MATLAB® Production Server™ Dashboard

Dashboard User's Guide

MATLAB®

R2020**b**

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

MATLAB® Production Server™ Dashboard User's Guide

Revision History

March 2017	Online only	New for Version 3.0 (Release R2017a)
September 2017	Online only	Revised for Version 3.0.1 (Release R2017b)
March 2018	Online only	Revised for Version 3.1 (Release R2018a)
September 2018	Online only	Revised for Version 4.0 (Release R2018b)
March 2019	Online only	Revised for Version 4.1 (Release R2019a)
September 2019	Online only	Revised for Version 4.2 (Release R2019b)
March 2020	Online only	Revised for Version 4.3 (Release R2020a)
September 2020	Online only	Revised for Version 4.4 (Release R2020b)

Contents

1 Manage Server Instances

Manage Server Instances

Create Server Instance

1. Navigate to the server machine for the new instance.

   For example: **Servers** > **localhost**
2. Select **Create New**.
3. In the **Name** field, enter a name for the instance.

   - The name can be any combination of characters and numbers without spaces.
   - The name indicates the purpose of the server. For example, a server instance hosting trading functions for use by east coast traders could be named `trading_east`.
   - It must be unique to the server machine.
4. In the **Description** field, provide an optional description for the server instance.

   The description describes the function of the server instance. For example, the description for `trading_east` may be:

   ```
   Instance hosting arbitrage functions joe, fred, and arlo. This instance
   uses strong encryption, and access to the functions is limited to a
   select group of clients.
   ```
5. Click **Create**.

The new server instance is added to the dashboard in a stopped state. You must manually start it before it can process requests.

Note To start your instance, you will need to complete two additional steps:

1. Set the MATLAB Runtime location. For more information, see "Set MATLAB Runtime Location" on page 1-4.
2. Specify license information. For more information, see "Specify Server Instance License Options" on page 1-18.

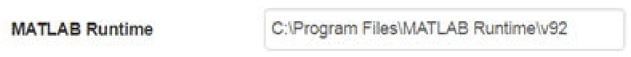
Failure to complete these steps will generate errors.

Set MATLAB Runtime Location

1. Select the server instance from the leftmost navigation pane.

   For example: **Servers** > **localhost** > **myinstance**
2. Select the **Settings** tab.
3. Expand the **Core** area.
4. Replace the value `mCRrOOTuNsET` in the field **MATLAB Runtime** with the location of your MATLAB Runtime.

   For example:



MATLAB Runtime C:\Program Files\MATLAB Runtime\v92

5. Click **Save**.
6. Start/Restart the server instance.

You will get the following error message if you do not set the location of the MATLAB Runtime:

```
Instance cannot be started: please update the MATLAB Runtime option in instance
settings by replacing the "mCRrOOTuNsET" with the full path to MATLAB runtime
directory
```

See Also

Related Procedures

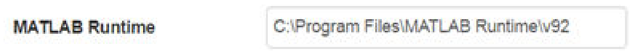
- “Set MATLAB Runtime Location” on page 1-4
- “Specify Server Instance License Options” on page 1-18
- “Start Server Instance” on page 1-5

Set MATLAB Runtime Location

1. Select the server instance from the leftmost navigation pane.

   For example: **Servers** > **localhost** > **myinstance**
2. Select the **Settings** tab.
3. Expand the **Core** area.
4. Replace the value `mCRrOOTuNsET` in the field **MATLAB Runtime** with the location of your MATLAB Runtime.

   For example:

   MATLAB Runtime C:\Program Files\MATLAB Runtime\v92
5. Click **Save**.
6. Start/Restart the server instance.

You will get the following error message if you do not set the location of the MATLAB Runtime:

```
Instance cannot be started: please update the MATLAB Runtime option in instance
settings by replacing the "mCRrOOTuNsET" with the full path to MATLAB runtime
directory
```

See Also

Related Procedures

- "Specify Server Instance License Options" on page 1-18
- "Start Server Instance" on page 1-5

Start Server Instance

In this section...
“Start Server Instance from a Server Information Page” on page 1-5
“Start Server Instance from an Instance Page” on page 1-5

Start Server Instance from a Server Information Page

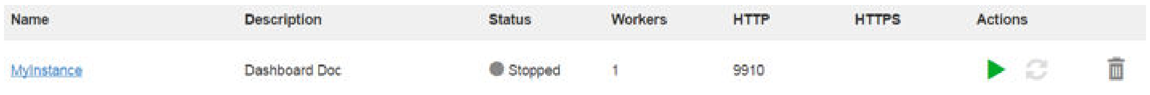
1 From the navigation tree, select the server machine.

For example: **Servers** > **localhost**

2 Locate the server instance in the instance list.

3 Click the green arrow start button in the **Actions** column.

For example:



Name	Description	Status	Workers	HTTP	HTTPS	Actions
MyInstance	Dashboard Doc	● Stopped	1	9910		

Start Server Instance from an Instance Page

1 From the navigation tree, select the server instance.

2 Click the green arrow start button at the top.

Note To start your instance, you will need to complete two additional steps:

1 Set the MATLAB Runtime location. For more information, see “Set MATLAB Runtime Location” on page 1-4.

2 Specify license information. For more information, see “Specify Server Instance License Options” on page 1-18.

Failure to complete these steps will generate errors.

See Also

Related Examples

- “Set MATLAB Runtime Location” on page 1-4
- “Specify Server Instance License Options” on page 1-18
- “Create Server Instance” on page 1-2
- “Stop Server Instance” on page 1-6

Stop Server Instance

In this section...
"Stop Server Instance from a Server Information Page" on page 1-6
"Stop Server Instance from an Instance Page" on page 1-6

Stop Server Instance from a Server Information Page

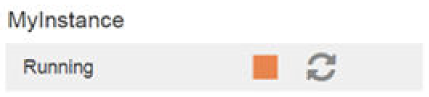
1. From the navigation tree, select the server machine.

   For example: **Servers** > **localhost**
2. Locate the server instance in the instance list.
3. Click the orange square stop button in the **Actions** column.

Stop Server Instance from an Instance Page

1. From the navigation tree, select the server instance.
2. Click the orange square stop button on the top.

   For example:



See Also

Related Procedures

- "Start Server Instance" on page 1-5
- "Restart Server Instance" on page 1-7
- "Remove Server Instance" on page 1-8

Restart Server Instance

In this section...
"Restart Server Instance from a Server Information Page" on page 1-7
"Restart Server Instance from an Instance Page" on page 1-7

Restart Server Instance from a Server Information Page

1 From the navigation tree, select the server machine.

For example: **Servers** > **localhost**

2 Locate the server instance in the instance list.

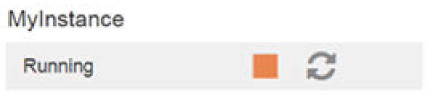
3 Click the dark gray circle restart button in the **Actions** column.

Restart Server Instance from an Instance Page

1 From the navigation tree, select the server instance.

2 Click the dark gray circle restart button on the top.

For example:



See Also

Related Procedures

- "Stop Server Instance" on page 1-6
- "Remove Server Instance" on page 1-8

Remove Server Instance

1. Ensure that the server instance is stopped.
2. From the navigation tree, select the server machine.

   For example: **Servers** > **localhost**
3. Locate the server instance in the instance list.
4. Click the trash can button in the **Actions** column.

See Also

Related Procedures

- "Stop Server Instance" on page 1-6

Edit Server Instance Configuration

1. Select the server instance from the navigation pane.
2. Select the **Settings** tab.
3. Edit the values for configuration settings.
4. Click **Save**.
5. Restart the server instance.

Configuration changes do not take effect until the server instance is restarted.

See Also

Related Procedures

- "Restart Server Instance" on page 1-7

Enable Security on Server Instance

To enable a server instance to use HTTPS:

1. Select the server instance from the leftmost navigation pane.
2. Select the **Settings** tab.
3. Open the **Http** area.
4. In the **Https** field, enter the port number the server instance will use for receiving requests over HTTPS.
5. Click **Save**.
6. Restart the server instance.

See Also

`https`

Related Procedures

- “Restart Server Instance” on page 1-7

Configure Client Authentication on Server Instance

To ensure that only trusted clients have access to a server instance, configure the server instance to require client authentication:

1. Select the server instance from the leftmost navigation pane.
2. Select the **Settings** tab.
3. Expand the **SSL** area.
4. Set **SSL Verify Peer Mode** to `verify-peer-require-peer-cert`.
5. Configure the server instance to use the system-provided CA store, a server-specific CA store, or both.

   Use these configuration properties to control the CA stores used by the server instance:

   - **X509 CA File Store** specifies a PEM formatted CA store to authenticate clients.
   - **X509 Use System Store** directs the server instance to use the system's CA store to authenticate clients.

   Note **X509 Use System Store** does not work on Windows.
6. Optionally select the **X509 Use CRL** property to configure the server instance to respect any certificate revocation lists (CRLs) in the CA store.

   If this property is not specified, the server instance ignores the CRLs and potentially authenticates clients using revoked credentials.
7. Click **Save**.
8. Restart the server instance.

Caution You must add a CRL list to the server's CA store before selecting the **X509 Use CRL** property. If the CA store does not include a CRL list, the server crashes.

See Also

`ssl-verify-peer-mode` | `x509-ca-file-store` | `x509-use-crl` | `x509-use-system-store`

Related Procedures

- “Restart Server Instance” on page 1-7

Adjust Security Protocols and Ciphers Used by Server Instance

By default, MATLAB Production Server instances try to use TLSv1.2 to secure connections between client and server. To change the list of protocols and ciphers available to the server instance:

1. Select the server instance from the leftmost navigation pane.
2. Select the **Settings** tab.
3. Expand the **SSL** area.
4. Set **SSL Protocols** to a comma-separated list of the protocols available to the server instance.

   To enable only TLSv1, set the property to `TLSv1`.

   Because TLSv1.1 and TLSv1.2 are not included in the list, the server instance does not enable the protocols.
5. Set **SSL Ciphers** to a comma-separated list of the cipher suites available to the server instance.

   To enable only high strength cipher suites, set the property to `HIGH`.
6. Click **Save**.
7. Restart the server instance.

See Also
`ssl-ciphers` | `ssl-protocols`

Related Procedures
- "Restart Server Instance" on page 1-7

Specify Client Access to Applications Deployed on Server Instance

By default, MATLAB Production Server instances allow all clients to access all hosted MATLAB programs. To specify a list of allowed clients:

1. Select the server instance from the leftmost navigation pane.
2. Select the **Settings** tab.
3. Expand the **SSL** area.
4. Set **SSL Allowed Client** to a comma-separated list of clients that can access the server instance.

   Clients are identified by the common name of their certificate.
5. Click **Save**.
6. Restart the server instance.

See Also

`ssl-allowed-clients`

Related Procedures

- "Restart Server Instance" on page 1-7

Control Worker Restarts

In this section...

Restart Workers Based on Up Time

As worker processes evaluate MATLAB functions, the MATLAB workspace accumulates saved state and other data. This accumulated data can occasionally cause a worker process to fail. One way to avoid random worker failures is to configure the server instances to restart worker processes when they have been running for set period. To do this:

1. Select the server instance from the leftmost navigation pane.
2. Select the **Settings** tab.
3. Expand the **Worker** area.
4. Set **Worker Restart Interval** to the restart interval.

   For example, to restart workers at intervals of 1 hour, 15 minutes, 5 seconds set the property to `1:15:05`.
5. Click **Save**.
6. Restart the server instance.

Restart Workers Based on Amount of Memory in Use

As worker processes evaluate MATLAB functions, the MATLAB workspace accumulates saved state and other data. This accumulated data can occasionally cause a worker process to fail. One way to avoid random worker failures is to configure the server instances to restart worker processes when they begin consuming a predefined amount of memory.

To do this, adjust three configuration properties:

- **Worker Memory Check Interval** — Interval at which workers are polled for memory usage
- **Worker Restart Memory Limit** — Size threshold at which to consider restarting a worker
- **Worker Restart Memory Limit Interval** — Interval for which a worker can exceed its memory limit before restart

To adjust memory-based restart thresholds:

1. Select the server instance from the leftmost navigation pane.
2. Select the **Settings** tab.
3. Expand the **Worker** area.
4. Set **Worker Memory Check Interval** to a restart interval.

   For example, to restart workers at intervals of 1 hour, 15minutes, 5 seconds, set the property to `1:15:05`.
5. Set **Worker Restart Memory Limit** to the memory limit at which workers are monitored for possible restart.

   For example, to consider restarting workers when they consume 1 GB of memory, set the property to `1GB`.

6. Set **Worker Restart Memory Limit Interval** to the interval during which a worker can exceed the memory limit.

   For example, to restart workers when they exceed the memory limit for 1 hour, set the property to `1:00:00`.
7. Click **Save**.
8. Restart the server instance.

See Also

`worker-memory-check-interval` | `worker-restart-interval` | `worker-restart-memory-limit` | `worker-restart-memory-limit-interval`

Related Procedures

- "Restart Server Instance" on page 1-7

Improve Start Time When Security Is Enabled

When a server instance is configured to use HTTPS, it generates an ephemeral DH key at startup. Generating the DH key at startup provides more security than reading it from a file on disk. However, this can add a couple of minutes to server instance startup time.

If you need the server instance to start without delay and are not concerned about the loss of security, you can configure the server instance to read the ephemeral DH key from a file:

1. Select the server instance from the leftmost navigation pane.
2. Select the **Settings** tab.
3. Expand the **SSL** area.
4. Set **SSL DH Key Parameter File** to the path of the file containing the DH key.
5. Click **Save**.
6. Restart the server instance.

See Also

`ssl-tmp-dh-param`

Related Procedures

- "Restart Server Instance" on page 1-7

Manage Log Files

Log data is written to the server instance log file for as long as a specific server instance is active, or until midnight. When the server is restarted, log data is written to an archive log.

You can set parameters that define when the log is archived:

1. Select the server instance from the leftmost navigation pane.
2. Select the **Settings** tab.
3. Expand the **Logging** area.
4. Set **Log Severity** to the level of detail stored in the log.

   The log level provides levels of information for troubleshooting:

   - `error` — Notification of problems or unexpected results.
   - `warning` — Events that could lead to problems if unaddressed.
   - `information` — High-level information about major server events.
   - `trace` — Detailed information about the internal state of the server.

   Before you call MathWorks® technical support, you should set logging levels to `trace`.
5. Set **Log Archive Max Size** to the maximum size of archived log information.

   When the combined size of the archive reaches this limit, archived logs are purged until the combined size of the archive is less than the specified size. Oldest archived logs are deleted first.
6. Set **Log Rotation Size** to the maximum size of the active log.

   When the active log reaches this limit, it is archived.
7. Click **Save**.
8. Restart the server instance.

See Also

`log-archive-max-size` | `log-rotation-size` | `log-severity`

Related Procedures

- "Restart Server Instance" on page 1-7

Specify Server Instance License Options

1. Select the server instance from the leftmost navigation pane.
2. Select the **Settings** tab.
3. Expand the **License** area.
4. Set **License** to the server, license files, or both.

   You can specify multiple license servers including port numbers (`*port_number@license_server_name*`), as well as license files. List where you want the product to search in order of precedence, using semi-colons (;) as separators on Windows® or colons (:) as separators on Linux®.

   For example, on a Linux system, you specify this value for `license`:

   ```
   27000@hostA:/opt/license/license.dat:27001@hostB:./license.dat
   ```

   The system searches these resources in this order:

   a. `27000@hostA:` (`hostA` configured on port `27000`)
   b. `/opt/license/license.dat` (local license data file)
   c. `27001@hostB:` (`hostB` configured on port `27001`)
   d. `./license.dat` (local license data file)
5. Set **License Grace Period** to the maximum length of time MATLAB Production Server responds to HTTP requests after the license server heartbeat has been lost.
6. Set **License Poll Interval** to the interval of time that must pass:

   - After license server heartbeat has been lost and MATLAB Production Server stops responding to HTTP requests
   - Before license server is polled, to verify and check out a valid license
7. Click **Save**.
8. Restart the server instance.

See Also

`license` | `license-grace-period` | `license-poll-interval`

Related Procedures

- “Restart Server Instance” on page 1-7

Monitor Server Instances

Determine Applications Deployed on a Server Instance

To determine the applications deployed on a server instance:

1. Select the server instance from the leftmost navigation list.
2. Select the **Applications** tab.

The displayed list includes the following information:

- **Base URL** — URL of the application.
- **Application**— Name of the application.
- **Deployed On** — Date and time when the application was deployed.
- **Actions** — Decide whether to **-Undeploy** or **+Deploy** an application.

See Also

Related Procedures

- "Determine Server Instances Where Application is Deployed" on page 3-6

Common Error Messages and Resolutions

In this section...

(404) Not Found

Commonly caused by requesting a component that is not deployed on the server, or trying to call a function that is not exported by the given component.

Verify that the name of the deployable archive specified in your `Uri` is the same as the name of the deployable archive hosted in your `auto_deploy` folder.

Error: Bad MATLAB Runtime Instance

You are not properly qualifying the path to the MATLAB Runtime. You must include the version number. For example, specify:

```
C:\Program Files\MATLAB\MATLAB Compiler Runtime\vn.n
```

not

```
C:\Program Files\MATLAB\MATLAB Compiler Runtime
```

Error: invalid target host or port

The port number specified has not been properly defined to your computer. Define a valid port and retry the command.

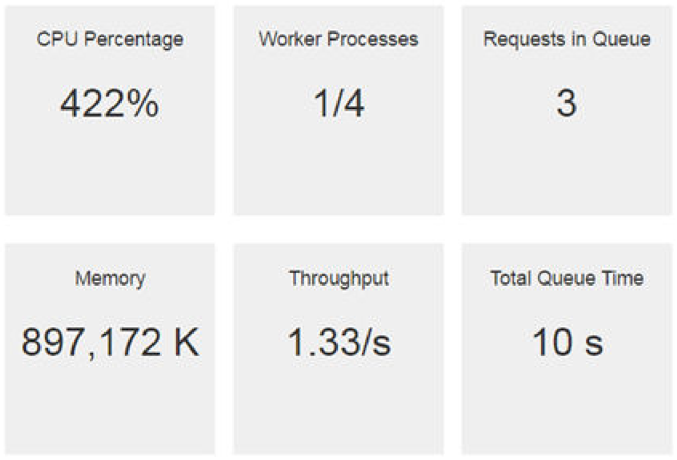
Error: HTTP error: HTTP/x.x 404 Component not found

This error can be caused by a number of reasons. Consult the log for further details on the precise cause of the problem.

View Server Instance Performance Metrics

1. Select the server instance from the leftmost navigation pane.
2. Select the **Overview** tab.

The **Overview** tab displays the following metrics:

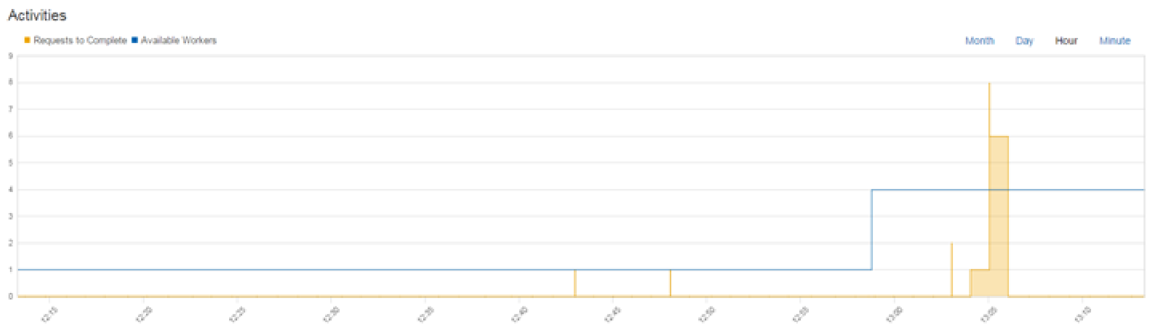


CPU Percentage	Worker Processes	Requests in Queue
422%	1/4	3
Memory	**Throughput**	**Total Queue Time**
897,172 K	1.33/s	10 s

Note Your performance metrics will be different. The image above is a sample

- **CPU Percentage** — Percentage of the server machine's CPU used by the instance. On a multi-core machine, this number can exceed 100% since it reports the cumulative CPU usage from all cores.
- **Worker Processes** — Number of active workers processing requests compared to **Maximum Workers** configured in the **Settings** tab of an instance.
- **Requests In Queue** — Number of requests waiting to be completed.
- **Memory** — Amount of memory the instance is using.
- **Throughput** — Request throughput.
- **Total Queue Time** — Total processing latency in seconds.

You can view a plot of the **Requests to Complete** and **Available Workers** over time in the **Activities** graph.



Note Your performance metrics will be different. The image above is a sample

Tip You can adjust the time scale displayed in the graph.

See Also

Verify Server Instance Status

In this section...
"Verify Server Instance Status From Navigation Tree" on page 2-6
"Verify Server Instance Status From Instance Overview" on page 2-6

Verify Server Instance Status From Navigation Tree

1 Locate the server instance in the leftmost navigation pane.
2 Verify the color of the icon representing the instance.

- A green icon specifies that the server instance is running
- A grey icon specifies that the server instance is stopped

Verify Server Instance Status From Instance Overview

1 Select the server instance from the leftmost navigation pane.
2 Select the **Overview** tab.
3 Verify the status displayed on the top of the page.

See Also

Related Procedures

- "Start Server Instance" on page 1-5
- "Start Server Instance" on page 1-5
- "Stop Server Instance" on page 1-6

Diagnose Corrupt MATLAB Runtime

1. Select the server instance from the leftmost navigation pane.
2. Select the **Logs** tab.
3. Scan the log for the following error message.

```
Dynamic exception type: class std::runtime_error
std::exception::what: bad MCR installation:
C:\Program Files\MATLAB\MATLAB Compiler Runtime\v902
(C:\Program Files\MATLAB\MATLAB Compiler Runtime\v902\bin\
win64\mps_worker_app could not be found)
```

Tip You can use the **Search** field to locate the message.

If the MATLAB Runtime is corrupt, you must reinstall it.

See Also

Related Procedures

- “Set MATLAB Runtime Location” on page 1-4

View Server Instance Log

1. Select the server instance from the leftmost navigation pane.
2. Select the **Logs** tab.

The log displays messages one entry at a time from newest to oldest.

See Also

Related Procedures

- “Manage Log Files” on page 1-17

Manage MATLAB Applications

Relationship Between Deployable Archives and MATLAB Applications

Deployable archives are the atomic packaging mechanism used by MATLAB Production Server. The Production Server Compiler app, part of the MATLAB Compiler SDK™ product, generates a deployable archive containing compiled MATLAB functions. A MATLAB Production Server instance loads a deployable archive and makes the compiled functions available at a URL containing the name of the archive and the name of the MATLAB function. MATLAB Production Server clients use the name of the deployable archive when evaluating MATLAB functions against a server instance.

The dashboard manages deployed archives using MATLAB applications.

Note Client developers must be informed of the name of the application when dashboard is being used so that they can modify the client code to use the correct deployable archive name when invoking functions.

See Also

Related Procedures

- “Add MATLAB Application” on page 3-3
- “Deploy MATLAB Application” on page 3-4
- “Undeploy MATLAB Application” on page 3-5
- “Determine Server Instances Where Application is Deployed” on page 3-6

Add MATLAB Application

MATLAB Production Server dashboard manages deployable archives in collections called applications. Adding a new application involves uploading a deployable archive. To do this:

1. Select **Applications** from the leftmost menu.
2. Click the **+Upload** button.
3. Click **Choose File** in the **Upload New Archive** pop-up.
4. In the file browser, select the deployable archive to upload.
5. In the **New Application Description** field, provide an optional description for the application.

   The description provides additional details about the function of the application. For example, the description for `derivative_risk` may be:

   ```
   Collection of functions used to asses the risks of a derivative trade. The functions include
   ```
6. Click **Upload**.

See Also

Related Procedures

- "Deploy MATLAB Application" on page 3-4

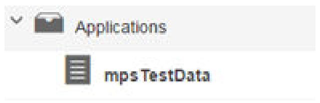
Deploy MATLAB Application

In this section...

Deploy MATLAB Application From Applications Page

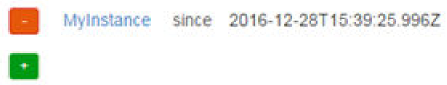
1 Select the application from the leftmost menu.

For example:



2 Click the **+** button.

For example:

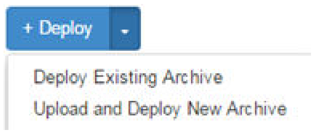


3 Select the server instance from the drop-down list.
4 Click **Deploy**.

Deploy MATLAB Application From Instance Page

1 Select the server instance from the leftmost menu.
2 Select the **Applications** tab.
3 Click the **+Deploy** button at the top.
4 Select the archive from the **Deploy Existing Archive** pop-up.
5 Confirm by clicking **Deploy**.

Note You can also upload and deploy a new archive by clicking the drop-down arrow in the **+Deploy** button.



See Also

Related Procedures

- "Undeploy MATLAB Application" on page 3-5
- "Determine Server Instances Where Application is Deployed" on page 3-6

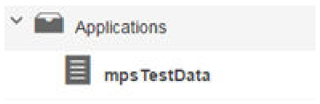
Undeploy MATLAB Application

In this section...

Undeploy MATLAB Application From Applications Page

1. Select the application from the leftmost menu.

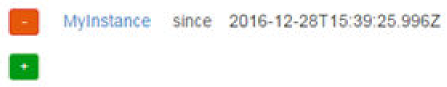
   For example:

2. Locate the server instance from which to undeploy the application.

   The server instances on which the application is deployed are listed in the **Deployed on** column.
3. Click the **-** button next to the selected instance.

   For example:

4. Confirm by clicking **Undeploy**.

Undeploy MATLAB Application From Instance Page

1. Select the server instance from the leftmost menu.
2. Select the **Applications** tab.
3. Locate the selected application in the list of deployed applications.
4. Click the **-Undeploy** button next to the selected application.
5. Confirm by clicking **Undeploy**.

See Also

Related Procedures

- "Deploy MATLAB Application" on page 3-4
- "Determine Server Instances Where Application is Deployed" on page 3-6

Determine Server Instances Where Application is Deployed

Select an application from the left-most navigation pane to display an information page. The information page lists:

- the name of the archive file
- a description
- when the application was uploaded
- when the application was deployed to a server instance

See Also

Related Procedures

- “Deploy MATLAB Application” on page 3-4
- “Undeploy MATLAB Application” on page 3-5

Set Up MATLAB Production Server Dashboard

Set Up and Log In to MATLAB Production Server Dashboard

In this section...
"Set Up the Dashboard" on page 4-2
"Log In to the Dashboard" on page 4-4
"Reset the Admin Password" on page 4-4

Follow these instructions to set up the dashboard for an on-premises installation of MATLAB Production Server.

Set Up the Dashboard

Warning You must have admin privileges on Windows to complete setup.

To set up an instance of the MATLAB Production Server dashboard:

1 Open a Terminal or Command Window with administrator privileges, and navigate to the `dashboard` folder in the MATLAB Production Server installation directory.

Platform	Default Directory Where the MATLAB Production Server dashboard is Installed
Windows *(Administrator)*	`C:\Program Files\MATLAB\MATLAB Production Server\R2020b \dashboard`
Linux	`/usr/local/MATLAB/MATLAB_Production_Server/R2020b/ dashboard`

2 Execute the script `mps-dashboard` with the `setup` option, and when prompted, specify the directory for dashboard setup. You must have write privileges to the directory from where you are running the `mps-dashboard` script, and to the directory where the dashboard is going to be set up.

Platform	Script for Dashboard Setup

Windows *(Administrator)*	`> mps-dashboard.bat setup`
	For example:
	`> mps-dashboard.bat setup`
	`Specify a workspace directory for MATLAB Production Server Dashboard: C:\mp`

Linux	`$ ./mps-dashboard.sh setup`
	For example:
	`$ ./mps-dashboard.sh setup`
	`Specify a workspace directory for MATLAB Production Server Dashboard: /opt/mps/dashboard`

You receive a message acknowledging that the dashboard has been successfully setup.

Tip To directly specify a directory when setting up the dashboard, use the `-C` option after the `setup` option and provide a directory name.

For example, in Windows: `> mps-dashboard.bat setup -C D:\mps\dashboard`

For example, in Linux: `$ ./mps-dashboard.sh setup -C /opt/mps/dashboard`

Note For a complete list of options that can be passed to the `mps-dashboard` script, pass a `?` as an option to the `mps-dashboard` script.

For example, in Windows type:

```
> mps-dashboard.bat ?
```

In Linux, type:

```
$ ./mps-dashboard.sh ?
```

The complete list of options are as follows.

```
setup | start | stop | remove | reset_admin_password
```

Tip Troubleshooting (Windows only): If the `mps-dashboard.bat setup` command fails with an error message that mentions a missing `MSVCR` DLL, add `$MPS_ROOT\bin\win64` to your system `PATH`, where `$MPS_ROOT` is the directory where you have MATLAB Production Server installed. The `$MPS_ROOT\bin\win64` folder contains DLLs that MATLAB Production Server uses.

---

3 Execute the `mps-dashboard` script with the `start` option to start the dashboard.

Platform	Script to Start Dashboard
Windows *(Administrator)*	`> mps-dashboard.bat start`
Linux	`$ ./mps-dashboard.sh start`

You will get a message indicating the host and port where the dashboard is running. The default host and port are `localhost` and `9090`, respectively.

Tip Windows only: To run the dashboard instance as a background process in Windows, precede the `mps-dashboard` script with command `start /B`.

For example: `> start /B mps-dashboard.bat start`

Note You can change the default port used by dashboard by editing the `--node_server_port` option in `config.txt` file. You can find the `config.txt` file here:

Platform	Location of `config.txt` File
Windows	`C:\Program Files\MATLAB\MATLAB Production Server\R2020b \dashboard\config\config.txt`
Linux	`/usr/local/MATLAB/MATLAB_Production_Server/R2020b/ dashboard/config/config.txt`

Other customizations to the setup process can be made by editing relevant parts of the `config.txt` file.

4. Open a web browser, and type the host and port number that were displayed in the previous step.

   For example:

   ```
   http://localhost:9090
   ```

Tip If you see garbled text in the dashboard logs, verify that the server machine uses UTF-8 encoding. You must execute `mps-dashboard.bat setup` again after setting the locale.

Log In to the Dashboard

To log in to MATLAB Production Server Dashboard follow this procedure:

1. Open a web browser, and type the host and port number that were displayed at the end of the install process.

   For example:

   ```
   http://localhost:9090
   ```
2. Type the following information at the login screen for the username and password:

   Username: `admin`

   Password: `admin`

You are now logged into the MATLAB Production Server Dashboard.

Reset the Admin Password

You can use the `mps-dashboard` script with the option `reset_admin_password` to change the admin password.

Platform	Script to Reset the Admin Password
Windows *(Administrator)*	`> mps-dashboard.bat reset_admin_password`
Linux	`$ ./mps-dashboard.sh reset_admin_password`

Warning The `reset_admin_password` option should not be executed while dashboard is still running. First, stop dashboard execution using the `mps-dashboard` script with the `stop` option and then reset the admin password.

See Also

Related Examples

- "Remove MATLAB Production Server Dashboard" on page 4-5

Remove MATLAB Production Server Dashboard

Follow these steps to remove the MATLAB Production Server dashboard in an on-premise server installation.

1. Open a Terminal or Command Window, and navigate to the `dashboard` folder in the MATLAB Production Server installation directory.

Platform	Default Directory Where MATLAB Production Server Dashboard is Installed
Windows *(Administrator)*	`C:\Program Files\MATLAB\MATLAB Production Server\R2017b \dashboard`
Linux	`/usr/local/MATLAB/MATLAB_Production_Server/R2017b/ dashboard`

2. Execute the `mps-dashboard` script with the `stop` option.

Platform	Script to Stop Dashboard
Windows *(Administrator)*	`> mps-dashboard.bat stop`
Linux	`$ ./mps-dashboard.sh stop`

   Note You need to complete this step only if dashboard is running.

3. Execute the `mps-dashboard` script with the `remove` option.

Platform	Script to Remove Dashboard
Windows *(Administrator)*	`> mps-dashboard.bat remove`
Linux	`$ ./mps-dashboard.sh remove`

   You receive a message acknowledging that dashboard was successfully removed.

   Note Attempting to remove the dashboard while it is still running will result in an error.

The procedure will remove the following directories and files from the directory where dashboard was set up:

```
data
mps_workspace
.pid
```

If you run into any issues while removing dashboard, manually delete the `.pid` file and re-run the `mps-dashboard` script with the `remove` option.

Note In Linux, if you started the dashboard using the & control operator, you don't need to open a new Terminal. The & control operator makes command run in the background.

In Windows, if dashboard is running, you will not have access to the command prompt. Therefore, you need to open a new Command Window to stop any running dashboard instances.

Note Removing dashboard does not uninstall it from the system. It removes the instance that was set up. The dashboard remains installed as part of MATLAB Production Server. If you want to set up the dashboard again, use the `mps-dashboard` script with the `setup` option.

See Also

Related Examples

- "Set Up and Log In to MATLAB Production Server Dashboard" on page 4-2

Configuration Properties

cors-allowed-origins

Specify the domain origins from which clients are allowed to make requests to the server

Description

`cors-allowed-origins` specifies the set of domain origins from which clients are allowed to make requests to a MATLAB Production Server instance. Cross-Origin Resource Sharing or CORS defines a way in which client-side web applications and a server can interact to safely determine whether or not to allow a cross-origin request. Most clients such as browsers use the `XMLHttpRequest` object to make a cross-domain request. This is especially true for client code written using JavaScript®. For MATLAB Production Server to support such requests, you must enable `cors-allowed-origins` on the server.

Parameters

`*`

Requests from any domain origin are allowed access to the sever.

`LIST`

Requests from a list of comma-separated domain origins are allowed access to the server.

Examples

Requests from any domain origin are allowed access to the sever.

```
*
```

Requests from a specific list of domain origins are allowed access to the server.

```
http://www.w3.org, https://www.apache.org
```

See Also

`http`

hide-matlab-error-stack

Hide the MATLAB stack from the clients

Description

`hide-matlab-error-stack` controls whether the MATLAB stack is exposed to the client. The stack can be sent to the client during development and debug phase, but can be turned off in production.

http

URL for insecure connections

Description

`http` specifies the interface port and optional address or host name.

Parameters

`host`

Host name or IP address of the machine running the server instance. If you do not specify the host, the server binds to any available interface.

`port`

Port number used by the server instance to accept connections. Bind to any available port by specifying `0`.

Examples

Restrict access to the HTTP interface for local clients only on port 9910.

```
9910
```

Bind to any free port.

```
0
```

Bind to a specific IP address and port.

```
234.27.101.3:9920
```

Bind to a specific host name on any free port

```
my.hostname.com:0
```

http-linger-threshold

Amount of data the server instance discards after an HTTP error and before the server instance closes the TCP connection

Description

`http-linger-threshold` sets the amount of data a server instance reads after an error. If an HTTP request is rejected and the server instance sends back an HTTP error response such as HTTP 404/413, the server instance does not close the TCP connection immediately. Instead it waits for the client to shut down the TCP connection. This ensures that the client receives the HTTP error response sent by the server instance. During this time, the server instance receives, and discards, data from the client, until the amount of data received equals `http-linger-threshold`. After that, the server instance resets the TCP connection.

By default, the threshold is unlimited and the server instance waits to receive the whole HTTP request.

Parameters

`size`

Amount of data received before the TCP connection is reset.

Examples

Set the linger threshold to be 64 MB.

```
64MB
```

Set the linger threshold to be 32 KB.

```
32KB
```

Set the linger threshold to be 1024 B.

```
1024
```

https

URL for secure connections

Description

`https` specifies the interface port and the optional address or host name to use for secure client-server communication.

Starting in R2019b, if you set the `https` property, you must set the x509-private-key and x509-cert-chain properties; otherwise, the server fails to start.

Parameters

`host`

Host name or IP address of the machine running the server instance. If you do not specify the host, the server binds to any available interface.

`port`

Port number used by the server instance to accept connections. Bind to any available port by specifying `0`.

Examples

Restrict access to the HTTPS interface for local clients only on port 9920.

```
9920
```

Bind to any free port.

```
0
```

Bind to a specific IP address and port.

```
234.27.101.3:9920
```

Bind to a specific host name on any free port.

```
my.hostname.com:0
```

See Also

x509-cert-chain | x509-private-key

Topics

"Enable Security on Server Instance" on page 1-10

license

Locations for valid licenses

Description

`license` specifies the address of the license servers or the path to the license files that a server instance uses. You can specify multiple license sources with this option.

If this option is not specified, the server searches for the license files in *`$MPS_INSTALL`*`/licenses`, where *`$MPS_INSTALL`* is the location in which MATLAB Production Server is installed.

Parameters

`pathList`

Path to one or more license servers or license files. Separate multiple entries by the appropriate path separator for the platform. Use a colon (`:`) as the path separator for Linux and semi-colon (`;`) as the path separator for Windows.

Examples

A Linux server looks for licenses using a license server hosted on port 27000 of `hostA` and in `/opt/license/license.dat`.

```
27000@hostA:/opt/license/license.dat
```

A Windows server looks for licenses using a license server hosted on port 27000 of `hostA` and in `c:\license\license.dat`.

```
27000@hostA;c:\license\license.dat
```

license-grace-period

Maximum length of time the server instance responds to HTTP requests after license server heartbeat has been lost

Description

`license-grace-period` specifies the grace period, which starts at the first heartbeat loss event. Once the grace period expires, the server instance rejects any new incoming HTTP requests.

The default grace period is 2 hours 30 minutes. The maximum value is 2 hours 30 minutes. The minimum value is 10 minutes.

Parameters

`hr`

Hours in interval.

`min`

Minutes in interval.

`sec`

Seconds in interval.

`fractSec`

Fractional seconds in interval.

Examples

The grace period lasts for 1 hour, 29 minutes, 5 seconds.

```
1:29:05
```

The grace period lasts for 10 minutes and 250 ms.

```
00:10:00.25
```

license-poll-interval

Interval of time before license server is polled to verify and check out a valid license after the grace period expires

Description

`license-poll-interval` specifies interval at which the server instance polls the license server after the license server has timed out or after the grace period has expired.

The default poll interval is 10 minutes. The minimum value is 10 minutes.

Parameters

`hr`

Hours in interval.

`min`

Minutes in interval.

`sec`

Seconds in interval.

`fractSec`

Fractional seconds in interval.

Examples

Poll for licenses at intervals of 1 hour, 29 minutes, 5 seconds.

```
1:29:05
```

Poll for licenses at intervals of 10 minutes and 250 ms.

```
00:10:00.25
```

log-archive-max-size

Maximum size of the log archive folder

Description

`log-archive-max-size` specifies the maximum size to which the log archive folder can grow before old log files are deleted.

If this property is not specified, then the log archive grows without limit.

Parameters

`size`
: Size, in bytes, of the archive folder.

Examples

Reap log archives when they reach 5 MB.

```
5MB
```

log-rotation-size

Size at which the log is archived

Description

`log-rotation-size` specifies the maximum size to which the log can grow before it is rotated into the archive area. If specified as less than 1 MB, a warning is issued and the effective size is increased to 1 MB.

No entry signifies that logs are never archived.

Parameters

`size`

Size, in bytes, of the log file.

Examples

Rotate logs when they reach 5 MB.

```
5MB
```

log-severity

Severity at which messages are logged

Description

`log-severity` specifies the level of detail at which to add information to the main log.

Parameters

`level`

Severity threshold at which messages are logged. Valid values are:

- `error` — Notification of problems or unexpected results.
- `warning` — Events that could lead to problems if not addressed.
- `information` — High-level information about major server events.
- `trace` — Detailed information about the internal state of the server.

The levels are cumulative; specifying `information` implies `warning` and `error`.

Examples

Enable all log messages.

```
trace
```

main-log-format

Text format for the main log file

Syntax

```
--main-log-format format
```

Description

`main-log-format` specifies the text format for logging events in the `main.log` file. If you do not set this property, the server writes the log data as plain text.

Parameters

`format`

Format for writing log events. Valid values are:

- `text/plain` — Log the data in plain text.
- `text/json` — Log the data in JSON format.
- `text/xml` — Log the data in XML format.

Examples

Enter this property in the **Additional Options** text box under **Settings**.

Log events in JSON format.

```
--main-log-format text/json
```

Introduced in R2020a

mcr-root

Location of a MATLAB Runtime installation

Description

`mcr-root` specifies the location of an installed MATLAB Runtime instance.

You can configure a server instance to use multiple MATLAB Runtime versions. To do so, specify the path to each MATLAB Runtime installations separated by a comma. Specify the versions from latest to the oldest. The server instance scans the list of specified `mcr-root` properties in order from first to last, then chooses the first MATLAB Runtime installation capable of processing a server request. A MATLAB Runtime installation can process a server request if it is compatible with the deployable archive containing the MATLAB function being evaluated. When you configure the server to use multiple MATLAB Runtime versions, the server uses dynamic worker pool management, where it starts the worker processes in response to demand, and stops them in response to system resource utilization. Specifying multiple MATLAB Runtime installations of the same version has no effect on performance.

Note An installation of MATLAB Production Server supports MATLAB Runtime versions up to six releases back.

Note

- Specify the path to a MATLAB Runtime installation on a local file system when configuring a server instance. Specifying a path on network partition might cause worker processes to fail.
- All values for `mcr-root` must be for the same operating system and hardware combination.

Parameters

`path`

Path to the root folder of the MATLAB Runtime installation.

Examples

Use the v98 version of the MATLAB Runtime.

```
/usr/local/MCR/v98
```

See Also

num-threads

Number of request-processing threads within the server instance

Description

`num-threads` sets the size of the thread pool available to process requests. Server instances do not allocate a unique thread to each client connection. Rather, when data is available on a connection, the required processing is scheduled on the pool of threads in the server main process.

The threads in this pool do not directly evaluate MATLAB functions. There is a single thread within each worker process that executes MATLAB code on behalf of the client.

Set this parameter to 1, and increase it only if the expected load consists of a high volume of short-running requests. This strategy ensures that the available processor resources are balanced between MATLAB function evaluation and processing client-server requests. There is usually no benefit to increasing this parameter to more than the number of available cores.

Parameters

`count`

Number of threads available in the thread pool.

This value must be one or greater.

Examples

Create a pool of 10 threads for processing requests.

```
10
```

See Also

`request-size-limit`

num-workers

Maximum number of workers allowed to process work simultaneously

Description

`num-workers` defines the number of concurrent MATLAB execution requests that a server instance can simultaneously process. It should correspond to the number of hardware threads available on the local host.

Parameters

`count`

Number of workers available to evaluate functions.

This value must be one or greater.

The maximum value is determined by the number of license keys available for MATLAB Production Server.

Examples

Allow 10 workers to process requests at a time.

```
10
```

See Also

mcr-root

profile

Log server profile information

Syntax

```
--profile state object
```

Description

`profile` logs server profile information in the main log for an *`object`* when the *`state`* is `on`. The default *`state`* is `off`, where the server does not log any profile information. You can log profile information for multiple objects by specifying multiple profile properties.

Specify this property in **Settings** > **Advanced** > **Additional Options** text box for a server instance.

Note Activating profiling has a negative impact on performance.

Parameters

`state`

Flag to control whether the server writes profile information to the main log. Valid states are:

- `on` — Log profile information.
- `off` — Do not log profile information.

`object`

Information to log. Valid objects are:

- `server` — Information about requests that the server receives and worker pool
- `server.request` — Information about server requests, which includes information about the requested archives and clients that make the requests
- `server.request.archive` — Information about archives in the request
- `server.request.client` — Information about clients that make the request
- `server.worker` and `server.worker.pool` — Information about the worker pool

The objects are hierarchical. For example, specifying `server.request` implies specifying `server.request.archive` and `server.request.client`.

If you do not specify an object, the server logs information for all the objects.

Examples

Log profile information for all the objects.

```
--profile on
```

Log profile information for all server requests only.

```
--profile on server.request
```

Log profile information for the clients in a request and workers.

```
--profile on server.request.client
--profile on server.worker
```

The following is an excerpt of the main log that contains profiling information for all objects.

```
93 [2020.03.19 13:05:56.554236] [profile] [client:[::1]:62736] [component:mymagic] [connection_id
[function:magic] [mode:sync] [request_id:0:1:1][service:http-connection] [type:request_arrive]
Request arrived and was placed in the queue.
94 [2020.03.19 13:05:56.554236] [profile]
Request to allocate next available worker
95 [2020.03.19 13:05:56.555240] [profile]
Lease created for worker-1
96 [2020.03.19 13:05:56.555240] [profile] [client:[::1]:62736] [request_id:0:1:1] [type:request_s
Request started executing on worker 1
...
99 [2020.03.19 13:05:56.558233] [profile] [client:[::1]:62736] [request_id:0:1:1] [type:request_
Request completed with HTTP status 200
100 [2020.03.19 13:05:56.558233] [profile]
Lease terminated for worker-1
101 [2020.03.19 13:05:56.558233] [profile]
worker-1 PASSED health check; returning to the pool
```

Compatibility Considerations

`requests` and `worker_pool` objects will be removed
Not recommended starting in R2020b

The objects `requests` and `worker_pool` will be removed in a future release. Use `server.request` instead of `requests` and `server.worker.pool` instead of `worker_pool` when specifying this property.

See Also

server-termination-grace-period

Duration after which all server instance processes are forcibly terminated

Syntax

```
--server-termination-grace-period hr:min:sec.fractSec
```

Description

`server-termination-grace-period` *`hr`*:*`min`*:*`sec`*.*`fractSec`* specifies the time interval after which all running server and worker processes are forcibly terminated on receipt of the `mps-stop` command, if they have not already stopped within *`hr`*:*`min`*:*`sec`*.*`fractSec`*.

If you specify the `--timeout` option when running the `mps-stop` command and have also set the `server-termination-grace-period` *`hr`*:*`min`*:*`sec`* property, server instance processes are forcibly terminated at *`hr`*:*`min`*:*`sec`*.

If you specify the both the `-k` and `--timeout` options when running the `mps-stop` command and have also set the `server-termination-grace-period` *`hr`*:*`min`*:*`sec`* property, server instance processes are forcibly terminated within the duration specified by the `--timeout` value.

Parameters

`hr`

Hours.

`min`

Minutes.

`sec`

Seconds.

`fractSec`

Fractional seconds.

Examples

Enter this property in the **Additional Options** text box under **Settings**.

Forcibly terminate server instance processes after 1 hour, 29 minutes, and 5 seconds.

```
--server-termination-grace-period 1:29:05
```

Forcibly terminate server instance processes after 10 minutes and 250 ms.

```
--server-termination-grace-period 00:10:00.25
```

Introduced in R2020a

ssl-allowed-client

MATLAB programs a client can access

Description

`ssl-allowed-client` authorizes clients based on the client certificate common name. Only authorized clients can request the evaluation of MATLAB functions.

If there are no archive names following the common name, the client can access all of the deployed archives. Otherwise, the client can access only the specified archives.

Parameters

`client`

Common name of the client.

`archive`

Name of an archive the clients can access.

Examples

Allow `client1` and `client2` to access `magic.ctf` and `helloworld.ctf`.

```
client1,client2:magic,helloworld
```

ssl-ciphers

List of cipher suites used for encryption

Description

`ssl-ciphers` provides a list of cipher suites that the server uses for encryption.

Parameters

`ciphers`

Cipher suites the server instance uses for encryption. Valid values are:

- `ALL` — Use all available cipher suites except eNULL.
- `HIGH` — Use all available high encryption cipher suites.
- *`list`* — Comma-separated list of cipher suites to use.

All OpenSSL configuration strings can be passed with the ciphers. This provides finer control over the selected cipher.

Examples

Use only high encryption cipher suites.

```
HIGH
```

Disable the use of ADH ciphers.

```
ALL:!ADH
```

Use the strongest available ECDHE ciphers.

```
ALL:@STRENGTH
```

Disable the use of ADH ciphers and use the strongest available ECDHE ciphers.

```
ALL:!ADH@STRENGTH
```

See Also

https | ssl-protocols

Topics

"Enable Security on Server Instance" on page 1-10
"Adjust Security Protocols and Ciphers Used by Server Instance" on page 1-12

ssl-protocols

List of allowed SSL protocols

Description

`ssl-protocols` lists the allowed SSL protocols. If you do not set this property, the server allows the use of all supported SSL protocols. Supported protocols are TLSv1, TLSv1.1, and TLSv1.2. The default server behavior is to attempt to use TLSv1.2.

Starting in R2019b, SSLv3 is no longer supported.

Parameters

`protocols`

Comma-separated list of allowed protocols. Valid entries are:

- `TLSv1`
- `TLSv1.1`
- `TLSv1.2`

Examples

Allow only TLSv1.

```
TLSv1
```

See Also

https | ssl-ciphers

Topics

"Enable Security on Server Instance" on page 1-10
"Adjust Security Protocols and Ciphers Used by Server Instance" on page 1-12

ssl-tmp-dh-param

File containing a pregenerated ephemeral DH key

Description

`ssl-tmp-dh-param` specifies the path to the pre-generated ephemeral DH key. If this parameter is not provided, the server instance automatically generates the DH key at start-up. Providing a pre-generated DH key can decrease instance start time.

Parameters

`path`

Path to the pre-generated DH key. Relative and absolute paths are valid.

Examples

The instance loads the DH key from `dh_param.pem` which is located at *`instance_root`*`/x509`.

```
./x509/dh_param.pem
```

ssl-tmp-ec-param

Elliptic curve used for the ECDHE ciphers

Syntax

`--ssl-tmp-ec-param` *`elliptic_curve_name`*

Description

`--ssl-tmp-ec-param` *`elliptic_curve_name`* specifies the name of the elliptic curve used for the ECDHE ciphers.

Starting in R2019b, ECDHE ciphers are enabled by default. If you do not specify the elliptic curve name, ECDHE ciphers use a default elliptic curve. The default elliptic curves are in the following order: x25519, secp256r1, x448, secp521r1, secp384r1. During the SSL/TLS handshake, the client advertises the curves that it supports. Based on this client-server negotiation, one of the default curves is used to establish a secure connection for the subsequent data exchange.

For earlier releases, if this property is not specified, all ECDHE ciphers are disabled.

Parameters

`elliptic_curve_name`

Name of curve. All curves supported by OpenSSL are supported.

Examples

Use the prime256v1 curve.

```
--ssl-tmp-ec-param prime256v1
```

ssl-verify-peer-mode

Level of client verification required by the server instance

Description

`ssl-verify-peer-mode` specifies whether the server requires clients to present a valid certificate to connect to it. Server instances allow clients to connect to it with or without providing a valid certificate. All requests will still require authorization.

If you set `ssl-verify-peer-mode` to `verify-peer-require-peer-cert`, you must set either the x509-ca-file-store or x509-use-system-store property.

Parameters

`mode`

Mode used to authenticate clients. Valid values are:

- `no-verify-peer` — No peer certificate verification. The client side does not need to provide a certificate.
- `verify-peer-require-peer-cert` — The client must provide a certificate and the certificate will be verified.

The default is `no-verify-peer`.

Examples

Require clients to provide a certificate.

```
verify-peer-require-peer-cert
```

See Also

https | x509-ca-file-store | x509-use-crl | x509-use-system-store

Topics

"Configure Client Authentication on Server Instance" on page 1-11

worker-memory-check-interval

Interval at which workers are polled for memory usage

Description

`worker-memory-check-interval` specifies how often to poll the memory usage of a worker process. This setting affects the behavior of all other settings that act based on worker memory usage such as `worker-memory-trigger`, `worker-memory-target`, and `worker-restart-memory-limit`.

Parameters

`hr`

Hours in interval.

`min`

Minutes in interval.

`sec`

Seconds in interval.

`fractSec`

Fractional seconds in interval.

Examples

Check memory usage every one and a half minutes.

```
0:01:30
```

See Also

`worker-restart-memory-limit` | `worker-restart-memory-limit-interval`

Topics

"Control Worker Restarts" on page 1-14

worker-restart-interval

Time interval at which a server instance stops and restarts its workers

Description

`worker-restart-interval` specifies the interval at which the server instance stops and restarts its worker processes. If this setting is not given, the workers are not restarted in response to time.

Parameters

`hr`

Hours in interval.

`min`

Minutes in interval.

`sec`

Seconds in interval.

`fractSec`

Fractional seconds in interval.

Examples

Restart workers at intervals of 1 hour, 29 minutes, 5 seconds.

```
1:29:05
```

Restart workers at intervals of 10 minutes and 250 ms.

```
00:10:00.25
```

See Also

Topics
"Control Worker Restarts" on page 1-14

worker-restart-memory-limit

Size threshold at which to consider restarting a worker

Description

`worker-restart-memory-limit` sets the memory usage limit of a worker process. If a worker's working set size exceeds `worker-restart-memory-limit` for an interval of time greater than `worker-restart-memory-limit-interval`, then that worker is restarted.

Parameters

`size`

Amount of memory used by worker.

Examples

Restart any worker whose working set size exceeds 1 GB for more than 1 hour.

```
Worker Restart Memory Limit 1GB
Worker Eestart Memory Limit Interval 1:00:00
```

See Also

`worker-memory-check-interval` | `worker-restart-memory-limit-interval`

Topics

"Control Worker Restarts" on page 1-14

worker-restart-memory-limit-interval

Interval for which a worker can exceed its memory limit before restart

Description

`worker-restart-memory-limit-interval` sets the interval for which a worker process can exceed its memory limit before restart. If a worker's working set size exceeds `worker-restart-memory-limit` for an interval of time greater than `worker-restart-memory-limit-interval`, then that worker is restarted.

Parameters

hr

Hours in interval.

min

Minutes in interval.

sec

Seconds in interval.

fractSec

Fractional seconds in interval.

Examples

Restart any worker whose working set size exceeds 1 GB for more than 1 hour.

```
Worker Restart Memory Limit 1GB
Worker Restart Memory Limit Interval 1:00:00
```

See Also

`worker-memory-check-interval` | `worker-restart-memory-limit`

Topics

"Control Worker Restarts" on page 1-14

x509-ca-file-store

File containing the server certificate authority file

Description

`x509-ca-file-store` specifies the certificate authority (CA) file to verify peer certificates. This file contains trusted certificates and certificate revocation lists.

You can also put intermediate certificates into the CA file. An intermediate certificate in the CA file becomes a trusted certificate.

Parameters

`path`

Path to the certificate CA file store. Relative and absolute paths are valid.

Examples

The instance loads the CA store from `ca_file.pem` which is located at *`instance_root`*`/x509`.

```
./x509/ca_file.pem
```

See Also

https | ssl-verify-peer-mode | x509-use-crl | x509-use-system-store

Topics

"Configure Client Authentication on Server Instance" on page 1-11

x509-cert-chain

File containing the server certificate chain

Description

`x509-cert-chain` specifies the server certificate chain file. It contains one or more PEM-format certificates. The chain begins with the server certificate. The server certificate is followed by a chain of untrusted certificates. To use the certificate chain file, specify the .

Starting in R2019b, if is enabled on the server, you must set the `x509-cert-chain` and `x509-cert-chain` properties; otherwise, the server fails to start.

Note Do not specify trusted certificates in the certificate chain file.

Parameters

`path`

Path to the certificate chain file. Relative and absolute paths are valid.

Examples

The instance loads the certificate chain from `cert_chain.pem` which is located at *`instance_root/`*`x509`.

```
./x509/cert_chain.pem
```

See Also

https | x509-private-key

Topics

"Enable Security on Server Instance" on page 1-10

x509-passphrase

File containing the passphrase that decodes the private key

Description

`x509-passphrase` specifies the path to the file containing the passphrase of the encrypted private-key. This is required if `x509-private-key` is specified and the private key file is encrypted. Otherwise, the private key fails to load.

Note This file must be owner read-only.

Parameters

`path`

Path to the passphrase file. Relative and absolute paths are valid.

Examples

The instance loads the passphrase from `key_passphrase.pem` which is located at *`instance_root`*`/x509`.

```
./x509/key_passphrase.pem
```

x509-private-key

File containing the private key in PEM format

Description

`x509-private-key` specifies the path to the private key. The key must be in PEM format.

If you do not set this property, the server instance does not load the private key or the server-side certificates.

Starting in R2019b, if https is enabled on the server, you must set the `x509-private-key` and `x509-cert-chain` properties; otherwise, the server fails to start.

Parameters

`path`

Path to the PEM-format private key file. Relative and absolute paths are valid.

Examples

The instance loads the private key from `private_key.pem`, which is located at *`instance_root/`*`x509`.

```
./x509/private_key.pem
```

See Also

https | x509-cert-chain

Topics

"Enable Security on Server Instance" on page 1-10

x509-use-crl

Use the certificate revocation list

Description

`x509-use-crl` specifies that the server instance uses the certificate revocation list (CRL). By default, instances do not use any CRLs. In this case, the CRLs in the certificate authority store are ignored.

If `x509-use-crl` is added, the CRLs are loaded and participate in the client certificate verification. If the CRL has expired, the SSL handshake is rejected.

See Also

https | ssl-verify-peer-mode | x509-ca-file-store | x509-use-system-store

Topics

"Configure Client Authentication on Server Instance" on page 1-11

x509-use-system-store

Use the certificate authority store provided by the system

Description

`x509-use-system-store` specifies that the server instance uses the system provided certificate authority (CA) store. By default, the server uses the file `/etc/ssl/certs/ca-certificates.crt` as trusted CA store and searches for trusted certificates under the folder `/etc/ssl/certs`. You can override these locations by setting the environment variables `SSL_CERT_FILE` and `SSL_CERT_DIR`.

See Also

https | ssl-verify-peer-mode | x509-ca-file-store | x509-use-crl

Topics

"Configure Client Authentication on Server Instance" on page 1-11

use-single-comp-thread

Start MATLAB Runtime with a single computational thread

Syntax

`--use-single-comp-thread`

Description

`--use-single-comp-thread` specifies that workers start the MATLAB Runtime with a single computational thread.

Examples

Start the MATLAB Runtime with a single computational thread.

```
--use-single-comp-thread
```

request-timeout

Duration after which the request times out and gets deleted after reaching a terminal state

Description

`request-timeout` specifies the duration after which the request times out upon reaching a terminal state. At this point, the request gets deleted unless a client process has already deleted the request.

Parameters

`hr`

Hours in interval.

`min`

Minutes in interval.

`sec`

Seconds in interval.

`fractSec`

Fractional seconds in interval.

Examples

Set the request to time-out after 2 hours.

```
2:00:00
```

See Also

`server-memory-threshold`

Introduced in R2016b

server-memory-threshold

Size threshold of server process at which action needs to be taken to manage responses

Description

`server-memory-threshold` sets the memory size limit of a server process. If a server process's size exceeds `SIZE` set by `server-memory-threshold`, then responses need to be either archived or purged by setting `server-memory-threshold-overflow-action` to `archive_responses` or `purge_responses` respectively. If `server-memory-threshold` is not set, then the responses will be bound to the server process causing the memory footprint of the server process to keep increasing. As a best practice, it is recommended that a client process delete a request after usage in order to prevent the memory of the server process from growing.

Parameters

`size`

Threshold size of the server process.

Examples

Archive responses if the size of the server process in memory exceeds 500 MB.

```
Server Memory Threshold  500MB
Server Memory Threshold Overflow Action archive_responses
```

Purge responses if the size of the server process in memory exceeds 2 GB.

```
Server Memory Threshold  2GB
Server Memory Threshold Overflow Action purge_responses
```

See Also

`server-memory-threshold-overflow-action`

server-memory-threshold-overflow-action

Action taken when the memory size threshold of server process is breached

Description

`server-memory-threshold-overflow-action` acts by either archiving responses or purging them when the size of the server process in memory set by `server-memory-threshold` is breached.

Parameters

ACTION

`archive_responses`

`purge_responses`

Examples

Archive responses if the size of the server process in memory exceeds 500 MB.

```
Server Memory Threshold  500MB
Server Memory Threshold Overflow Action archive_responses
```

Purge responses if the size of the server process in memory exceeds 2 GB.

```
Server Memory Threshold  2GB
Server Memory Threshold Overflow Action purge_responses
```

See Also

`server-memory-threshold`

response-archive-root

Path to the location where responses are archived

Description

`response-archive-root` shows the location where responses specified by *PATH* are archived. This option must be set if the server-memory-threshold-overflow-action option is set to `archive_responses`. The server process must have read & write permissions to the *PATH*.

Parameters

PATH

Location specified as a string.

Examples

Set the archive root.

```
./.response_archive
```

See Also

`response-archive-limit`

response-archive-limit

Maximum disk space available to the server process for archiving

Description

`response-archive-limit` specifies the maximum disk space available to the server process for archiving. If the limit set by ***`SIZE`*** is reached, the archives will be deleted in a 'First-In First-Out' order until the space for the server process fall below ***`SIZE`***. If this limit is not specified, the server process will assume there is no limit to disk usage for archiving.

Parameters

`size`

Size of the server process.

Examples

Set the size of the archive to be 5GB

```
5GB
```

See Also

`response-archive-root`

Introduced in R2016b

request-size-limit

Set the maximum size of a request

Description

`request-size-limit` specifies the maximum size of a request specified by *`size`*. The default request size is 64MB.

Parameters

`size`

Size, in bytes, of the request.

Examples

Set the request size to 128MB.

```
128MB
```

See Also

`num-threads`

user-data

Associate MATLAB data value with string key

Description

`user-data` associates MATLAB data value with key string. *`KEY`* and *`VALUE`* are strings. Use the double quotes (`"`) character around strings with spaces. The backslash (`\`) character is the escape character and is used to insert double quotes or backslash characters: `\" \\`. The application can retrieve the data value by using `getmcruserdata(key)`.

Parameters

`KEY VALUE`

MATLAB *`value`* to be associated with *`key`*.

Examples

Set user data with parallel profile settings.

```
ParallelProfile c:\\MPS\\myprofile.settings
```

Use quotes.

```
MyValue "Quoted string with escaped \"quotes\" and \\backslash."
```

See Also

Introduced in R2016a

Persistence

Create a Persistence Service

Local Service

1. Select **Persistence** from the left navigation tree.
2. Click the down arrow next to the **+ Create** button.
3. Select **Local Service**.
4. Enter details in the **Create Local Persistence Service** dialog.

   - **Provider**: The persistence provider defaults to Redis™. Currently, Redis is the only supported provider.
   - **Port**: Enter a port number that you want the service to use. Only non-SSL ports are supported.
   - **Persistence Configuration File***(optional)*: Enter the persistence provider configuration file only if you want to override any of the default settings of the persistence provider. For Redis, the configuration file is called `redis.conf` and on Windowscan be found in `C:\Program Files\MATLAB\MATLAB Production Server\R2020b\bin\win64`.
5. Click **Create**.

Remote Service

Note When you create a remote persistence service, you are actually creating a connection to a remote persistence service.

1. Select **Persistence** from the left navigation tree.
2. Click the down arrow next to the **+ Create** button.
3. Select **Remote Service**.
4. Enter details in the **Create Remote Persistence Service** dialog.

   - **Provider**: The persistence provider defaults to Redis. Currently, Redis is the only supported provider.
   - **Host**: Type the remote host name. For example: `persistence.mathworks.com`.
   - **Port**: Enter a port number that you want the service to use. Only non-SSL ports are supported.
   - **Access Key**: Access key string to connect to an Azure® Redis Cache instance obtained from the Azure portal.
5. Click **Create**.

Tip To retrieve an access key to connect to an Azure Redis Cache instance:

- Log in to your Azure portal and select yourAzure Redis Cache instance.
- Select **Overview** and under **Keys** click **Show access keys**.
- In the resulting blade, copy the access key string listed under **Primary**.

See Also

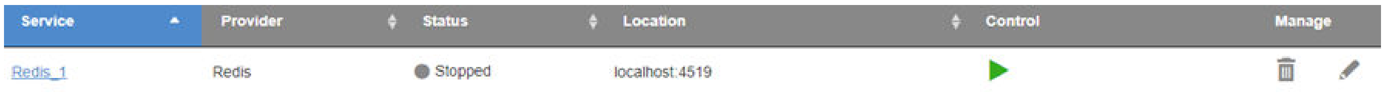
More About

- “Start a Persistence Service” on page 6-4
- “Stop a Persistence Service” on page 6-5
- “Restart a Persistence Service” on page 6-6
- “Delete a Persistence Service” on page 6-7

Start a Persistence Service

1 Select **Persistence** from the left navigation tree.
2 Locate the service in the list.
3 Click the green arrow start button in the **Control** column.

For example:



Service	Provider	Status	Location	Control	Manage
Redis_1	Redis	● Stopped	localhost:4519	▶	

See Also

More About

- “Stop a Persistence Service” on page 6-5
- “Restart a Persistence Service” on page 6-6
- “Delete a Persistence Service” on page 6-7
- “Create a Persistence Service” on page 6-2

Stop a Persistence Service

1. Select **Persistence** from the left navigation tree.
2. Locate the service in the list.
3. Click the orange square stop button in the **Control** column.

See Also

More About

- “Start a Persistence Service” on page 6-4
- “Restart a Persistence Service” on page 6-6
- “Delete a Persistence Service” on page 6-7
- “Create a Persistence Service” on page 6-2

Restart a Persistence Service

1 Select **Persistence** from the left navigation tree.
2 Locate the service in the list.
3 Click the dark gray circle restart button in the **Control** column.

See Also

More About

- “Start a Persistence Service” on page 6-4
- “Stop a Persistence Service” on page 6-5
- “Delete a Persistence Service” on page 6-7
- “Create a Persistence Service” on page 6-2

Delete a Persistence Service

1 Select **Persistence** from the left navigation tree.
2 Locate the service in the list.
3 Click the trash can button in the **Manage** column.

See Also

More About

- "Create a Persistence Service" on page 6-2
- "Start a Persistence Service" on page 6-4
- "Stop a Persistence Service" on page 6-5
- "Restart a Persistence Service" on page 6-6

Add a Persistence Service to a Server Instance

Add a Persistence Service from the Persistence Overview Page

1. Select **Persistence** from the left navigation tree.
2. Click the service in the list you want to add to a server instance
3. Click the green **+** button.
4. In the **Add Connection** dialog, from the drop-down menu, select the server instance to which you want to add the persistence service.
5. Specify a *connection name*. A valid connection name starts with a letter, followed by letters, digits, or underscores. MATLAB is case sensitive, so `A` and `a` are *not* the same variable. The maximum length is 63 characters.

Add a Persistence Service from a Server Instance Page

1. Select the server instance from the navigation pane.
2. Select the **Persistence** tab.
3. Click the **+ Add** button.
4. Select a persistence service to add to the server instance.
5. Specify a *connection name*. A valid connection name starts with a letter, followed by letters, digits, or underscores. MATLAB is case sensitive, so `A` and `a` are *not* the same variable. The maximum length is 63 characters.

Note A persistence service is connected to a server instance using a connection name. This connection name is used in MATLAB code to create data caches for a particular server instance. It can either be hard coded within MATLAB code or passed in as a parameter by a client application.

See Also

More About

- "Connect a Persistence Service to a Server Instance" on page 6-9
- "Create a Persistence Service" on page 6-2

Connect a Persistence Service to a Server Instance

Note A persistence service must be started and running before you can connect it to a server instance. For more information, see, “Start a Persistence Service” on page 6-4.

1 Select the server instance from the navigation pane.
2 Select the **Persistence** tab.
3 Locate the connection name that you want to use to connect the persistence service to the server instance.
4 Click the paperclip button in the **Actions** column to attach the connection to the server instance and enable the connection.

See Also

More About

- “Disconnect a Persistence Service from a Server Instance” on page 6-10
- “Create a Persistence Service” on page 6-2

Disconnect a Persistence Service from a Server Instance

Note A persistence service must be started and running before you can disconnect it from a server instance. For more information, see, “Start a Persistence Service” on page 6-4.

1. Select the server instance from the navigation pane.
2. Select the **Persistence** tab.
3. Locate the connection name that you want to use to disconnect the persistence service from the server instance.
4. Click the scissors button in the **Actions** column to detach the connection from the server instance and disable the connection.

See Also

More About

- “Connect a Persistence Service to a Server Instance” on page 6-9
- “Create a Persistence Service” on page 6-2

Edit a Persistence Service Configuration

A persistence service must be stopped before editing it's configuration. For more information, see, “Stop a Persistence Service” on page 6-5.

1. Select **Persistence** from the left navigation tree.
2. Locate the service in the list whose configuration you want to edit.
3. Click the pencil button in the **Manage** column to edit the configuration.

See Also

More About

- “Create a Persistence Service” on page 6-2