLAB/`
	`MATLAB_Web_App_Server/R2022a/`
	`script`

macOS *(sudo)*	`/Applications/MATLAB/`
	`MATLAB_Web_App_Server/R2022a/`
	`script`

2 Execute the `webapps-runtime` command with the `add` option and a path to the MATLAB Runtime installation.

```
webapps-runtime add <Path to MATLAB Runtime installation>
```

See Also

More About

- "Set Up MATLAB Web App Server" on page 3-4

Customize MATLAB Web App Server Apps Home Page

To change the title, background color, font type, and font size of the banner in the apps home page, navigate to the `customization` folder and edit the `CustomConstants.json` and `custom-css.css` files.

Operating System	Default Location of Command-Line Scripts
Windows *(Administrator)*	C:\Program Files\MATLAB\MATLAB Web App Server\R2022a \resources\spfres\services\mdwas\home\customization
Linux *(sudo)*	/usr/local/MATLAB/MATLAB_Web_App_Server/R2022a/resources/ spfres/services/mdwas/home/customization
macOS *(sudo)*	/Applications/MATLAB/MATLAB_Web_App_Server/R2022a/ resources/spfres/services/mdwas/home/customization

Change the Banner Title

1   Open the `CustomConstants.json` file in a text editor.

```
{
    "WEB_APPS_HEADER": "MATLAB Web App Server"
}
```

2   Change the `"MATLAB Web App Server"` default title to a custom title. The maximum length of the title can be around 50 characters depending on the font type and size.

3   Save the file and refresh the apps home page.

Change the Background Color, Font Type, and Font Size of the Banner

1   Open the `custom-css.css` file in a text editor.

```
/*
-----------------------------------------------------------------
    CSS that can be customized to accomodate company branding
-----------------------------------------------------------------
*/

/* Banner Color */
.clientmdwas .headerBanner {
    background-color: rgb(32, 126, 178);

}

/* Banner Title Font and Font Size*/
/* Recommended default settings */
.clientmdwas .bannerTitle {
    font-family: Arial, cursive;
    font-size: 28px; /* Recommended range 20-30px, depending on font family used */
}
```

2   To change the banner color, set the `background-color` property to a valid CSS color. For details, see CSS Colors.

3   To change the font type, set the `font-family` property to a valid CSS font type. For details, see CSS Fonts.

4   To change the font size, set the `font-size` property to a valid CSS font size. The recommended range is 20-30px. For details, see CSS Font Size.

5   Save the file and refresh the apps home page.

See Also

More About

- "Set Up MATLAB Web App Server" on page 3-4

External Websites

- https://www.w3schools.com/css/css_colors.asp
- https://www.w3schools.com/css/css_font.asp
- https://www.w3schools.com/css/css_font_size.asp

Security

Enable SSL

Note

- MATLAB Web App Server only supports the `.pem` SSL certificate format.
- SSL keys must be 2048 bits in length and must be private.
- Intermediate certificates are not supported by the server.
- Private key should not be password protected.

To enable Secure Sockets Layer (SSL) on the server:

1 Obtain a certificate file and private key file for the server from a certificate authority and place these files in a location accessible to the MATLAB Web App Server.

2 At the terminal, navigate to the folder containing MATLAB Web App Server command-line scripts.

Operating System	Default Location of Command-Line Scripts
Windows *(Administrator)*	`C:\Program Files\MATLAB\MATLAB Web App Server\R2022a\script`
Linux *(sudo)*	`/usr/local/MATLAB/ MATLAB_Web_App_Server/R2022a/ script`
macOS *(sudo)*	`/Applications/MATLAB/ MATLAB_Web_App_Server/R2022a/ script`

3 At the terminal, type:

`webapps-config set ssl_certificate_file `*`<path to certificate file>`*

4 At the terminal, type:

`webapps-config set ssl_private_key_file `*`<path to private key file>`*

5 At the terminal, type:

`webapps-config set ssl_enabled true`

6 Restart the server.

Note The private key file must be in a location that is readable only by the server account.

A convenient location to store the certificate file and private key file is the `webapps_private` folder. For the location of the `webapps_private` folder, see "Authentication" on page 4-4.

See Also

More About

- "Authentication" on page 4-4
- "Role-Based Access" on page 4-10

External Websites

- How do I generate a self-signed certificate for testing HTTPS functionality in MATLAB Web App Server?

Authentication

> **Note** Authentication is supported in the standalone MATLAB Web App Server product and not the development version included in MATLAB Compiler. For details, see "MATLAB Web App Server Differences" on page 1-3.

Prerequisite

- Enable SSL on the server. For more information, see "Enable SSL" on page 4-2.

Create webapps_authn.json File

Authentication lets you validate a user's credentials and helps you control which users can access web apps deployed on the server.

MATLAB Web App Server supports authentication using Lightweight Directory Access Protocol (LDAP) and OpenID Connect (OIDC).

To enable authentication:

1. Check if SSL is enabled. For more information, see "Enable SSL" on page 4-2.
2. Create a file named `webapps_authn.json` and place it in the `webapps_private` folder, which is located within the `config` folder. The format for `webapps_authn.json` depends on whether you are using LDAP or OIDC for authentication.

    The `webapps_private` folder can be found in:

Operating System	Folder Location
Windows	`%ProgramData%\MathWorks\webapps\R2022a\config\webapps_private`
Linux	`/local/MathWorks/webapps/R2022a/config/webapps_private`
macOS	`/Library/Application Support/MathWorks/webapps/R2022a/config/webapps_private`

LDAP Authentication

An LDAP directory server stores information about users, groups, and applications. Each entry in the directory consists of three components: a distinguished name (DN), a collection of attributes, and a collection of object classes.

To use LDAP authentication, create a file named `webapps_authn.json` using the following JSON schema and place it in the `webapps_private` folder.

```
{
    "version": "1.2.0",
    "type": "ldap",
    "authnConfig": {
```

```
            "host": "<LDAP server host name>",
            "port": "<LDAP server port number>",
            "searcherDN": "",
            "searcherPassword": "",
            "baseDN": "<Point in LDAP from where to start search for a user>",
            "userFilter": "<Filter syntax>"
        },
        "appConfig": {
            "checkSSLCA": "<Boolean indicating whether to check for trusted SSL certificate>",
            "trustedSSLCA": "<Path to trusted SSL certificate>",
            "displayName": "<Identifier to display on MATLAB Web App Server home page>",
            "tokenExpirationMin": "<Token expiration duration in minutes>",
            "userAttributeName": "<Attribute name to identify user objects>",
            "groupAttributeName": "<Attribute name to identify group objects>"
        }
}
```

- **version**: Specify the version of the JSON schema. Default value for R2022a is: `1.2.0`.
- **type**: Specify the type of authentication to use. Set this value to `ldap`.
- **host**: Specify the LDAP directory server host name. For example: `myldap.myboston.com`.
- **port**: *(Optional)* Specify the LDAP directory server port number. For example: `389`. If a port number is not specified, the default port will be used. The MATLAB Web App Server uses SSL/STARTTLS to secure communication with the LDAP server. This ensures that usernames and passwords that are transmitted through an encrypted channel between MATLAB Web App Server and the LDAP server. By default, the server uses the standard port `636` for SSL on Windows and port `389` for STARTTLS on Linux and macOS. The LDAP server must be configured to allow SSL/STARTTLS connection over the specified (or default) LDAP port; otherwise, authentication will fail.
- **searcherDN**: Specify the searcher account's DN in the directory. The default value is `""`. Searcher DN refers the account allowed to search the LDAP directory server. For example: `"cn=admin,dc=myboston,dc=com"`.
- **searcherPassword**: Searcher account's password. The default value is `""`.

  MATLAB Web App Server uses the values for **searcherDN** and **searcherPassword** to search for a user's DN using a `userFilter`. The discovered DN is subsequently validated against with the entered password through LDAP. Values for **searcherDN** and **searcherPassword** are not required if the LDAP server provides access for anonymous authentication.

  Since the `webapps_authn.json` file lives within the `webapps_private` folder, which is only readable by the server account, the searcher's credentials are protected from apps or other users who log in to the server.
- **baseDN**: Specify the base DN in the directory. The base DN is the location in the directory where the application starts searching for a user. For example: `dc=myldap,dc=myboston,dc=com`.
- **userFilter**: Specify a filter to find a user's DN. MATLAB Web App Server uses **userFilter** to find the user's DN that matches the entered username, represented as *{username}* in the filter. If no match is found or multiple matches are found, authentication fails. The filter can be specified using standard LDAP filter syntax. For example: `(&(objectClass=User) (sAMAccountName={username}))`.
- **checkSSLCA**: Check whether the LDAP server's SSL certificate was signed by a recognized certificate authority (CA). Setting this property to `true` checks for a valid SSL certificate and setting it to `false` with forgo checking. If set to `true`, you need to specify a value for **trustedSSLCA**. If set to `false`, usernames and passwords are still transmitted between MATLAB

Web App Server and the LDAP server through an encrypted channel. However, this check is recommended for additional security.

- **trustedSSLCA**: On Linux and macOS systems, specify the path to the root certificate issued by the certification authority (CA) that signed the site certificate. On Windows systems, you do not need to specify the path. As long as the root certificate is in the *Trusted Root Certification Authorities certificate store*, MATLAB Web App Server will automatically find it.

- **displayName**: Configure how the user's identity is displayed on the MATLAB Web App Server home page by specifying an attribute of a user's LDAP entry. For example, setting this property to `uid` displays the user ID. Default is the username that is entered during the authentication process.

- **tokenExpirationMin**: Specify the token expiration duration in minutes. For example: `60`. Default value is `""`, which means the tokens do not expire.

- **userAttributeName**: Specify an attribute name to identify user objects. For example: `uid`. You must set a value for this property if you use policy-based access. Otherwise, do not include this property in your JSON file. For more information, see "Policy-Based Access" on page 4-14.

- **groupAttributeName**: Specify an attribute name to identify group objects. For example: `member`. You must set a value for this property if you use policy-based access. Otherwise, do not include this property in your JSON file. For more information, see "Policy-Based Access" on page 4-14.

Example webapps_authn.json File for LDAP

```
{
    "version": "1.2.0",
    "type": "ldap",
    "authnConfig": {
        "host": "myldap.myboston.com",
        "port": "",
        "searcherDN": "",
        "searcherPassword": "",
        "baseDN": "DC=myldap,DC=myboston,DC=com",
        "userFilter": "(&(objectClass=User)(sAMAccountName={username}))"
    },
    "appConfig": {
        "checkSSLCA": "false",
        "trustedSSLCA": "",
        "displayName": "uid",
        "tokenExpirationMin": "60"
    }
}
```

Example webapps_authn.json File for LDAP When Using Policy-Based Access

```
{
    "version": "1.2.0",
    "type": "ldap",
    "authnConfig": {
        "host": "myldap.myboston.com",
        "port": "",
        "searcherDN": "",
        "searcherPassword": "",
        "baseDN": "DC=myldap,DC=myboston,DC=com",
```

```
            "userFilter": "(&(objectClass=User)(sAMAccountName={username}))"
    },
    "appConfig": {
        "checkSSLCA": "false",
        "trustedSSLCA": "",
        "displayName": "uid",
        "tokenExpirationMin": "60",
        "userAttributeName": "uid",
        "groupAttributeName": "memberOf"
    }
}
```

OIDC Authentication

OpenID Connect (OIDC) allows MATLAB Web App Server to verify the identity of an end user based on the authentication performed by a third-party identity provider (IdP). To use OIDC authentication on the server, you need to register with an IdP such as Microsoft® Azure® AD, or Google® Identity Platform.

To use OIDC authentication, create a file named `webapps_authn.json` using the following JSON schema and place it in the `webapps_private` folder.

```
{
    "version": "1.2.0",
    "type": "oidc",
    "authnConfig": {
        "issuer": "<OIDC IdP issuer URI>",
        "clientId": "<Client ID from IdP>",
        "clientSecret": "<Client secret from IdP>",
        "redirectUrl": "<Redirection URL>",
        "scope": ["<scope1> <scope2>"]
    },
    "appConfig": {
        "port": "<OIDC authentication port number used by MATLAB Web App Server>",
        "displayName": "<Identity to display on MATLAB Web App Server home page>",
        "tokenExpirationMin": "<Token expiration duration in minutes>",
        "userAttributeName": "<Attribute name to identify user objects>",
        "groupAttributeName": "<Attribute name to identify group objects>"
    }
}
```

- **version**: Specify the version of the JSON schema. The default value for R2022a is: `1.2.0`.
- **type**: Specify the type of authentication to use. Set this value to `oidc`.
- **issuer**: Specify the OIDC IdP issuer URI. For example, if using Google Identity Platform: `https://accounts.google.com/.well-known/openid-configuration`.
- **clientId**: Specify the *client ID* you obtained while registering your credentials with an IdP. For example, if using Google Identity Platform: `1234567890-xxxxxxxxxxxx.apps.googleusercontent.com`.
- **clientSecret**: Specify the *client secret* you obtained while registering your credentials with an IdP. For example, if using Google Identity Platform: `_xxxxxxxxxxxxx_Xxxxxx_xX`.

  Since the `webapps_authn.json` file lives within the `webapps_private` folder, which is only readable by the server account, **clientId** and **clientSecret** are protected from apps or other users who log in to the server.

- **redirectUrl**: *(Optional)* Specify the redirect URL you used while configuring OIDC authentication with the IdP. If left empty, the host name and port number of the computer running the MATLAB Web App Server is used as a callback. The format of the URL is: *https:// <MATLABWebAppServer_hostname>:<port_server_is_running_on>/webapps/extauth/ callback*. For example, if MATLAB Web App Server is running on port `9988`, then the redirect URL is: `https://myboston.com:9988/webapps/extauth/callback`.

  Prior to R2021a, the format of the redirect URL was: *https:// <MATLABWebAppServer_hostname>:<port>/oidc/callback*. For example: `https:// myboston.com:3000/oidc/callback`.

- **scope**: Specify the identifiers for resources that an administrator wants MATLAB Web App Server to access. For example, if using Google Identity Platform: `openid profile email`.

- **port**: *(Optional)* Specify the port number used by the MATLAB Web App Server process internally for OIDC authentication. For example: `4000`. This port number must be different from the `redirectUrl` port number. If a port number is not specified, the MATLAB Web App Server process automatically picks a port for OIDC authentication.

- **displayName**: Configure how the user's identity is displayed on the MATLAB Web App Server home page, by specifying an attribute name of an authenticated user object. For example, if using Google Identity Platform, `given_name` displays the user's name. The default is the `sub` attribute.

- **tokenExpirationMin**: Specify the token expiration duration in minutes. For example: `60`. The default value is `""`, which means the tokens do not expire.

- **userAttributeName**: Specify an attribute name to identify user objects. For example: `uid`. You must set a value for this property if you use policy-based access. Otherwise, do not include this property in your JSON file. If you are not using policy-based access but decide to include this property in your JSON file, set the value to `""`. For more information, see "Policy-Based Access" on page 4-14.

- **groupAttributeName**: Specify an attribute name to identify group objects. For example: `member`. You must set a value for this property if you use policy-based access. Otherwise, do not include this property in your JSON file. If you are not using policy-based access but decide to include this property in your JSON file, set the value to `""`. For more information, see "Policy-Based Access" on page 4-14.

Note

1 If you use OIDC authentication, you need to register MATLAB Web App Server as an *application* with the IdP.

2 During the registration process, you need a redirect URL for MATLAB Web App Server. The format of the URL is: *https:// <MATLABWebAppServer_hostname>:<port_server_is_running_on>/webapps/extauth/ callback*. For example: `https://myboston.com:9988/webapps/extauth/callback`.

Example webapps_authn.json File for OIDC Using Google Identity Platform

```
{
    "version": "1.2.0",
    "type": "oidc",
    "authnConfig": {
```

```
        "issuer": "https://accounts.google.com/.well-known/openid-configuration",
        "clientId": "1234567890-xxxxxxxxxxxx.apps.googleusercontent.com",
        "clientSecret": "_xxxxxxxxxxxxx_Xxxxxx_xX",
        "redirectUrl": "https://myboston.com:9988/webapps/extauth/callback",
        "scope": ["openid profile email"]
    },
    "appConfig": {
        "port": "4000",
        "displayName": "given_name",
        "tokenExpirationMin": "60"
    }
}
```

Tip After setting up authentication, if you are unable to login from your browser, try clearing your browser's cache and cookies, or try a different browser.

Caution The JSON schema syntax for `webapps_authn.json` is strictly enforced. Errors in the schema syntax may result in the server not starting, or being denied access to the server when you try to login.

See Also

More About
- "Role-Based Access" on page 4-10
- "Policy-Based Access" on page 4-14

External Websites
- https://developers.google.com/identity/protocols/oauth2/openid-connect
- https://www.oauth.com/oauth2-servers/openid-connect

Role-Based Access

> **Note** Role-based access is supported in the standalone MATLAB Web App Server product and not the development version included in MATLAB Compiler. For details, see "MATLAB Web App Server Differences" on page 1-3.

Prerequisites

- Enable SSL on the server. For more information, see "Enable SSL" on page 4-2.
- Enable authentication on the server. For more information, see "Authentication" on page 4-4.

Create webapps_app_roles.json File

Enabling role-based access on the server lets you decide which users can author apps and which ones can use them.

MATLAB Web App Server supports two roles for role-based access: *Author* and *User*.

- An *Author* can add, delete, and run web apps from MATLAB Web App Server. An *Author* sees a **Manage Apps** button on the server home page.
- A *User* can only run web apps from the MATLAB Web App Server home page. A *User* sees a **Diagnostics** button on the server home page.

You can use role-based access along with policy-based access to finely determine who can run apps on the server and who can modify them. For details, see "Policy-Based Access" on page 4-14.

To enable role-based access:

1 Check if SSL is enabled. For more information, see "Enable SSL" on page 4-2.

2 Check if authentication is enabled. For more information, see "Authentication" on page 4-4.

3 Create a file named `webapps_app_roles.json` and place it in the `webapps_private` folder.

The `webapps_private` folder can be found in:

Operating System	Folder Location
Windows	`%ProgramData%\MathWorks\webapps\R2022a\config\webapps_private`
Linux	`/local/MathWorks/webapps/R2022a/config/webapps_private`
macOS	`/Library/Application Support/MathWorks/webapps/R2022a/config/webapps_private`

The JSON schema for `webapps_app_roles.json` is:

```
{
  "version": "1.0.0",
  "appRoles": [
    {
```

```
      "id": "User",
      "description": <Text describing the User role>,
      "users":  { <Attribute name and values to identify end-users assigned to User role> },
      "groups": { <Attribute name and values to identify groups assigned to User role> }
    },
    {
      "id": "Author",
      "description": <Text describing the Author role>,
      "users":  { <Attribute name and values to identify end-users assigned to Author role> },
      "groups": { <Attribute name and values to identify groups assigned to Author role> }
    }
  ]
}
```

- **version**: Specify the version of the JSON schema. The default value for R2022a is: `1.0.0`.
- **id**: Specify the role name. You can specify either `User` or `Author`. Only these two roles are supported.
- **description**: Specify a description for each role. For example:

  ```
  "description" : "An Author can upload, delete, and execute web apps."
  ```
- **users**: Specify an attribute that uniquely identifies the set of authenticated end users who can assume the role of an *Author* or a *User*.

  The attribute names depend on the type of authentication you are using.

  For example, if you are using LDAP for authentication, you can fill in the JSON schema as follows:

  ```
  "users":{ "email": ["bishop@myboston.com", "queen@myboston.com"] }
  ```

  In the above schema, once an end-user is authenticated, MATLAB Web App Server checks if the authenticated user has `email` as an attribute, and checks to see if the attribute value (email address in this case) is listed in the schema. When both checks succeed, the end-user will be assigned a role.
- **groups**: Specify an attribute name and corresponding values that uniquely identify the group of authenticated end users who can assume the role of an *Author* or a *User*.

  The attribute names depend on the type of authentication you are using. Using **groups** lets you assign entire sets of end-users a role at once.

  For example, if you are using LDAP for authentication, you can fill in the JSON schema as follows:

  ```
  "groups": { "memberOf": [ "CN=Marketing,OU=Mail,DC=ldap,DC=myboston,DC=com",
                            "CN=Development,OU=Mail,DC=ldap,DC=myboston,DC=com"] }
  ```

  In the above schema, once an end-user is authenticated, MATLAB Web App Server checks if the authenticated user has `memberOf` as an attribute, and checks to see if the attribute's values are listed in the schema. When both checks succeed, the end-user will be assigned a role.

  Attributes specified in the schema need to be collective or group attributes.

Tip

1   You do not need to specify both `users` and `groups` in the schema for each role unless that is the only way to obtain a unique set of end users.

2   If you use an attribute in the `users` field in the *User* role to identify a set of users, you need use the same attribute in the `users` field in the *Author* role to identify a set of users. The same condition applies to `groups` as well.

MATLAB Web App Server first checks if an authenticated user can assume the role of an *Author* before checking the *User* role. If checks against both roles fails, the end-user is denied access to the server.

Example webapps_app_roles.json File for LDAP Authentication

```
{
    "version": "1.0.0",
    "appRoles": [
        {
            "id": "User",
            "description": "A User can only execute web apps.",
            "groups": {
                "memberOf": [
                    "CN=Marketing,OU=Mail,DC=ldap,DC=myboston,DC=com",
                    "CN=Development,OU=Mail,DC=ldap,DC=myboston,DC=com"
                ]
            }
        },
        {
            "id": "Author",
            "description": "An Author can upload, delete, and execute web apps.",
            "users": { "email": [
                    "bishop@myboston.com",
                    "queen@myboston.com"
                ]
            }
        }
    ]
}
```

Example webapps_app_roles.json File for Azure AD Authentication

```
{
    "version": "1.0.0",
    "appRoles": [
        {
            "id": "User",
            "description": "A User can only execute web apps.",
            "groups": {
                "groups": [
                    "1a23456-ab2c-4444-a123-12345b3a81af",
                    "2b3456cd-e8ed-4fcf-ac55-6b79b0781eed "
                ]
            }
        },
        {
            "id": "Author",
```

```
            "description": "An Author can upload, delete, and execute web apps.",
            "users": { "email": [
                    "bishop@myboston.com",
                    "queen@myboston.com"
                ]
            }
        }
    ]
}
```

Caution The JSON schema syntax for `webapps_app_roles.json` is strictly enforced. Errors in the schema syntax may result in the server not starting, or you being denied access to the server when you try to log in.

See Also

More About

- "Authentication" on page 4-4
- "Policy-Based Access" on page 4-14

Policy-Based Access

Note Policy-based access is supported in the standalone MATLAB Web App Server product and not the development version included in MATLAB Compiler. For details, see "MATLAB Web App Server Differences" on page 1-3.

Prerequisites

- Enable SSL on the server. For more information, see "Enable SSL" on page 4-2.
- Enable authentication on the server. For more information, see "Authentication" on page 4-4.

Create webapps_acc_ctl.json File

Enabling policy-based access on the server lets you create attribute-based rules that permit authenticated subjects to access web apps on the server.

To enable policy-based access:

1 Check if SSL is enabled. For more information, see "Enable SSL" on page 4-2.

2 Check if authentication is enabled and verify that either or both the `userAttributeName` and `groupAttributeName` values have been specified in the `webapps_authn.json` file. For details, see "Authentication" on page 4-4.

3 Check if folders exist within the `apps` root folder on the server. While folders are not necessary to enable policy-based access, having folders lets you organize web apps and specify them as a resource that can be accessed. For details, see "Create Folders to Organize Web Apps" on page 3-7.

4 Create a file named `webapps_acc_ctl.json` and place it in the `webapps_private` folder.

The `webapps_private` folder can be found in:

Operating System	Folder Location
Windows	`%ProgramData%\MathWorks\webapps\R2022a\config\webapps_private`
Linux	`/local/MathWorks/webapps/R2022a/config/webapps_private`
macOS	`/Library/Application Support/MathWorks/webapps/R2022a/config/webapps_private`

The JSON schema for `webapps_acc_ctl.json` is:

```
{
  "version": "<major>.<minor>.<patch>",
  "policy": [
    {
      "id": "<policy_id>",
      "description": "<policy_description>",
      "rule": [
        {
```

```
          "id": "<rule_id>",
          "description": "<rule_description>",
          "subject": { "<attribute_name_of_auth_subject>": [ "<attribute_value_of_auth_subject>"
          "resource": { "<app OR folder>": [ "<app_name OR folder_name>" ] },
          "action": [ "<action_type>" ]
        },
        {
          "id": "<rule_id>",
          "description": "<rule_description>",
          "subject": { "<attribute_name_of_auth_subject>": [ "<attribute_value_of_auth_subject>"
          "resource": { "<app OR folder>": [ "<app_name OR folder_name>" ] },
          "action": [ "<action_type>" ]
        }
      ]
    }
  ]
}
```

- **version**: Specify the version of the JSON schema. The default value for R2022a is `1.0.0`.

- **policy**: The policy block contains a list of policies required for policy-based access. Only a single policy can be specified in a policy file.

- **id**: Specify a policy ID for the policy. The policy ID must be a set of alphanumeric characters. Any leading or trailing white space is removed. For example:

  ```
  "id" : "policy420"
  ```

- **description**: Specify a description for the policy. For example:

  ```
  "description" : "Company policy for accessing web apps."
  ```

- **rule**: The rule block contains a list of rule objects. Multiple rules can exist in a rule block. An ID is required for each rule and *<rule_id>* must be unique for each rule.

- **id**: Specify a unique rule ID for each rule. The rule ID must be a set of alphanumeric characters. Any leading or trailing white space is removed. For example:

  ```
  "id" : "rule101"
  ```

- **description**: Specify a description for each rule. For example:

  ```
  "description" : "Only the Hercules group can run the BloodPressure app."
  ```

- **subject**: Specify an attribute name-value pair of an authenticated subject that can access resources.

  For example:

  ```
  "subject": { "memberOf": ["CN=Middle,OU=middle,ou=groups,DC=school,DC=com"] }
  ```

  ```
  "subject": { "groups": ["Sales"] }
  ```

  ```
  "subject": { "uid": ["fbueller", "cfrye"]  }
  ```

  If you are using an Azure Active directory and need to retrieve group IDs, see "Using Azure Active Directory" on page 4-29.

- **resource**: Specify the type and name of the resources that can be accessed. Only resources of type `app` or `folder` are supported. If you do not specify a resource, then an authenticated subject will not have access to the app or folder even if it exists on the server. For information on how to create folders, see "Create Folders to Organize Web Apps" on page 3-7.

For example, to specify access to an app at the root level:

`"resource": { "app": ["BloodPressure"] }`

For example, to specify access to an app in a particular folder:

`"resource": { "app": ["MagicFolder/CardTricks"] }`

For example, to specify access to all apps in a particular folder:

`"resource": { "folder": ["MagicFolder"] }`

For example, to specify access to all apps in the root folder:

`"resource": { "folder": ["/"] }`

- **action**: Specify the type of action the authenticated subject can perform. If the resource type is `app`, the action supported is `execute`, which lets the subject run a web app. If the resource type is `folder`, the actions supported are `execute` and `modify`. The `execute` action, in this case, lets the subject run all the web apps in a specified folder. The `modify` action lets the subject upload or delete a web app to or from a folder if the subject is assigned the role of Author in the `webapps_app_roles.json` role-based access file. A subject assigned the role of User in the `webapps_app_roles.json` role-based access file cannot upload or delete a web app to or from a folder even if they are assigned a `modify` action. For example:

`"action": ["execute", "modify"]`

For details, see "Role-Based Access" on page 4-10.

Example webapps_acc_ctl.json File for LDAP Authentication

```
{
  "version": "1.0.0",
  "policy" : [
    {
      "id": "policy1001",
      "description": "Web Apps Access Control Policy",
      "rule": [
        {
          "id": "rule101",
          "description": "Sales group can run the BloodPressure app.",
          "subject": { "memberOf": ["cn=Sales,ou=sales,ou=groups,dc=myboston,dc=com"] },
          "resource": { "app": ["BloodPressure"] },
          "action": ["execute"]
        },
        {
          "id": "rule102",
          "description": "Specified subjects can run the CardTricks app in the MagicDir folder."
          "subject": { "uid": ["erooney"]  },
          "resource": { "app": ["MagicDir/CardTricks"] },
          "action": ["execute"]
        },
        {
          "id": "rule103",
          "description": "Specified subjects can run all apps in the MagicDir folder and modify
          "subject": { "uid": ["fbueller"]  },
          "resource": { "folder": ["MagicDir"] },
          "action": ["execute", "modify"]
```

```
      },
      {
        "id": "rule104",
        "description": "Specified subjects can run all apps under the DayOff folder.",
        "subject": { "uid": ["cfrye", "psloane"]  },
        "resource": { "folder": ["DayOff"] },
        "action": ["execute"]
      },
      {
        "id": "rule105",
        "description": "Specified subjects can run all apps in the apps root folder and modify
        "subject": { "uid": ["jbueller"]  },
        "resource": { "folder": ["/"] },
        "action": ["execute", "modify"]
      }
    ]
  }
 ]
}
```

Caution The JSON schema syntax for `webapps_acc_ctl.json` is strictly enforced. Errors in the schema syntax may result in the server not starting, or you being denied access to the server when you try to log in.

Using Policy-Based Access with Authentication and Role-Based Access

Policy-Based Access with Authentication

- If you use policy-based access, you must include the following properties in the `appConfig` block of the `webapps_authn.json` authentication file and set appropriate values:

  - `userAttributeName`
  - `groupAttributeName`

  A failure to include and set values for these properties results in the server not starting. For details, see "Authentication" on page 4-4.

- If you use policy-based access, and set values for `userAttributeName` and `groupAttributeName` in the `webapps_authn.json` authentication file, then the attributes specified for the `subject` property in the `webapps_acc_ctl.json` policy-based access file must match the values you set in the authentication file.

Authentication File (webapps_authn.json)	Policy-Based Access File (webapps_acc_ctl.json)
`"userAttributeName": "<value>",` `"groupAttributeName": "<value>"`	`"subject": { "<attrib>": ["..."] }` `"subject": { "<attrib>": ["..."]  }`

For example, if you set the following values in the `webapps_authn.json` authentication file:

```
"userAttributeName": "uid",
"groupAttributeName": "memberOf"
```

Then, the `webapps_acc_ctl.json` policy-based access file must use the same values:

```
"subject": { "uid": ["..."] }
```

```
"subject": { "memberOf": ["..."]  }
```

A mismatch of attributes in the JSON files results in the server not starting. For details, see
"Policy-Based Access" on page 4-14.

Note You can use policy-based access with authentication independent of role-based access.
However, this type of access results in authenticated subjects being able to only execute web apps
but not modify them.

Policy-Based and Role-Based Access with Authentication

- If you use policy-based access and role-based access simultaneously, and set values for
  `userAttributeName` and/or `groupAttributeName` in the `webapps_authn.json`
  authentication file, then the attributes specified for the `subject` property in the
  `webapps_acc_ctl.json` policy-based access file and the attributes specified for the `groups` and
  `users` properties must match the values you set in the authentication file.

Authentication File (webapps_authn.json)	Policy-Based Access File (webapps_acc_ctl.json)	Role-Based Access File (webapps_app_roles.json)
"userAttributeName": "<*value*>" "groupAttributeName": "<*value*>"	"subject": { "<*attrib*>": ["..."] "subject": { "<*attrib*>": ["..."]	"users": { "<*attrib*>": ["..."] } "groups": { "<*attrib*>": ["..."] }

For example, if you set the following values in the `webapps_authn.json` authentication file:

```
"userAttributeName": "uid",
"groupAttributeName": "memberOf"
```

Then, the `webapps_app_roles.json` role-based access file must use the same attributes:

```
"appRoles": [
        {
          "id": "User",
          "description": "User role info",
          "groups": { "memberOf": ["..."] },
          "users": { "uid": ["..."] }
        },
        {
          "id": "Author",
          "description": "Author role info",
          "groups": { "memberOf": ["..."] },
          "users": { "uid": ["..."] }
        }
        ]
```

And, the `webapps_acc_ctl.json` policy-based access file must use the same values:

```
"subject": { "memberOf": ["..."]  }
```

```
"subject": { "uid": ["..."] }
```

A mismatch of values results in the server not starting. For details, see "Role-Based Access" on
page 4-10.

Note When you use policy-based access and role-based access with authentication, an authenticated subject must be assigned the role of *Author* in the `webapps_app_roles.json` role-based access file and have `modify` as an action in the `webapps_acc_ctl.json` policy-based access file in order for the subject to be able to modify web apps.

See Also

More About
- "Enable SSL" on page 4-2
- "Authentication" on page 4-4
- "Role-Based Access" on page 4-10
- "Create Folders to Organize Web Apps" on page 3-7
- "Using Azure Active Directory" on page 4-29

Customize Web App Behavior Based on User

Note The ability to customize web app behavior based on the user is supported in the standalone MATLAB Web App Server product and not the development version included in MATLAB Compiler. For details, see "MATLAB Web App Server Differences" on page 1-3.

Prerequisites

- Enable SSL on the server. For more information, see "Enable SSL" on page 4-2.
- Enable authentication on the server. For more information, see "Authentication" on page 4-4.

Create userinfo.json File

You can customize the behavior of a web app based on which user is logged in. To customize behavior:

1   Create a file named `userinfo.json` and place it in the `webapps_private` folder on the server.

    The `webapps_private` folder is in:

Operating System	Folder Location
Windows	`%ProgramData%\MathWorks\webapps\R2022a\config\webapps_private`
Linux	`/local/MathWorks/webapps/R2022a/config/webapps_private`
macOS	`/Library/Application Support/MathWorks/webapps/R2022a/config/webapps_private`

2   While authoring your web app using App Designer in MATLAB, use the `compiler.UserInfo` function in your app code to retrieve user-specific details from the `userinfo.json` file.

The JSON schema for `userinfo.json` is:

```
{
    "version": "<major>.<minor>.<patch>",
    "userInfo.doc": "Property values to be fetched during login from IdP",
    "userInfo": {
      "UserID": "<uid_or_upn>",
      "DisplayName": "<user_name_that_is_displayed>",
      "Groups": "<group_membership_of_user>",
      "<propertyName1>": "<value1>",
      "<propertyName2>": "<value2>",
      "<propertyName3>": "<value3>",
      ...
    },
    "appAccess.doc": "Policy for allowing access to user properties within an app or group of app
    "appAccess": {
      "<appName>": ["<userInfo_propertyName>","<userInfo_propertyName>", ...],
      ...
      "*": "*"
```

```
        }
    }
```

- **version**: Specify the version of the JSON schema. The default value for R2022a is `1.0.0`.
- **userInfo.doc**: Text describing the purpose of the `userInfo` block.
- **userInfo**: The `userInfo` block contains a list of property names and values that help identify users. The property names `UserID`, `DisplayName`, and `Groups` are required in every `userinfo.json` file. Other property names and values can be included as necessary. Property names and values correspond to LDAP or OIDC attributes. For example, if you want to use an email address as part of a user's information, you can specify `Email` as a property name and attribute value for the `EmailAddress` as the property value.
- **UserID**: Specify the LDAP or OIDC attribute type that corresponds to a user's ID as a property value. `UserID` is a required property name. For example:

  `"UserID": "uid"`

  If you do not specify an attribute type as a property value to the `UserID` property name, `compiler.UserInfo`, which queries user details, returns `<missing>` as a property value.
- **DisplayName**: Specify the LDAP or OIDC attribute type that corresponds to a user's preferred name. For example:

  `"DisplayName": "displayName"`
- **Groups**: Specify the LDAP or OIDC attribute type that corresponds to the group that the user belongs to. For example:

  `"DisplayName": "groups"`

Note `UserID`, `DisplayName`, and `Groups` property names are required in every `userinfo.json` file. You can add custom property names to the `userInfo` block and assign property values based on LDAP or OIDC attribute types. For example, if you want to use an email address as part of a user's information, you can specify `Email` as a property name and the LDAP attribute type `EmailAddress` (or `mail` if using OIDC) as a property value.

- **appAccess.doc**: Text describing the purpose of the `appAccess` block.
- **appAccess**: The `appAccess` block contains a list of property names that correspond to app names hosted on the server and property values that correspond to property names from the `userInfo` block. You can set access to an app by specifying a combination of property names from the `userInfo` block to identify a unique set of users.
- **<appName>**: Specify an app name as the property name and a combination of property names from the `userInfo` block as property values to uniquely identify a set of users who can access the app. For example:

  `"BloodPressure": ["UserID", "Email"]`

Tip

- You can use an asterisk (*) wildcard character as both a property name and property value to indicate that all apps can be accessed by all users. For example:

  `"*": "*"`
- The property names `WebAppsRole` and `WebAppsDisplayName` are reserved and cannot be used in the `userInfo` block. However, they can be used in the `appAccess` block as property values. For example:

```
    "Mystery": ["UserID", "Email", "WebAppsRole", "WebAppsDisplayName"]
```

- • `WebAppsRole` corresponds to the user's role: *Author* and *User*. For details, see "Role-Based Access" on page 4-10.
- • `WebAppsDisplayName` corresponds to the name displayed on the apps home page.

- • If you make any changes in the `userInfo` block, you must restart the server. For details, see `webapps-restart`.

Example Using the userinfo.json File and compiler.UserInfo Function

In the following sample `userinfo.json` file the `userInfo` block contains the required property names: `UserID`, `DisplayName`, and `Groups`. In addition, it contains two custom property names, `LastName` and `Email`. All property names are assigned OIDC attributes as property values.

The `appAccess` block contains three apps: `BloodPressure`, `Mortgage`, and `Mystery`.

- • Access to the `BloodPressure` app is restricted based on `UserID` and `DisplayName` properties from the `userInfo` block.
- • Access to the `Mortgage` app is restricted based on `UserID` and `LastName` properties from the `userInfo` block.
- • Access to the `Mystery` app is restricted based on `UserID` and `DisplayName`, and the reserved property names `WebAppsRole` and `WebAppsDisplayName`.

```
{
    "version": "1.0.0",
    "userInfo.doc": "Property values to be fetched during login from IdP",
    "userInfo": {
      "UserID": "upn",
      "DisplayName": "displayName",
      "Groups": "groups",
      "LastName": "surname",
      "Email": "mail"
    },
    "appAccess.doc": "Policy for allowing access to user properties within an app or group of apps",
    "appAccess": {
      "BloodPressure": ["UserID","Email"],
      "Mortgage": ["UserID","LastName"],
      "Mystery": ["UserID","Email","WebAppsRole","WebAppsDisplayName"]
    }
  }
```

Given the `userinfo.json` file above, the `BloodPressure` app can use the `compiler.UserInfo` function within the app code as follows:

```
function startupFcn(app)
try
    user = compiler.UserInfo();
catch me
    error(me.message);
    return
end

if ~ismissing(user.UserID)
    % app code
    % Example:
    % app.userIDLabel.Text = [app.userIDLabel.Text user.UserID];
end
if isprop(user, 'Email')
```

```
    % app code
    % Example:
    % app.EmailLabel.Text = [app.EmailLabel.Text user.Email];
end
...
```

Given the `userinfo.json` file above, the `Mystery` app can use the `compiler.UserInfo` function within the app code as follows:

```
function startupFcn(app)
try
    user = compiler.UserInfo();
catch me
    error(me.message);
    return
end

if isprop(user, 'WebAppsDisplayName')
    % app code
    % Example:
    % app.DisplayNameLabel.Text = [app.DisplayNameLabel.Text user.WebAppsDisplayName];
end
if isprop(user, 'WebAppsRole')
    % app code
    % Example:
    % app.RoleLabel.Text = [app.RoleLabel.Text user.WebAppsRole];
end
...
```

See Also

compiler.UserInfo

MATLAB Web App Server Security

> **Caution** It is strongly recommended that you consult with your IT system administrator and discuss the security implications of installing the MATLAB Web App Server.

Installing and running the server on your network exposes your network and file system to risks. The machine running the server is most at risk from accidental or deliberate misuse of deployed web applications. Therefore, you must install the server software only on dedicated hardware. This machine can be a physical or virtual machine whose only purpose is to host web applications that connect to the server software. Using a physical or virtual machine limits the risk in the event that the machine is compromised.

Setting up MATLAB Web App Server creates two low-privileged user accounts on the host machine—one for the server and one for applications. However, you can choose to use the same account. However, using the same account can introduce additional risks. In addition, through a process known as *privilege escalation*, attackers may be able to exploit bugs in the operating system or network to obtain the privileges of ordinary or even administrative users. They can then attempt to access files or other intellectual property without permission.

You may be able to mitigate some of these risks by taking these precautions:

- *Restrict network access.* Only trusted users can access the server and its associated applications.
- *Execute only trusted applications.* Trust applications developed by only well-known, trusted, and authenticated sources.
- *Limit application functionality.* Include in the application only those features of MATLAB required for the application to perform its function. For more information, see "Authoring Secure Web Apps" on page 4-26.

For a list of additional risks, see "Potential Risks" on page 4-25.

See Also

More About
- "Potential Risks" on page 4-25
- "Enable SSL" on page 4-2
- "Authoring Secure Web Apps" on page 4-26
- "Securely Deploying Web Apps" on page 4-28

Potential Risks

- The MATLAB Web App Server has no specific mechanism to prevent HTTP request capture and replay.
- Installation of the MATLAB Web App Server creates two low-privileged user accounts on the host machine.

  These low-privileged accounts may inherit privileges given to all users. Care should be taken to restrict privileges given to all users.
- While the server and applications run under two different low-privileged user accounts, all applications hosted by the server run under the same low-privileged user account.

  If multiple copies of the same application run simultaneously, they might interfere with each other. This situation happens if the application writes data to any shared resource, for example, a file or a non-concurrent database.
- When deploying multiple applications to the server, the server shares cookies across sessions, which can result in crosstalk between applications for a single user accessing more than one application.

  This situation could allow unintentional crosstalk between multiple applications run by the same user.
- Deployed web applications are potentially vulnerable to *data* or *code injection* attacks whereby malicious or malformed inputs can be used to attempt to subvert the system. The server does not contain explicit protection against either type of injection attack. Certain MATLAB features, particularly the `eval()` function, can increase the risk of injection attacks. A common countermeasure is input sanitization or input whitelisting. MATLAB contains functions like `regexp` and `regexprep` that can assist in validating untrusted input.

  - Your application may indirectly call `eval()`, potentially making it vulnerable to code-injection attacks.
  - Other MATLAB functions may exhibit the same code injection vulnerabilities; any function that processes code-like input (XML, SQL, JSON, to name a few) is potentially vulnerable to code injection.
  - Any application that accesses the operating system via MATLAB `system()`, `dos()`, or `unix()` commands might also be vulnerable to code injection.
- Authentication tokens can potentially be abused.

Note This list identifies known risks and is not meant to be comprehensive.

See Also

More About

- "Enable SSL" on page 4-2
- "Authoring Secure Web Apps" on page 4-26
- "Securely Deploying Web Apps" on page 4-28
- "MATLAB Web App Server Security" on page 4-24

Authoring Secure Web Apps

Most of the potential risk associated with deploying web apps comes from the code in each app. By limiting the features that your app uses and by following the secure coding practices listed here, you can reduce the potential risk to your applications.

Authentication and Authorization

If your app requires access to sensitive data or performs potentially dangerous actions, you can consider implementing your own authentication and authorization schemes. Consult your network security group for advice.

Do Not Call eval()

The MATLAB `eval()` function turns text strings into commands. This powerful function allows users to execute arbitrary MATLAB code. This code can, in turn, enable execution of any installed program that is available to the low-privileged user or access to any file or data that the low-privileged users has access to. Applications created for web deployment and access must not contain calls to `eval()`. See "Alternatives to the eval Function" (MATLAB) for ways to eliminate `eval()` from your web app code. Relying on input sanitization can help mitigate the risk of any indirect calls to `eval()`. See "Sanitize User Input" on page 4-26.

Limit Free-Text User Input

Use menus, sliders, dials, and buttons instead of editable text fields in your app user interface. In addition to providing a better user experience, this practice limits the types of input users can provide, and the risks such inputs might introduce.

Sanitize User Input

To a security expert, user supplied data is considered untrusted because user input is a common attack vector for hackers. If your app must accept free-text input, the app must carefully examine the input for potential code injection attacks—text that contains special characters that coerce the app to interpret the input as commands rather than data.

In MATLAB, code injection attacks are most likely to be directed against XML, JSON, SQL, and other similar types of structured input. If your app accepts structured input, consult your IT or security group for suggestions on how to sanitize that input. It is never a good idea to allow the user to directly enter any type of code (such as MATLAB, Java®, or JavaScript®) for immediate evaluation.

Sanitize Data Read from Files

Reading data from files exposes the app to the same types of risk as collecting interactive user input. The same countermeasures apply. Also, you can protect read-only data files from tampering by a cryptographically secure hashing algorithm to digitally fingerprint files.

Minimize File System Write Operations

Limiting your application to read-only access greatly reduces the potential risk associated with your application. If you must write to the file system, remember that the server runs multiple copies of

your application simultaneously if multiple users access it at the same time. You must manage simultaneous writes either through the use of runtime-generated unique file names or the use of a database that can typically handle multiple simultaneous accesses. If simultaneous writes are not properly managed, data corruption may occur.

Verify Trustworthiness of Third-Party Code

If your app includes MATLAB files, shared libraries, Java classes, or any other type of code developed by a third party, you must make sure that code is free of viruses, worms, Trojan horses, and other web-based attack and infiltration vectors. You can discuss this issue with the author of the code and your IT and security staff. In the case of binaries or Java classes, consider running a virus scanner or other security software on the code before including it in your deployed app.

Reduce Exposure Level

One way to reduce exposure is to limit the time that the app is running to only those times when it is needed. For example, do not run it continuously from your desktop.

See Also

More About
- "Potential Risks" on page 4-25
- "Enable SSL" on page 4-2
- "Securely Deploying Web Apps" on page 4-28
- "MATLAB Web App Server Security" on page 4-24

Securely Deploying Web Apps

- Install the MATLAB Web App Server on a dedicated physical or virtual machine, and do not use this machine for any other purpose.
- Run web apps behind your organization firewall. Do not allow access from the open Internet.
- Install web apps only from trusted and verified people and organizations.
- Limit the features and functionality you build into the web apps you develop.

    - Do not call the MATLAB function `eval()`.
    - Avoid free-text input where you can, and use menus, lists, buttons, and other affordances instead.
    - Sanitize input from the app user interface and data files.
    - Limit file, network, and other resource access to the minimum required by your app.
    - Verify the trustworthiness of any third-party code included in your app.

- If your application accesses sensitive data, use in-application authentication to limit access.
- Reduce exposure level by limiting the time that the app runs to only those times when it is needed. For example, do not run it 24 hours a day, 7 days a week from your desktop.

See Also

More About

- "Authoring Secure Web Apps" on page 4-26
- "Potential Risks" on page 4-25
- "Enable SSL" on page 4-2
- "MATLAB Web App Server Security" on page 4-24

Using Azure Active Directory

Get Group ID from Azure Active Directory Based on Group Display Name

1 Login to the Azure AD Graph Explorer at:

https://graphexplorer.azurewebsites.net

2 Select GET as your HTTP query method.

3 Specify the following query in the text box:

```
https://graph.windows.net/<tenant>/groups?
$filter=startswith(displayName,'<groupname>')
```

Replace *<tenant>* with your tenant name and *<groupname>* with the group name whose ID you are retrieving.

4 Search for objectId of the specific group in the response. This is your group ID.

Get All Group IDs for a Certain User from Azure Active Directory

1 Login to the Azure AD Graph Explorer at:

https://graphexplorer.azurewebsites.net

2 Select GET as your HTTP query method.

3 Specify the following query in the text box:

```
https://graph.windows.net/<tenant>/users/<username>@<tenant>/memberOf
```

Replace *<tenant>* with your tenant name and *<username>* with the user name whose group IDs you are retrieving.

4 Search for objectId in the response for all groups where securityEnabled is set to true. These are your group IDs for a user.

See Also

More About

• "Policy-Based Access" on page 4-14

External Websites

• https://docs.microsoft.com/en-us/azure/active-directory/develop/microsoft-graph-intro

Cloud Deployment

Run MATLAB Web App Server on AWS Using Reference Architecture

You can deploy MATLAB Web App Server to AWS using a customizable reference architecture.

Requirements

- A MATLAB Web App Server license that meets the following conditions:

  - Current on Software Maintenance Service (SMS).

  - Linked to a MathWorks Account.

  - Concurrent license type. To check your license type, view your MathWorks Account.

  - Configured to use a network license manager on the virtual network. By default, the deployment of MATLAB Production Server includes a network license manager, but you can also use an existing license manager. In either case, activate or move the license after deployment. For details, see "Configure MATLAB Web App Server License for Use on the Cloud" on page 5-3.

- An AWS account. You are responsible for the costs of all AWS services.

- An SSH Key Pair for your AWS account in the appropriate region. For more information, see Amazon EC2 Key Pairs.

Run from GitHub

To launch MATLAB Web App Server in AWS, use the reference architecture templates provided in the following GitHub® repository:

- MATLAB Web App Server on AWS

You can run the template directly from the link in the GitHub repository. Choose your MATLAB Web App Server release, and then click the **Launch Stack** icon to deploy your resources.

See Also

More About

- "Configure MATLAB Web App Server License for Use on the Cloud" on page 5-3

External Websites

- https://github.com/mathworks-ref-arch/matlab-web-app-server-on-aws

Configure MATLAB Web App Server License for Use on the Cloud

This document shows administrators how to set up a MATLAB Web App Server license for use on the cloud. If you are an end user, contact your administrator. To use the reference architecture, you must have R2021b or later.

If you have questions about any part of this licensing configuration process, contact MathWorks Cloud Support.

License Configuration Requirements

To activate your license for use on the cloud:

- You must be an administrator of a MATLAB Web App Server license. If you do not have a MATLAB Web App Server, you can request a trial by contacting your MathWorks sales representative. If you are not sure that you are an administrator, sign in to your MathWorks Account to check which licenses you are associated to as an administrator.
- The MATLAB Web App Server license you are using must be current on Software Maintenance Service. Learn more.

The MATLAB Web App Server reference architecture is configured to use a network license manager by default. The license manager is automatically installed on a virtual machine in the cloud. However, the reference architecture provides an option to use an existing license manager as well. From the following scenarios, select a procedure for activating your license:

Scenario	Procedure
You need to activate a license or trial that has never been activated before.	"Activate a License for the First Time" on page 5-3
Your MATLAB Web App Server license has been previously activated in another location, on-premises, or in the cloud.	"Move a License That Has Been Previously Activated" on page 5-4

Activate a License for the First Time

Follow this procedure to activate a license or trial that has never been activated before.

Deploy MATLAB Web App Server in the Cloud

Deploy MATLAB Web App Server in the cloud using the reference architecture. To use the license manager in the cloud, you must provide a fixed MAC address when you activate the license. You can get a MAC address to the virtual machine hosting the license server only after the deployment is complete.

- MATLAB Production Server Reference Architecture on AWS

Activate License

1   Sign in to your MathWorks Account, and select the MATLAB Web App Server license.
2   Go to **Install** and **Activate**.

3 Click **Activate to Retrieve License File**.

4 In the **Host ID field**, enter the fixed MAC address of the network license manager.

You can get the MAC address of the network license manager from the Network License Manager for MATLAB dashboard that's deployed to the cloud. In the dashboard, click **Administration > Manage License**.

Note When you delete the cloud resources, you also delete the fixed MAC address. For hosting the license manager in the cloud, you must have a fixed address:

- AWS: Use an Elastic Network Interface (ENI), which contains a (non-changeable) static MAC address attribute.

Changing the fixed MAC address in MathWorks License Center is limited to 4 times per year.

---

5 Finish all fields, and click **Continue to finish activation**.

Upload License

You need to upload the license file you get from the activation process to the Network License Manager for MATLAB dashboard. To upload a license file through the dashboard:

1 On the dashboard menu, select **Administration > Manage License**.

2 Click **Browse License File** and select a file.

3 Click **Upload**.

Move a License That Has Been Previously Activated

Follow this procedure to activate a license that has been previously activated in another location.

Shut Down the Current License Server

Platform	Procedure
Windows	**1** In your current `LMtools.exe`, navigate to the **Start/Stop/Reread** tab and choose **Stop Server**. **2** Navigate to the **Config Services** tab and choose **Remove Service**. **3** Close out of the current `LMtools.exe`.
Linux	**1** Run `lmdown`. **2** Delete the existing network license manager by deleting the entire folder where you previously placed the license manager binaries.

Deactivate Existing License

1 Sign in to your MathWorks Account, and select the existing MATLAB Web App Server license.

2 Go to **Install** and **Activate**.

3   Find your activated license, and click ⬚ to deactivate your license.

4   Click **Deactivate** on the next screen to complete the process.

Activate License

Activate the license as described in "Activate a License for the First Time" on page 5-3.

Web App Packaging and Deployment

Create Web App

Note To create a web app, you need an installation of the MATLAB Compiler product.

App Designer Prerequisite

Before you can package and then deploy a web app, you need to create an app using MATLAB App Designer. For more information, see "Create and Run a Simple App Using App Designer" (MATLAB).

Steps to Package and Create a Web App

1   Type `webAppCompiler` at the MATLAB command line to open the **Web App Compiler** app.
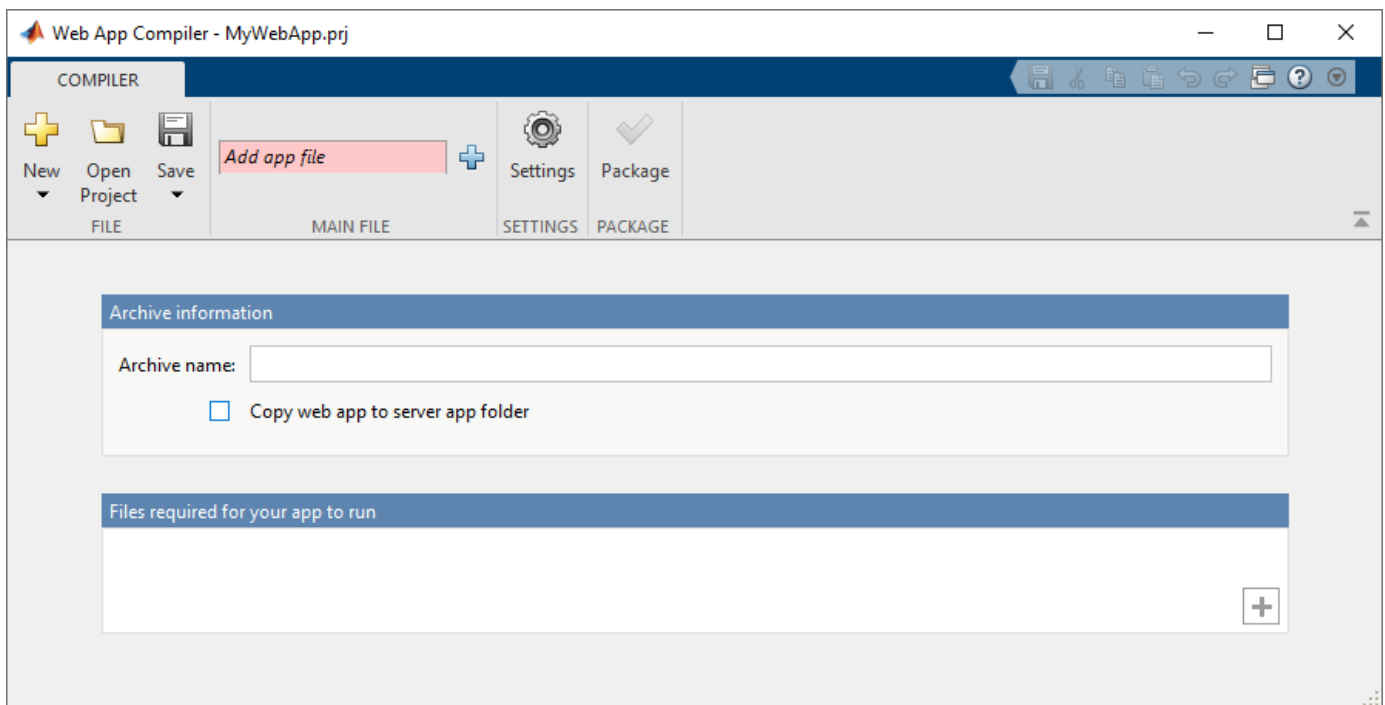
2   In the **MAIN FILE** section of the toolstrip, click the ⊕ button to add your App Designer `.mlapp` file to the project. The Web App Compiler automatically resizes to include an **App details** section that contains information about the app such as app name, author, summary, description, and version. You can edit information about the app in App Designer by clicking **Edit App Details**. Click **Refresh** to update Web App Compiler with any changes you have made.

3   *(Optional)* Select **Copy web app to server app folder** check box and specify the path to the app folder on the server where you want the web app archive (`.ctf` file) to be automatically copied. If you leave this check box cleared, the Web App Compiler will create the web app archive (`.ctf` file) in the project folder. You must manually copy or upload the web app archive (`.ctf` file) to the app folder on the server. .

4   Add supporting files, if any, in the **Files required for your app to run** section. Supporting files include any MAT-files, images used by your web app, or user-written MATLAB functions not found by MATLAB Compiler.

5   Click **Package** to package the app, and create a web app archive (`.ctf` file).

In the **Save Project** dialog box that opens, specify a project name and a location where you want to save the web app project. Web App Compiler saves your project and opens a **Package** dialog box.

6   Once packaging is complete, in the **Package** dialog box, click **Open output folder**. This step opens the project folder which contains the following files:

- *webAppArchiveName*`.ctf`
- `mccExcludedFiles.log`
- `PackagingLog.html`
- `requiredMCRProducts.txt`

You can view the log file, `PackagingLog.html`, to see the exact `mcc` syntax used to package and create the web app archive.

7   To use the web app, deploy the web app archive file, *webAppArchiveName*`.ctf`. For more information, see "Deploy Web App" on page 6-4.

See Also

More About

- "Web App Limitations and Unsupported Functionality" on page 6-7
- "Deploy Web App" on page 6-4
- "Run Web App" on page 6-6
- "Simple Mortgage Calculator Web App" on page 6-9

Deploy Web App

You can deploy web apps to the server in one of two ways:

- Copy the web app archive (`.ctf` file) generated by Web App Compiler (in MATLAB Compiler) to the apps folder configured by the server.
- Upload the web app archive (`.ctf` file) generated by Web App Compiler (in MATLAB Compiler) to the apps folder configured by the server. This deployment option is available only if you have the MATLAB Web App Server product installed. For more information, see "Install or Uninstall MATLAB Web App Server Product" on page 3-2.

Deploy a Web App by Copying

1 Navigate to the project folder generated by Web App Compiler (in MATLAB Compiler) during the packing process.

2 Copy the file *webAppArchiveName*`.ctf` to the app folder configured by the MATLAB Web App Server. If you selected the **Copy web app to server app folder** check box in the Web App Compiler while creating the web app, your web app archive (`.ctf` file) is automatically be copied to the app folder configured by the server.

You can access the app folder by clicking the **Open App Folder** button in the MATLAB Web App Server utility.

Operating System	Apps Folder Location
Windows	`%ProgramData%\MathWorks\webapps\R2022a\apps`
Linux	`/local/MathWorks/webapps/R2022a/apps`
macOS	`/Library/Application Support/MathWorks/webapps/R2022a/apps`

Note You must have write permissions to the app folder to copy a web app archive (`.ctf` file) to folder.

Your web app is now deployed and can be accessed from the web apps home page. For more information, see "Run Web App" on page 6-6.

Deploy a Web App by Uploading (Available Only with MATLAB Web App Server Product)

Prerequisites

- Verify that you have the MATLAB Web App Server product installed. For more information, see "Install or Uninstall MATLAB Web App Server Product" on page 3-2.
- Verify that "Authentication" on page 4-4 and "Role-Based Access" on page 4-10 are enabled on the server.
- Verify that you are designated as an *author* while configuring "Role-Based Access" on page 4-10.

Procedure

1   Navigate to the web apps home page configured by the server. You can obtain the URL to the home page by executing `webapps-status` at the system command-line or by getting the URL from an individual who is administering the server. The format of the home page URL is:

`http://`*`webAppServer`*`:`*`PortNumber`*`/webapps/home/index.html`

Here, *webAppServer* is your web app server hostname, and *PortNumber* is the port specified when configuring the server.

2   Click on the web apps home page to open the *Manage Apps* page.

3   Click and navigate to the project folder generated by Web App Compiler (in MATLAB Compiler) during the packaging process.

4   Select the file *webAppArchiveName*`.ctf` and click **Open** to upload the web app. You get a confirmation that the *webAppArchiveName*`.ctf` was successfully uploaded to the server.

See Also

More About

- "Run Web App" on page 6-6
- "Create Web App" on page 6-2

Run Web App

Using a web browser, you can access web apps running on the server by navigating to the web apps home page.

You can obtain the URL to the home page by executing `webapps-status` at the system command line or by getting the URL from a system administrator. The format of the home page URL is:

`http://webAppServer:PortNumber/webapps/home/index.html`

Here, *webAppServer* is your web app server hostname, and *PortNumber* is the port specified when configuring the server.

1   To run a web app, click the web app tile on the home page.
2   To see a list of all web apps with their status and diagnostic messages, click the **Diagnostics** link or **Manage Apps** link on the top-right corner of the home page. The choice of whether you see the **Diagnostics** link or the **Manage Apps** link depends on what role you have been assigned to while configuring role-based access.

    If you are designated with role of *User*, you see the **Diagnostics** link. If you are designated with the role of *Author*, you see the **Manage Apps** link. For more information, see "Role-Based Access" on page 4-10.
3   To go back to the home page of your web apps, click **MATLAB Web Apps** on the breadcrumb trail at the top of the page.

See Also

More About

Web App Limitations and Unsupported Functionality

When packaging a MATLAB app into a web app consider the following functional limitations. Using certain functions may result in an error or unexpected behavior.

Behavior Differences

Because a deployed web app is hosted on a server, functions that use system information have access only to server-side information, not client-side information. In addition, the web app is displayed in a browser window as opposed to a separate figure window. As a result, certain functions and properties behave differently in a deployed web app than on your local system. This table lists some of the differences that are most relevant to app building workflows.

Category	Limitations
Graphics root object properties	The `MonitorPositions`, `PointerLocation`, `ScreenDepth`, `ScreenPixelsPerInch`, and `ScreenSize` properties of the `groot` object contain server-side information, not client-side information.
Figure properties	The `CloseRequestFcn`, `Visible`, `WindowState`, and `WindowStyle` properties have no effect in a web app. You can specify the initial figure size using the `Position` or `InnerPosition` properties, but programmatically resizing the figure by setting these properties while the app is running might result in unexpected behavior.
File dialog boxes	Although you can upload and download files from a local system in a deployed web app using `uigetfile` and `uiputfile`, there are some limitations: • You cannot download files while an external application is writing them. • You cannot specify the default path or file name.
System commands	Commands that return system information, such as `computer`, `ispc`, `isunix`, `ismac`, and `listfonts`, return server-side information, not client-side information.

Unsupported Functionality

Some functionality is not supported in deployed web apps. This table lists the unsupported functionality that is most relevant to app building workflows.

Category	Not Supported
Multiwindow apps	Multiple calls to `figure` or `uifigure` are not supported. In addition, functions that create dialog boxes that appear as a separate window are not supported. These functions include `dialog`, `msgbox`, `errordlg`, `warndlg`, `helpdlg`, `listdlg`, `questdlg`, `inputdlg`, `uisetcolor`, and `uisetfont`. However, functions that create dialog boxes within a figure window, such as `uialert`, `uiconfirm`, and `uiprogressdlg`, are supported.
File dialog boxes	Opening a folder selection dialog box on the client using `uigetdir` is not supported.
Saving and printing	The `print`, `printpreview`, and `exportapp` functions are not supported.
Plotting functions	The `wordcloud` function is not supported.
Axes toolbar interactions	Data brushing is not supported.
Copy and paste	Copying and pasting text to and from the system clipboard using the `clipboard` function is not supported.
Project features	Project features such as `matlab.project.createProject` and `matlab.project.loadProject` are not supported.
OLE Automation	The `actxserver` function is not supported.

This unsupported functionality is in addition to App Designer graphics limitations. For more information, see "Display Graphics in App Designer" (MATLAB).

See Also

More About

- "Create Web App" on page 6-2
- "Authoring Secure Web Apps" on page 4-26

Simple Mortgage Calculator Web App

This example shows how to create a web app and host it on the MATLAB Web App Server . The example uses the simple calculator app from App Designer as a basis for the web app. For information about the app and the numerical values you can enter, see "App That Calculates and Plots Data Based on Numerical Input" (MATLAB). In the workflow, you:

- Package the simple calculator app from App Designer using the Web App Compiler app or `compiler.build.webAppArchive` in MATLAB Compiler. This step creates a web app archive (`.ctf`) file.
- Deploy the web app archive (`.ctf`) file to the MATLAB Web App Server.
- Run the web app from the web apps home page.

Prerequisites

1   Install the MATLAB Web App Server product and set up the server.

   - For more information on installing the product, see "Install or Uninstall MATLAB Web App Server Product" on page 3-2.
   - For more information on setting up the server, see "Set Up MATLAB Web App Server" on page 3-4.

2   Copy the App Designer file `Mortgage.mlapp` to your current working directory. The default location of the file is:

Operating System	Default File Location
Windows	`C:\Program Files\MATLAB\R2022a\examples\matlab\main\Mortgage.mlapp`
Linux	`/usr/local/MATLAB/R2022a/examples/matlab/main/Mortgage.mlapp`
macOS	`/Applications/MATLAB/R2022a.app/examples/matlab/main/Mortgage.mlapp`

Package and Create Web App Using Web App Compiler

1   Start MATLAB.

2   Type `webAppCompiler` at the MATLAB command line to open the **Web App Compiler** app.

3   In the **MAIN FILE** section of the toolstrip, click the button to add the `Mortgage.mlapp` file to the project. The Web App Compiler automatically resizes to include an **App details** section that contains information about the app such as app name, author, summary, description, and version. You can edit information about the app in App Designer by clicking **Edit App Details**. Click **Refresh** to update Web App Compiler with any changes you have made.

   - *(Optional)* Make sure to use a display name that is easy to distinguish when your web app is deployed to the server.
   - *(Optional)* Provide a version number for tracking purposes. The version number is visible on the web apps home page.

- *(Optional)* Add a description for your web app in the **Summary** field. This description is visible on the web apps home page.

4 In the **Archive information** section, specify the archive name as `myMortgageWebApp`.

5 Click **Package** to package the app, and create a web app archive (`.ctf` file).

In the **Save Project** dialog box that opens, specify a project name and a location where you want to save the web app project. **Web App Compiler** saves your project and opens a **Package** dialog box.

6 Once packaging is complete, in the **Package** dialog box, click **Open output folder**. This opens the project folder, which contains the following files:

- `myMortgageWebApp.ctf`
- `mccExcludedFiles.log`
- `PackagingLog.html`
- `requiredMCRProducts.txt`

You can view the log file, `PackagingLog.html`, to see the exact `mcc` syntax used to package and create the web app archive.

Package and Create Web App Using compiler.build.webAppArchive

Note If you have already created a standalone application using the **Web App Compiler** app, you can skip this section. However, if you want to know how to create a web app from the MATLAB command window using a programmatic approach, follow these instructions.

1 Save the path to the file `Mortgage.mlapp` located in *matlabroot*\examples\matlab\main.

```
mortgageApp = fullfile(matlabroot,'examples','matlab','main','Mortgage.mlapp');
```

2 Build a web app archive using the `compiler.build.webAppArchive` command. Use name-value arguments to specify the archive name.

```
buildResults = compiler.build.webAppArchive(mortgageApp,...
    'ArchiveName','myMortgageWebApp');
```

The `compiler.build.Results` object `buildResults` contains information on the build type, generated files, and build options.

3 The build function generates the following files within a folder named `myMortgageWebAppwebAppArchive` in your current working directory:

- `mccExcludedFiles.log` — Log file that contains a list of any toolbox functions that were not included in the web app. For more information on non-supported functions, see MATLAB Compiler Limitations (MATLAB Compiler).

- `myMortgageWebApp.ctf` — Component technology file that contains the deployable archive.

- `requiredMCRProducts.txt` — Text file that contains product IDs of products required by MATLAB Runtime to run the web app.

- `unresolvedSymbols.txt` — Text file that contains information on unresolved symbols.

Deploy Web App

1 Navigate to the project folder generated by MATLAB Compiler during the packaging process.

2 Copy the web app archive file `myMortgageWebApp.ctf` to the app folder configured by the server. The default location is:

Operating System	Apps Folder Location
Windows	`%ProgramData%\MathWorks\webapps\R2022a\apps`
Linux	`/local/MathWorks/webapps/R2022a/apps`
macOS	`/Library/Application Support/MathWorks/webapps/R2022a/apps`

You can also get the location of the apps folder by executing `webapps-status` at the system command shell.

3 Open a web browser and navigate to the web apps home page using the URL obtained from executing the `webapps-status` command. You see a tile displaying the simple mortgage calculator web app. Your web app is now deployed.
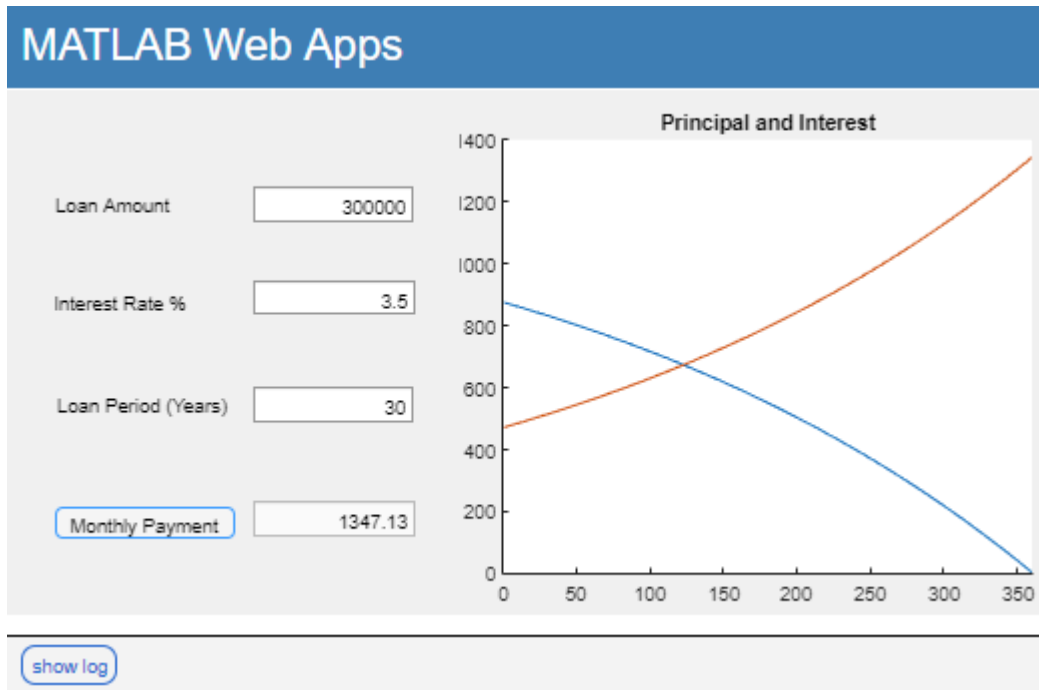
Run Web App

1 To run a web app, click the `myMortgageWebApp` tile on the web apps home page.

The web app opens in a new tab.

2 Click the **Monthly Payment** button to get the monthly payment and the principal and interest graph.

You have successfully created, deployed, and run a web app.

Mortgage Calculator Web App

See Also

More About

- "Create Web App" on page 6-2
- "Deploy Web App" on page 6-4
- "Run Web App" on page 6-6
- "Simulink Simulation Web App" on page 6-13

Simulink Simulation Web App

This example shows how to create a web app containing a Simulink simulation and host it on the MATLAB Web App Server. The example uses the mass spring damper model in Simulink and a MATLAB app that invokes the model as a basis for the web app. The APIs for creating a simulation can be found in the Simulink Compiler product. In the workflow, you:

- Package the MATLAB app containing Simulink simulation using the Web App Compiler app in MATLAB Compiler. This step creates a web app archive (`.ctf`) file.
- Deploy the web app archive (`.ctf`) file to the MATLAB Web App Server.
- Run the web app from the web apps home page.

Prerequisites

Note This example requires the Simulink Compiler product. For details, see "Simulink Compiler Workflow Overview" (Simulink Compiler).

---

1. Install the MATLAB Web App Server product and set up the server.

   - For more information on installing the product, see "Install or Uninstall MATLAB Web App Server Product" on page 3-2.
   - For more information on setting up the server, see "Set Up MATLAB Web App Server" on page 3-4.

2. Copy the Simulink model file `MassSpringDamperModel.slx` and the corresponding MATLAB app `MassSpringDamperApp.mlapp` to your current working directory. The default location for the files is:

Operating System	Default Location for Files
Windows	`C:\Program Files\MATLAB\R2022a\examples\simulinkcompiler\main\`
Linux	`/usr/local/MATLAB/R2022a/examples/simulinkcompiler/main`
macOS	`/Applications/MATLAB/R2022a.app/examples/simulinkcompiler/main`

3. Open the `MassSpringDamperApp.mlapp` in MATLAB App Designer and switch to **Code View**. Verify that the Simulink Compiler APIs to create a simulation are present in the `SimulateButtonPushed` callback.

Package and Create Web App

1. Start MATLAB.

2. Type `webAppCompiler` at the MATLAB command line to open the **Web App Compiler** app.

3. In the **MAIN FILE** section of the toolstrip, click the ⊞ button to add the `MassSpringDamperApp.mlapp` file to the project. The Web App Compiler automatically resizes to include an **App details** section that contains information about the app such as app name, author, summary, description, and version. You can edit information about the app in App

Designer by clicking **Edit App Details**. Click **Refresh** to update Web App Compiler with any changes you have made.

- *(Optional)* Make sure to use a display name that is easy to distinguish when your web app is deployed to the server.
- *(Optional)* Provide a version number for tracking purposes. The version number is visible on the web apps home page.
- *(Optional)* Add a description for your web app in the **Summary** field. This description is visible on the web apps home page.

4   In the **Archive information** section, specify the archive name as mySimulinkSimulationWebApp.

5   Click **Package** to package the app, and create a web app archive (.ctf file).

In the **Save Project** dialog box that opens, specify a project name and a location where you want to save the web app project. **Web App Compiler** saves your project and opens a **Package** dialog box.

6   Once packaging is complete, in the **Package** dialog box, click **Open output folder**. This opens the project folder, which contains the following files:

- mySimulinkSimulationWebApp.ctf
- mccExcludedFiles.log
- PackagingLog.html
- requiredMCRProducts.txt

You can view the log file, PackagingLog.html, to see the exact mcc syntax used to package and create the web app archive.

Deploy Web App

1   Navigate to the project folder generated by Web App Compiler during the packaging process.

2   Copy the web app archive file mySimulinkSimulationWebApp.ctf to the app folder configured by the server. The default location is:

Operating System	Apps Folder Location
Windows	%ProgramData%\MathWorks\webapps\R2022a\apps
Linux	/local/MathWorks/webapps/R2022a/apps
macOS	/Library/Application Support/MathWorks/webapps/ R2022a/apps

You can also get the location of the apps folder by executing webapps-status at the system command shell.

3   Open a web browser and navigate to the web apps home page using the URL obtained from executing the webapps-status command. You see a tile displaying the simple mortgage calculator web app. Your web app is now deployed.

Run Web App

1 To run a web app, click the `mySimulinkSimulationWebApp` tile on the web apps home page.

The web app opens in a new tab.

2 Click the **Simulate** button to run the simulation.

You have successfully created, deployed, and run a web app.

See Also

More About

- "Create Web App" on page 6-2
- "Deploy Web App" on page 6-4
- "Run Web App" on page 6-6
- "Simple Mortgage Calculator Web App" on page 6-9

Troubleshooting

Server Startup Failures

If your server does not start when you execute `webapps-start`, it may be due to one or more of the following reasons.

License Manager

Your server will not start if you do not have a license manager running. For more information, check the logs listed in "Server Logs" on page 7-4.

To fix this issue, verify that you have a license manager running and that you have configured the license manager for use with the server using `webapps-config`.

SSL Configuration

Your server will not start if you do not have SSL configured correctly. For more information, see "Enable SSL" on page 4-2.

Authentication Syntax

Your server will not start if the `webapps_authn.json` file is not syntactically correct. For more information, see "Authentication" on page 4-4.

Previous Installation

If you previously installed the MATLAB Web App Server product but failed to run `webapps-uninstall` from the system command-line before uninstalling the product from your system, your current installation of the server may not start.

Executing `webapps-uninstall` unregisters services and removes user accounts used by the services. Failing to execute `webapps-uninstall` results in orphan services and user accounts that cause your current installation of the server to not start.

To fix this issue, execute `webapps-uninstall` from the `script` folder of your current installation before setting up your new server using `webapps-setup`.

Operating System	Default Location of Command-Line Scripts
Windows *(Administrator)*	`C:\Program Files\MATLAB\MATLAB Web App Server\R2022a\script`
Linux *(sudo)*	`/usr/local/MATLAB/MATLAB_Web_App_Server/R2022a/script`
macOS *(sudo)*	`/Applications/MATLAB/MATLAB_Web_App_Server/R2022a/script`

On Windows systems, you may also need to delete the USR folder located in `C:\ProgramData\MathWorks\webapps\R2022a` before starting the new server.

See Also
`webapps-start` | `webapps-config` | `webapps-uninstall`

More About

- "Server Logs" on page 7-4
- "Enable SSL" on page 4-2
- "Authentication" on page 4-4

Server Logs

The default location for the server logs is:

Operating System	Logs Folder Location
Windows	`%ProgramData%\MathWorks\webapps\R2022a\logs`
Linux	`/local/MathWorks/webapps/R2022a/logs`
macOS	`/Library/Application Support/MathWorks/webapps/R2022a/logs`

The server generates multiple log files, each with a distinct purpose. Depending on the issue you encounter, you may need to review one or more of these log files.

If you configured a different folder location for the server logs, execute `webapps-config get logs_path` from the system command-line to get the location.

Log File Name	Purpose
`webapps_<timestamp>.log`	Log file for the service that runs the server.
`webapps_launcher_<timestamp>.log`	Log file for the service that runs applications.
`webapps_launcher_service_start.err`	Log file that captures errors associated with starting the service that runs applications.
`webapps_launcher_service_start.out`	Log file that captures standard output when the service that runs applications fails to start.
`webapps_service_start.err`	Log file that captures errors when the service that runs the server fails to start.
`webapps_service_start.out`	Log file that captures standard output when the service that runs the server fails to start.

Note If you are unable to see log files in the logs folder, check to see if the logs folder has write permissions.

See Also

More About

- "Server Startup Failures" on page 7-2
- "Diagnostics" on page 7-5
- "Web App Session Log" on page 7-6

Diagnostics

Click **Diagnostics** or **Manage Apps** on the web apps home page to get information about each web app.

In the MATLAB Web App Server product, if you are using role-based access, you see the following buttons:

- If your designated role is *Author*, you see a **Manage Apps** link.
- If your designated role is *User*, you see a **Diagnostics** link.

When you click **Diagnostics** or **Manage Apps**, you see a page with all the web apps deployed to the server along with their status. The various statuses are:

- **OK**—The web app is available to run.
- **Corrupt CTF**—The deployed web app archive file cannot be read.
- **Required Runtime Unavailable**—The web app uses a different version of MATLAB Runtime than the one on the server.
- **Not a Web App**—The deployed `.ctf` file is not a web apps `.ctf` file.
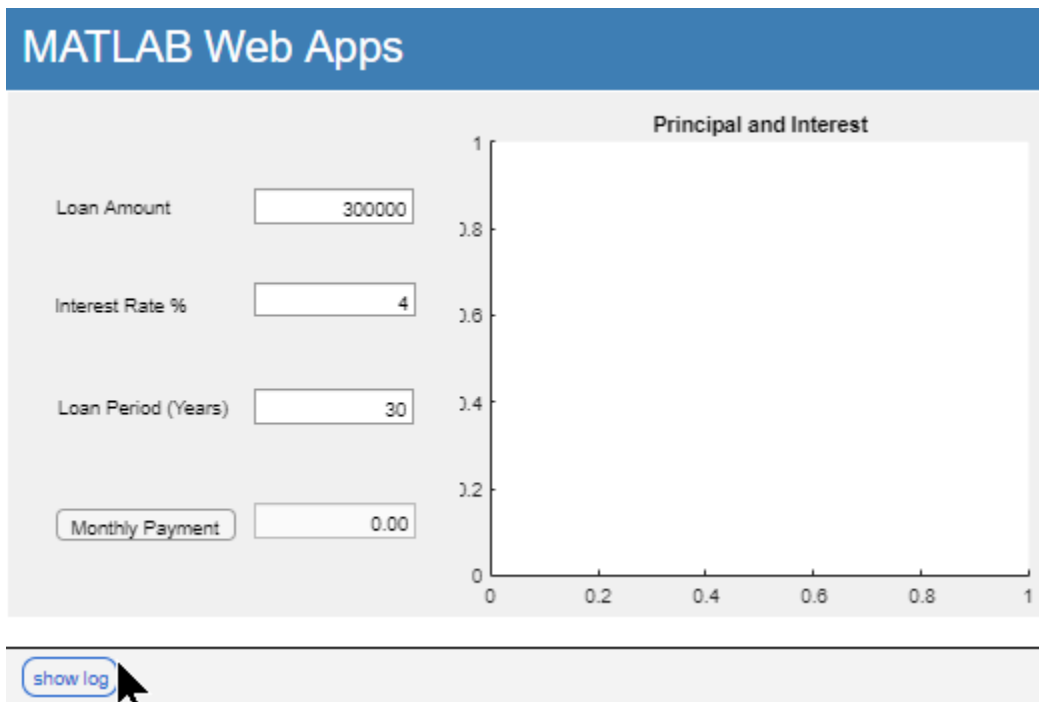
See Also

More About
- "Web App Session Log" on page 7-6
- "Server Logs" on page 7-4
- "Server Startup Failures" on page 7-2

Web App Session Log

The web app session log captures all interaction associated with each web app. If you encounter any issues, you can check the log for more information. You can pass this information to the server administrator for troubleshooting. You can access the log by clicking the **show log** link at the bottom left corner of the browser.

Show Log

See Also

More About

- "Diagnostics" on page 7-5
- "Server Logs" on page 7-4
- "Server Startup Failures" on page 7-2