

MATLAB® Web App Server™

User's Guide



MATLAB®

R2020a



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

MATLAB® Web App Server™ User's Guide

© COPYRIGHT 2020 by The MathWorks, Inc.

The software described in this document is furnished under a license agreement. The software may be used or copied only under the terms of the license agreement. No part of this manual may be photocopied or reproduced in any form without prior written consent from The MathWorks, Inc.

FEDERAL ACQUISITION: This provision applies to all acquisitions of the Program and Documentation by, for, or through the federal government of the United States. By accepting delivery of the Program or Documentation, the government hereby agrees that this software or documentation qualifies as commercial computer software or commercial computer software documentation as such terms are used or defined in FAR 12.212, DFARS Part 227.72, and DFARS 252.227-7014. Accordingly, the terms and conditions of this Agreement and only those rights specified in this Agreement, shall pertain to and govern the use, modification, reproduction, release, performance, display, and disclosure of the Program and Documentation by the federal government (or other entity acquiring for or through the federal government) and shall supersede any conflicting contractual terms or conditions. If this License fails to meet the government's needs or is inconsistent in any respect with federal procurement law, the government agrees to return the Program and Documentation, unused, to The MathWorks, Inc.

Trademarks

MATLAB and Simulink are registered trademarks of The MathWorks, Inc. See www.mathworks.com/trademarks for a list of additional trademarks. Other product or brand names may be trademarks or registered trademarks of their respective holders.

Patents

MathWorks products are protected by one or more U.S. patents. Please see www.mathworks.com/patents for more information.

Revision History

March 2020	Online only	New for Version 1.0 (Release 2020a)
------------	-------------	-------------------------------------

Introduction

1	MATLAB Web App Server Product Description	1-2
----------	--	------------

System Requirements

2	System Requirements for Development Version of MATLAB Web App	
	Server	2-2
	Supported Platforms	2-2
	Hardware Requirements	2-2
	Software Requirements	2-2
	Supported Browsers and Platform Incompatibilities	2-3
	Supported Browsers	2-3
	Platform Incompatibilities	2-3

Installation and Setup

3	Install or Uninstall MATLAB Web App Server Product	3-2
	Install	3-2
	Uninstall	3-2
	Set Up MATLAB Web App Server	3-4
	License Manager	3-4
	Install MATLAB Runtime	3-4
	Set Up the Server	3-4
	Service Information, Groups, and Folder Locations	3-5
	Upgrading to MATLAB Web App Server Product	3-8
	Service Information, Groups, and Folder Locations	3-9
	Service Info	3-9
	Groups	3-9
	Folder Locations	3-10
	Configure Server to Use MATLAB Runtime	3-11
	Install MATLAB Runtime	3-11
	Configure the Server	3-11

4

Enable SSL	4-2
Authentication	4-3
LDAP Authentication	4-3
Example webapps_authn.json File for LDAP	4-5
OIDC Authentication	4-5
Example webapps_authn.json File for OIDC Using Google Identity Platform	4-6
Role-Based Access	4-8
Example webapps_app_roles.json File for LDAP Authentication	4-10
Example webapps_app_roles.json file for Azure AD Authentication	4-10
MATLAB Web App Server Security	4-12
Potential Risks	4-13
Authoring Secure Web Apps	4-14
Authentication and Authorization	4-14
Do Not Call eval()	4-14
Limit Free-Text User Input	4-14
Sanitize User Input	4-14
Sanitize Data Read from Files	4-14
Minimize File System Write Operations	4-14
Verify Trustworthiness of Third-Party Code	4-15
Reduce Exposure Level	4-15
Securely Deploying Web Apps	4-16

Web App Packaging and Deployment

5

Create Web App	5-2
App Designer Prerequisite	5-2
Steps to Package and Create a Web App	5-2
Deploy Web App	5-4
Deploy a Web App by Copying	5-4
Deploy a Web App by Uploading (Available Only with MATLAB Web App Server Product)	5-4
Run Web App	5-6
Limitations and Unsupported Functionality	5-7
Simple Mortgage Calculator Web App	5-8
Prerequisites	5-8
Package and Create Web App	5-8

Deploy Web App	5-9
Run Web App	5-9
Simulink Simulation Web App	5-11
Prerequisites	5-11
Package and Create Web App	5-11
Deploy Web App	5-12
Run Web App	5-12

Troubleshooting

6

Server Startup Failures	6-2
License Manager	6-2
SSL Configuration	6-2
Authentication Syntax	6-2
Previous Installation	6-2
Server Logs	6-4
Diagnostics	6-5
Web App Session Log	6-6

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.

System Requirements

System Requirements for Development Version of MATLAB Web App Server

System requirements for MATLAB Web App Server.

Supported Platforms

Windows®, Linux®, macOS

Hardware Requirements

The server can be installed on all hardware platforms and operating systems that MATLAB supports. The hardware requirements are:

- Minimum 60 GB of disk capacity to accommodate the server software installation and log files.
- Minimum 1GB of RAM per worker.
- Allocation of 1 processor core (or virtual core) per 4 workers; 2 cores minimum.

Software Requirements

- An installation of MATLAB Runtime. The MATLAB Runtime version number must match the version of MATLAB you used to package the web app archive (.ctf file).

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

- MATLAB Compiler is required to package MATLAB apps as web app archives (.ctf files) to run on the server.
- MATLAB App Designer.

See Also

More About

- “Create and Run a Simple App Using App Designer” (MATLAB)

Supported Browsers and Platform Incompatibilities

Supported Browsers

Web apps are compatible with commonly used browsers like 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 4-12.

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” (Installation and Licensing).

The installation process creates a default installation folder based on your operating system and the release version. For instance, the default installation folder for R2020a is as follows:

Operating System	Default Installation Folder
Windows	C:\Program Files\MATLAB\MATLAB Web App Server\R2020a
Linux	/usr/local/MATLAB/ MATLAB_Web_App_Server/R2020a
macOS	/Applications/MATLAB/ MATLAB_Web_App_Server/R2020a

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” (Installation and Licensing).

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

Operating System	Folders
Windows	Apps Folder: %ProgramData%\MathWorks\webapps\R2020a\apps Logs Folder: %ProgramData%\MathWorks\webapps\R2020a\logs Config Folder: %ProgramData%\MathWorks\webapps\R2020a\config
Linux	Apps Folder: /local/MathWorks/webapps/R2020a/apps Logs Folder: /local/MathWorks/webapps/R2020a/logs Config Folder: /local/MathWorks/webapps/R2020a/config
macOS	Apps Folder: /Library/Application Support/MathWorks/webapps/R2020a/apps Logs Folder: /Library/Application Support/MathWorks/webapps/R2020a/logs Config Folder: /Library/Application Support/MathWorks/webapps/R2020a/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 “Installation Procedures for Network License Manager” (Installation and Licensing).

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: <code>compiler.runtime.download</code>

- 2 Install the MATLAB Runtime using the installer. For installation instructions, see “Install and Configure the 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 (<i>Administrator</i>)	C:\Program Files\MATLAB\MATLAB Web App Server\R2020a\script
Linux (<i>sudo</i>)	/usr/local/MATLAB/MATLAB_Web_App_Server/R2020a/script
macOS (<i>sudo</i>)	/Applications/MATLAB/MATLAB_Web_App_Server/R2020a/script

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

Operating System	Command
Windows (<i>Administrator</i>)	webapps - setup
Linux (<i>sudo</i>)	
macOS (<i>sudo</i>)	

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.

The default account for running the server service processes HTTP and HTTPS traffic and manages 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.

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.

- 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: MwWebAppServerR2020a Service Name: mw-webapps -R2020a	Account Name: MwWebAppWorkerR2020a Service Name: mw-webapps-launcher-R2020a
Linux	Account Name: MwWebAppsServerR2020a Service Name: mw-webapps -R2020a Service File: /etc/systemd/system/mw-webapps-R2020a.service	Account Name: MwWebAppsWorkerR2020a Service Name: mw-webapps-launcher-R2020a Service File: /etc/systemd/system/mw-webapps-launcher-R2020a.service
macOS	Account Name: MwWebAppsServerR2020a Service Name: com.mathworks.mw-webapps-R2020a Service File: /Library/LaunchDaemons/com.mathworks.mw-webapps-R2020a.plist	Account Name: MwWebAppsWorkerR2020a Service Name: com.mathworks.mw-webapps-launcher-R2020a Service File: /Library/LaunchDaemons/com.mathworks.mw-webapps-launcher-R2020a.plist

Groups

Operating System	Group Name	Description
Windows	MwWebAppAuthorsR2020a	Members of this group can upload applications to the server.
Linux	MwWebAppsAuthorsR2020a	Members of this group can upload applications to the server.
macOS	MwWebAppsAuthorsR2020a	Members of this group can upload applications to the server.

Folder Locations

Operating System	Folders
Windows	<p>Apps Folder: %ProgramData%\MathWorks\webapps\R2020a\apps</p> <p>Logs Folder: %ProgramData%\MathWorks\webapps\R2020a\logs</p> <p>Config Folder: %ProgramData%\MathWorks\webapps\R2020a\config</p>
Linux	<p>Apps Folder: /local/MathWorks/webapps/R2020a/apps</p> <p>Logs Folder: /local/MathWorks/webapps/R2020a/logs</p> <p>Config Folder: /local/MathWorks/webapps/R2020a/config</p>
macOS	<p>Apps Folder: /Library/Application Support/MathWorks/webapps/R2020a/apps</p> <p>Logs Folder: /Library/Application Support/MathWorks/webapps/R2020a/logs</p> <p>Config Folder: /Library/Application Support/MathWorks/webapps/R2020a/config</p>

See Also

More About

- “Install or Uninstall MATLAB Web App Server Product” on page 3-2

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 3-9.

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: MwWebAppServerR2020a Service Name: mw-webapps - R2020a	Account Name: MwWebAppWorkerR2020a Service Name: mw-webapps-launcher-R2020a
Linux	Account Name: MwWebAppsServerR2020a Service Name: mw-webapps - R2020a Service File: /etc/systemd/system/mw-webapps-R2020a.service	Account Name: MwWebAppsWorkerR2020a Service Name: mw-webapps-launcher-R2020a Service File: /etc/systemd/system/mw-webapps-launcher-R2020a.service
macOS	Account Name: MwWebAppsServerR2020a Service Name: com.mathworks.mw-webapps-R2020a Service File: /Library/LaunchDaemons/com.mathworks.mw-webapps-R2020a.plist	Account Name: MwWebAppsWorkerR2020a Service Name: com.mathworks.mw-webapps-launcher-R2020a Service File: /Library/LaunchDaemons/com.mathworks.mw-webapps-launcher-R2020a.plist

Groups

Operating System	Group Name	Description
Windows	MwWebAppAuthorsR2020a	Members of this group can upload applications to the server.
Linux	MwWebAppsAuthorsR2020a	Members of this group can upload applications to the server.
macOS	MwWebAppsAuthorsR2020a	Members of this group can upload applications to the server.

Folder Locations

Operating System	Folders
Windows	Apps Folder: %ProgramData%\MathWorks\webapps\R2020a\apps Logs Folder: %ProgramData%\MathWorks\webapps\R2020a\logs Config Folder: %ProgramData%\MathWorks\webapps\R2020a\config
Linux	Apps Folder: /local/MathWorks/webapps/R2020a/apps Logs Folder: /local/MathWorks/webapps/R2020a/logs Config Folder: /local/MathWorks/webapps/R2020a/config
macOS	Apps Folder: /Library/Application Support/MathWorks/webapps/R2020a/apps Logs Folder: /Library/Application Support/MathWorks/webapps/R2020a/logs Config Folder: /Library/Application Support/MathWorks/webapps/R2020a/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: <code>compiler.runtime.download</code>

- 2 Install MATLAB Runtime using the installer. For installation instructions, see “Install and Configure the 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 (<i>Administrator</i>)	C:\Program Files\MATLAB\MATLAB Web App Server\R2020a\script
Linux (<i>sudo</i>)	/usr/local/MATLAB/ MATLAB_Web_App_Server/R2020a/ script
macOS (<i>sudo</i>)	/Applications/MATLAB/ MATLAB_Web_App_Server/R2020a/ 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

Security

Enable SSL

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 (<i>Administrator</i>)	C:\Program Files\MATLAB\MATLAB Web App Server\R2020a\script
Linux (<i>sudo</i>)	/usr/local/MATLAB/MATLAB_Web_App_Server/R2020a/script
macOS (<i>sudo</i>)	/Applications/MATLAB/MATLAB_Web_App_Server/R2020a/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 4-3.

See Also

More About

- “Authentication” on page 4-3
- “Role-Based Access” on page 4-8

Authentication

Note To use authentication, you need to enable SSL on the server. For more information, see “Enable SSL” on page 4-2.

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\R2020a\config\webapps_private
Linux	/local/MathWorks/webapps/R2020a/config/webapps_private
macOS	/Library/Application Support/MathWorks/webapps/R2020a/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": "1.0.0",
  "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>"
  }
}

```

- **version:** Specify the version of the JSON schema. Default value for R2020a is: 1.0.0.
- **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.

Example webapps_authn.json File for LDAP

```
{
  "version": "1.0.0",
  "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"
  }
}
```

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": "1.0.0",
  "type": "oidc",
  "authnConfig": {
    "issuer": "<OIDC IdP issuer URI>",
    "clientId": "<Client ID from IdP>",
    "clientSecret": "<Client secret from IdP>",
    "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>"
  }
}
```

- **version:** Specify the version of the JSON schema. The default value for R2020a is: 1.0.0.
- **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-xxxxxxxxxxxxx.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.

- **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:** Specify the port number used by MATLAB Web App Server for OIDC authentication. For example: `3000`.
- **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.

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 URI for MATLAB Web App Server. The format of the URI is: `https://<MATLABWebAppServer_hostname>:<port>/oidc/callback`. For example: `https://myboston.com:3000/oidc/callback`.
-

Example `webapps_authn.json` File for OIDC Using Google Identity Platform

```
{
  "version": "1.0.0",
  "type": "oidc",
  "authnConfig": {
    "issuer": "https://accounts.google.com/.well-known/openid-configuration",
    "clientId": "1234567890-xxxxxxxxxxxxx.apps.googleusercontent.com",
    "clientSecret": "_xxxxxxxxxxxxx_Xxxxxx_xX",
    "scope": ["openid profile email"]
  },
  "appConfig": {
    "port": "3000",
    "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 4-8

External Websites

- <https://developers.google.com/identity/protocols/OpenIDConnect>
- <https://www.oauth.com/oauth2-servers/openid-connect>

Role-Based Access

Note To use role-based access, you need to:

- 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 4-3.

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.

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 4-3.
- 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\R2020a\config\webapps_private
Linux	/local/MathWorks/webapps/R2020a/config/webapps_private
macOS	/Library/Application Support/MathWorks/webapps/R2020a/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 R2020a 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": { "upn": [
        "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 being denied access to the server when you try to login.

See Also

More About

- “Authentication” on page 4-3

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.

The development version of the server relies on the authentication and authorization scheme of its host machine and network. Other than supporting HTTPS, it does not contain any additional mechanisms for authenticating or authorizing web application users. For more information, see .

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 4-14.

For a list of additional risks, see “Potential Risks” on page 4-13.

See Also

More About

- “Potential Risks” on page 4-13
- “Enable SSL” on page 4-2
- “Authoring Secure Web Apps” on page 4-14
- “Securely Deploying Web Apps” on page 4-16

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 4-14
- “Securely Deploying Web Apps” on page 4-16
- “MATLAB Web App Server Security” on page 4-12

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 4-14.

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 4-13
- “Enable SSL” on page 4-2
- “Securely Deploying Web Apps” on page 4-16
- “MATLAB Web App Server Security” on page 4-12

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 4-14
- “Potential Risks” on page 4-13
- “Enable SSL” on page 4-2
- “MATLAB Web App Server Security” on page 4-12

Web App Packaging and Deployment

Create Web App

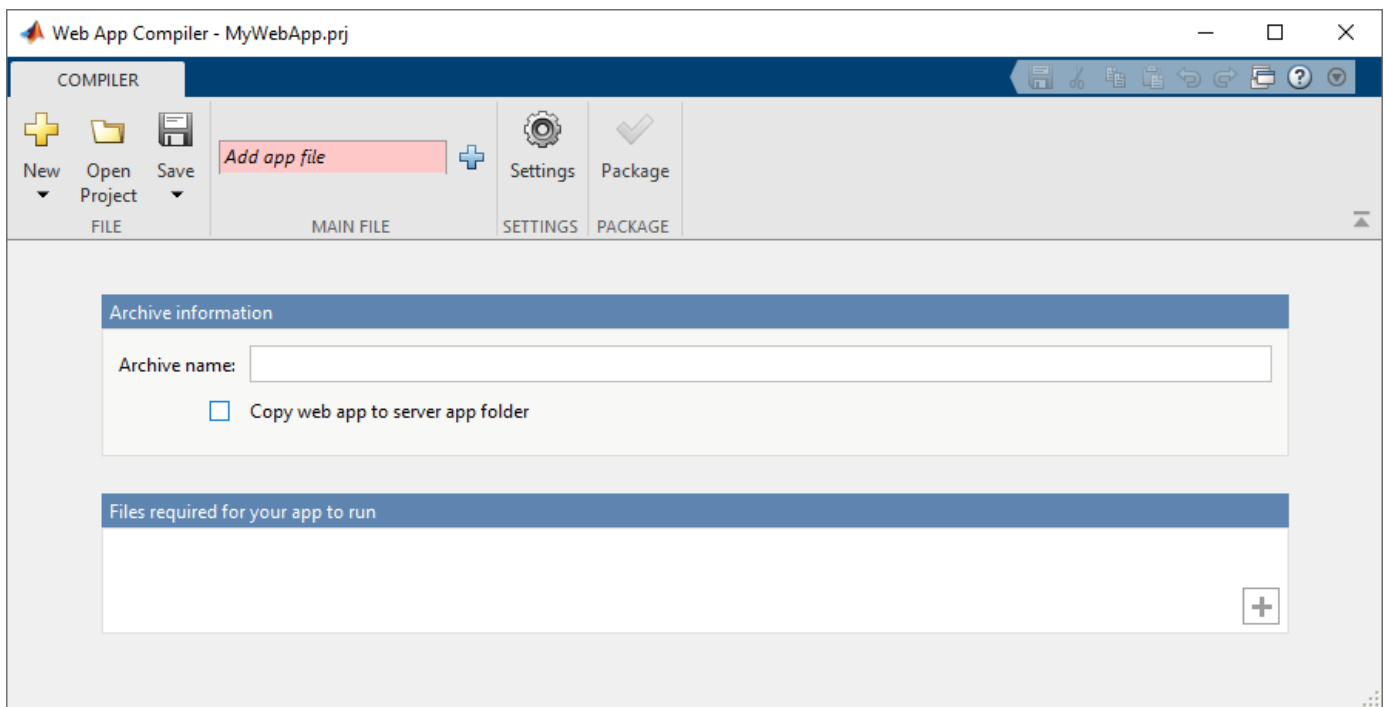
Note To create a web app, you need an installation of the MATLAB Compiler product.


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 5-4.

See Also

More About

- “Limitations and Unsupported Functionality” on page 5-7
- “Deploy Web App” on page 5-4
- “Run Web App” on page 5-6
- “Simple Mortgage Calculator Web App” on page 5-8

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.

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 5-6.

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 4-3 and “Role-Based Access” on page 4-8 are enabled on the server.
- Verify that you are designated as an *author* while configuring “Role-Based Access” on page 4-8.

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 5-6
- “Create Web App” on page 5-2

Run Web App

Note To run a web app, you must also have MATLAB Web App Server running. For more information, see “Install or Uninstall MATLAB Web App Server Product” on page 3-2 and “Set Up MATLAB Web App Server” on page 3-4.

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 4-8.

- 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

- “Create Web App” on page 5-2
- “Deploy Web App” on page 5-4

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.

- Multiwindow apps
 - Multiple calls to `figure` or `uifigure` are not supported.
 - Dialog boxes: `dialog`, `msgbox`, `errorDlg`, `warnDlg`, `helpDlg`, `listDlg`, `questDlg`, `inputDlg`, `uicolor`, and `uifont` are not supported. However, `uiAlert`, `uiConfirm`, and `uiProgressDlg` are supported.
- Although you can upload and download files from a local system in a deployed web app using `uigetfile` and `uiputfile`, you cannot download files while an external application is writing them. Opening a folder selection dialog box on the client using `uigetdir` is not supported.
- Setting the `WindowState` property on a figure has no effect on a web app.
- Printing: `print`, `printpreview` are not supported.
- Graphics root object properties: `MonitorPositions`, `PointerLocation`, `ScreenDepth`, `ScreenPixelsPerInch`, `ScreenSize` are not supported.
- Figure properties: `CloseRequestFcn`, `Position`, `InnerPosition`, `Resize`, `Visible` are not supported.
- System Commands: The system commands `computer`, `ispc`, `isunix`, `ismac`, and `listfonts` will execute but they will not return client-side state information. Only server-side state information is returned.
- Tooltips: Data tips for graphics are not supported.
- Axes toolbar interactions: Data brushing is not supported.

These limitations are 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 5-2
- “Authoring Secure Web Apps” on page 4-14

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 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\R2020a\examples\matlab\main\Mortgage.mlapp
Linux	/usr/local/MATLAB/R2020a/examples/matlab/main/Mortgage.mlapp
macOS	/Applications/MATLAB/R2020a.app/examples/matlab/main/Mortgage.mlapp

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 `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.

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 `myMortgageWebApp.ctf` to the app folder configured by the server. The default location is:

Operating System	Apps Folder Location
Windows	<code>%ProgramData%\MathWorks\webapps\R2020a\apps</code>
Linux	<code>/local/MathWorks/webapps/R2020a/apps</code>
macOS	<code>/Library/Application Support/MathWorks/webapps/R2020a/apps</code>

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.

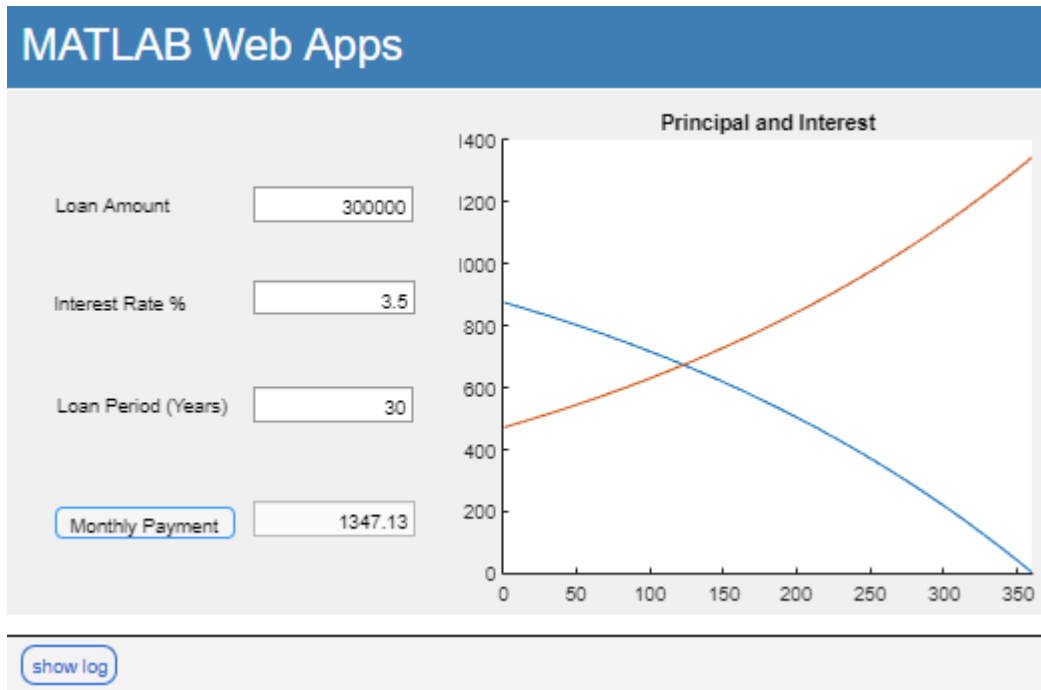
Run Web App

- 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.



See Also

More About

- “Create Web App” on page 5-2
- “Deploy Web App” on page 5-4
- “Run Web App” on page 5-6
- “Simulink Simulation Web App” on page 5-11

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\R2020a\examples\simulinkcompiler\main\
Linux	/usr/local/MATLAB/R2020a/examples/simulinkcompiler/main
macOS	/Applications/MATLAB/R2020a.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:

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 5-2
- “Deploy Web App” on page 5-4
- “Run Web App” on page 5-6
- “Simple Mortgage Calculator Web App” on page 5-8

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 6-4.

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 4-3.

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 (<i>Administrator</i>)	C:\Program Files\MATLAB\MATLAB Web App Server\R2020a\script
Linux (<i>sudo</i>)	/usr/local/MATLAB/ MATLAB_Web_App_Server/R2020a/script
macOS (<i>sudo</i>)	/Applications/MATLAB/ MATLAB_Web_App_Server/R2020a/script

On Windows systems, you may also need to delete the USR folder located in C:\ProgramData\MathWorks\webapps\R2020a before starting the new server.

See Also

`webapps - config` | `webapps - start` | `webapps - uninstall`

More About

- “Server Logs” on page 6-4
- “Enable SSL” on page 4-2
- “Authentication” on page 4-3

Server Logs

The default location for the server logs is:

Operating System	Logs Folder Location
Windows	%ProgramData%\MathWorks\webapps\R2020a\logs
Linux	/local/MathWorks/webapps/R2020a/logs
macOS	/Library/Application Support/MathWorks/webapps/R2020a/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
<code>webapps_<timestamp>.log</code>	Log file for the service that runs the server.
<code>webapps_launcher_<timestamp>.log</code>	Log file for the service that runs applications.
<code>webapps_launcher_service_start.err</code>	Log file that captures errors associated with starting the service that runs applications.
<code>webapps_launcher_service_start.out</code>	Log file that captures standard output when the service that runs applications fails to start.
<code>webapps_service_start.err</code>	Log file that captures errors when the service that runs the server fails to start.
<code>webapps_service_start.out</code>	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 6-2
- “Diagnostics” on page 6-5
- “Web App Session Log” on page 6-6

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.

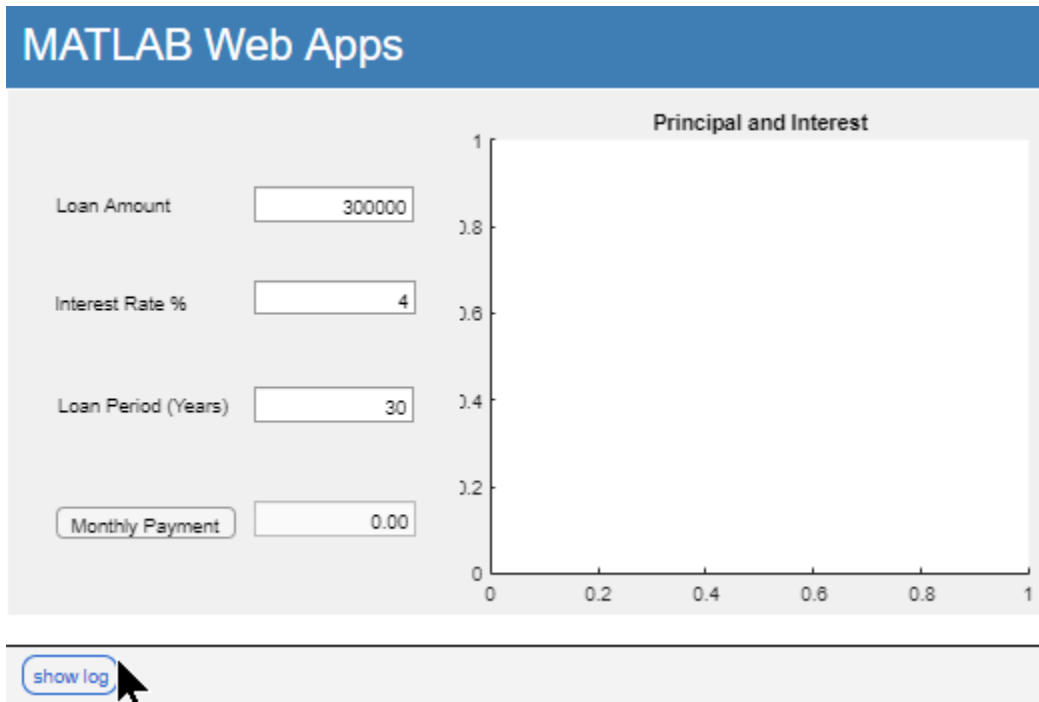
See Also

More About

- “Web App Session Log” on page 6-6
- “Server Logs” on page 6-4
- “Server Startup Failures” on page 6-2

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.



See Also

More About

- "Diagnostics" on page 6-5
- "Server Logs" on page 6-4
- "Server Startup Failures" on page 6-2