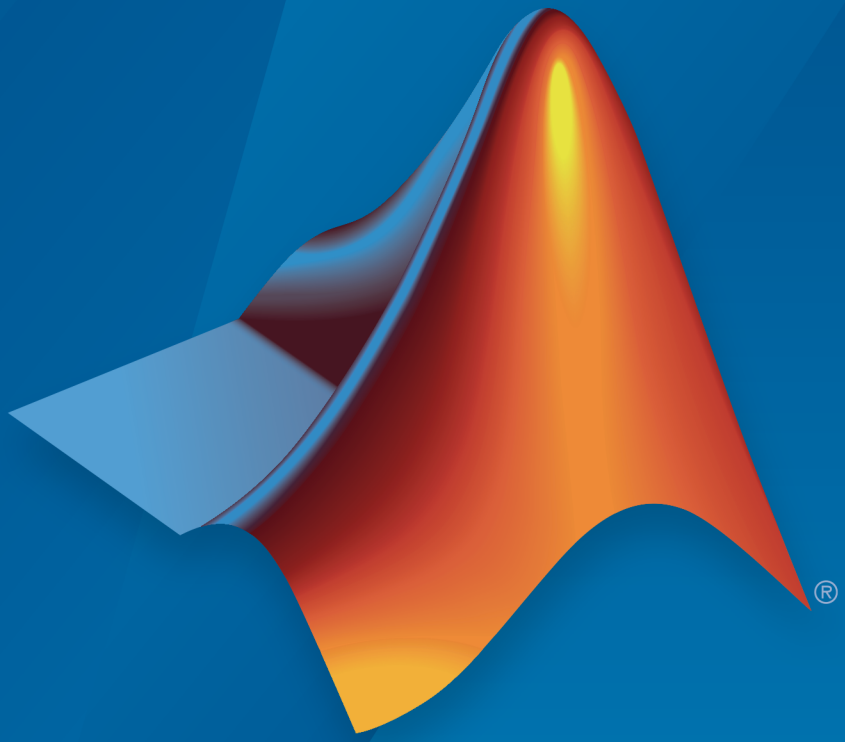
MATLAB® Production Server™ Dashboard

Dashboard User Guide



MATLAB®

R2017a

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.
3 Apple Hill Drive
Natick, MA 01760-2098

MATLAB® Production Server™ Dashboard User Guide

Revision History

March 2017	Online only	New for Version 3.0 (Release R2017a)

Contents

1 Manage Server Instances

2 Monitor Server Instances

3 Manage MATLAB Applications

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
   ```
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 1-19.

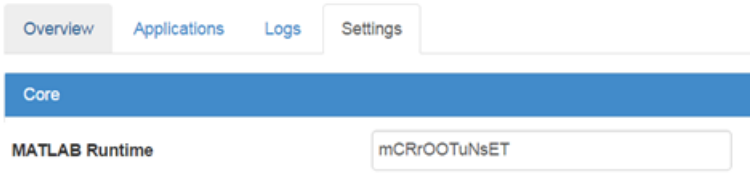
Failure to complete these steps will generate errors.

Set MATLAB Runtime Location

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

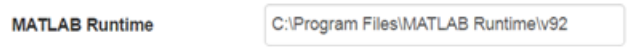
   For example: **Servers** > **localhost** > **myinstance**
2. Select the **Settings** tab.

3 Expand the **Core** area.



4 Replace the value `mCRrOOTuNsET` in the field **MATLAB Runtime** with the location of your MATLAB Runtime.

For example:



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 setting
```

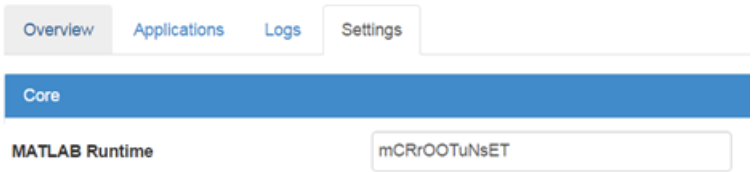
Related Procedures

- “Set MATLAB Runtime Location” on page 1-4
- “Specify Server Instance License Options” on page 1-19
- “Start Server Instance” on page 1-5

Set MATLAB Runtime Location

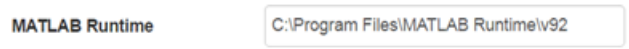
1. Select the server instance from the leftmost navigation pane.

   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:



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 setting
```

Related Procedures

- “Specify Server Instance License Options” on page 1-19
- “Start Server Instance” on page 1-5

Start Server Instance

In this section...

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.

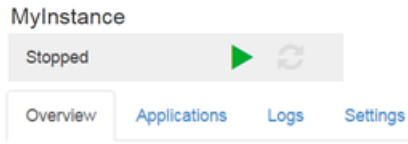
   For example:

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.

   For example:



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 1-19.

Failure to complete these steps will generate errors.

Related Examples

- “Set MATLAB Runtime Location” on page 1-4
- “Specify Server Instance License Options” on page 1-19
- “Create Server Instance” on page 1-2
- “Stop Server Instance” on page 1-7

Stop Server Instance

In this section...

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:

Related Procedures

- “Start Server Instance” on page 1-5
- “Restart Server Instance” on page 1-8
- “Remove Server Instance” on page 1-9

Restart Server Instance

In this section...

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.

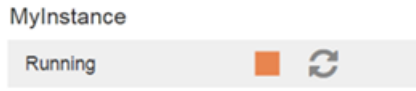
3 Click the dark gray circle restart button in the **Actions** column.

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:



Related Procedures

- “Stop Server Instance” on page 1-7
- “Remove Server Instance” on page 1-9

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.

Related Procedures

- "Stop Server Instance" on page 1-7

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.

Related Procedures

- "Restart Server Instance" on page 1-8

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 1-8

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 1-8

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 disable SSL protocols, set the property to `TLSv1`.

   Because SSLv2 and SSLv3 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 1-8

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 1-8

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 1-8

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 1-8

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 1-8

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 1-8

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.

Related Procedures

- "Determine Server Instances Where Application is Deployed" on page 3-9

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\v`*`n.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.

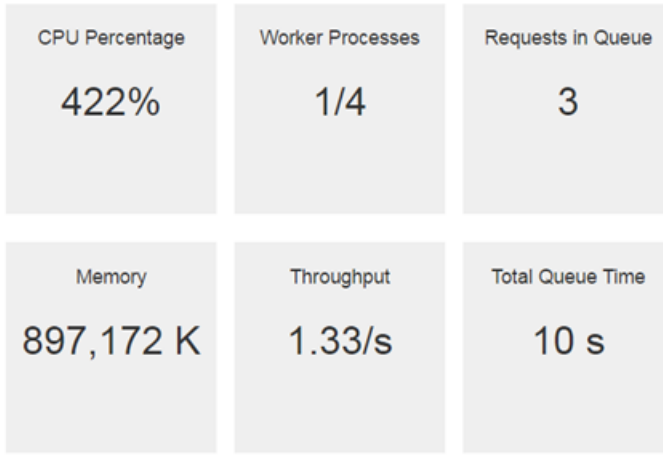
Error: HTTP error: HTTP/x.x 404 Component not found

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.

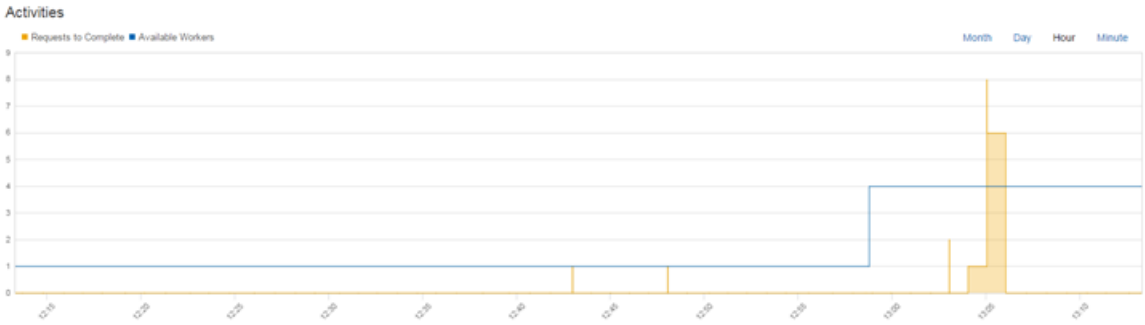
The **Overview** tab displays the following metrics:



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.

Verify Server Instance Status

In this section...

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.

Related Procedures

- "Start Server Instance" on page 1-5
- "Start Server Instance" on page 1-5
- "Stop Server Instance" on page 1-7

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.

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.

Related Procedures

- "Manage Log Files" on page 1-18

3

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.

Related Procedures

- "Add MATLAB Application" on page 3-3
- "Deploy MATLAB Application" on page 3-4
- "Undeploy MATLAB Application" on page 3-6
- "Determine Server Instances Where Application is Deployed" on page 3-9

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 function
   ```
6. Click **Upload**.

Related Procedures

- "Deploy MATLAB Application" on page 3-4

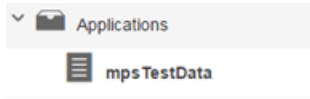
Deploy MATLAB Application

In this section...

Deploy MATLAB Application From Applications Page

1 Select the application from the leftmost menu.

For example:



2 Click the **+** button.

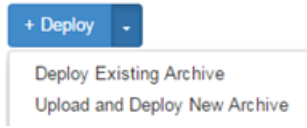
For example:

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.



Related Procedures

- "Undeploy MATLAB Application" on page 3-6
- "Determine Server Instances Where Application is Deployed" on page 3-9

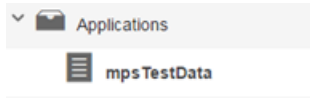
Undeploy MATLAB Application

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**.

Related Procedures

- “Deploy MATLAB Application” on page 3-4

- “Determine Server Instances Where Application is Deployed” on page 3-9

Update MATLAB Application

In this section...

Add New Version to Application

When you add a new version to an application, you add a new deployable archive to the collection.

1. Select the application from the leftmost menu.
2. Click **+New Version**.
3. In the file browser that opens, select the deployable archive to upload.
4. In the **Comment** field, provide an optional reason for uploading the new version.

   The comment provides details about the differences between the newer version and the older versions. For example, a useful comment may be:

   ```
   Updated application to accept a thrid variance paremeter.
   ```
5. Click **Upload**.

Remove Version from Application

When you remove a version from an application, you delete a deployable archive from the collection.

1. Select the application from the leftmost menu.
2. Locate the version to be removed.
3. Click the trash can icon button next to the selected version.
4. Confirm by clicking **Delete**.

Related Procedures

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

Related Procedures

- "Deploy MATLAB Application" on page 3-4
- "Undeploy MATLAB Application" on page 3-6

Set Up MATLAB Production Server Dashboard

Set Up and Log In to MATLAB Production Server Dashboard

In this section...

Set Up the Dashboard

Warning: You must have admin or superuser privileges to complete setup.

To set up MATLAB Production Server Dashboard:

1. Create a directory in your system where you want the dashboard to be set up.

   For example:

Platform	Create a Directory Where You Want Dashboard to be Set Up
Windows *(Administrator*	`> mkdir C:\mps\dashboard`
Linux	`$ sudo mkdir /opt/mps/dashboard`

2. 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 \R2017a\dashboard`
Linux	`/usr/local/MATLAB/MATLAB_Production_Server/R2017a/ dashboard`

3. Execute the `setup` script with the `configure` option, and when prompted, specify the directory for dashboard setup. This is the directory created in Step 1. If the directory does not exist or you failed to create it in Step 1, create the directory before executing the `setup` script. Otherwise, set up will fail. You must have write

privileges to the directory from where you are running the `setup` script, and to the directory where the dashboard is going to be set up.

Platform	Script to Configure the Directory for Dashboard Setup

Windows *(Administrator*	`> setup.bat configure`
	
	For example:
	
	`> setup.bat configure`
	`Setup needs to configure an installation directory.`
	`Specify an installation directory for MATLAB Production Server Dashbo`

Linux	`$ sudo ./setup.sh configure`
	
	For example:
	
	`$ sudo ./setup.sh configure`
	`Setup needs to configure an installation directory.`
	`Specify an installation directory for MATLAB Production Server Dashbo`

You receive a message acknowledging the successful configuration of the directory for dashboard setup.

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

For example, in Windows type:

```
> setup.bat ?
```

In Linux, type:

```
$ sudo ./setup.bat ?
```

The complete list of options are:

```
configure | install | start | stop | uninstall |
reset_admin_password
```

---

4 Execute the `setup` script with the `install` option to install the Dashboard workspace files in the configured directory.

Platform	Script to Install Dashboard Workspace Files
Windows *(Administrator*	`> setup.bat install`
Linux	`$ sudo ./setup.sh install`

You receive a message acknowledging that the dashboard workspace files were successfully installed.

5 Execute the `setup` script with the `start` option to start Dashboard.

Platform	Script to Start Dashboard
Windows *(Administrator*	`> setup.bat start`
Linux	`$ sudo ./setup.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.

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 \R2017a\dashboard\config\config.txt`
Linux	`/usr/local/MATLAB/MATLAB_Production_Server/R2017a/ dashboard/config/config.txt`

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

---

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

For example:

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

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 `setup` script with the option `reset_admin_password` to change the admin password.

Platform	Script to Start Dashboard
Windows *(Administrator)*	`> setup.bat reset_admin_password`
Linux	`$ sudo ./setup.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 `setup` script with the `stop` option and then reset the admin password.

Related Examples

- “Uninstall MATLAB Production Server Dashboard” on page 4-6

Uninstall MATLAB Production Server Dashboard

To uninstall MATLAB Production Server Dashboard:

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 \R2017a\dashboard`
Linux	`/usr/local/MATLAB/MATLAB_Production_Server/R2017a/ dashboard`

2. Execute the `setup` script with the `stop` option.

Platform	Script to Stop Dashboard
Windows *(Administrator*	`> setup.bat stop`
Linux	`$ sudo ./setup.sh stop`

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

3. Execute the `setup` script with the `uninstall` option.

Platform	Script to Uninstall Dashboard
Windows *(Administrator*	`> setup.bat uninstall`
Linux	`$ sudo ./setup.sh uninstall`

You receive a message acknowledging that dashboard was successfully uninstalled.

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

The uninstall procedure will remove the following directories and files from the dashboard configured directory:

```
data
mps_workspace
.pid
```

If you run into any issues while uninstalling dashboard, manually delete the `.pid` file and re-run the `setup` script with the `uninstall` 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.

Related Examples

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

5

Configuration Properties—Alphabetical List

auto-deploy-root

Folder the server instance scans for deployable archives

Syntax

```
--auto-deploy-root path
```

Description

`--auto-deploy-root` *`path`* specifies the folder the server instance scans for deployable archives. Deployable archives placed in this folder are automatically unpacked and deployed when the instance is started. No restart is necessary when a deployable archive is added, updated, or removed. Many instances may share a single `auto-deploy-root`. Using this folder allows near-simultaneous hot deployment to multiple instances. The folder is scanned every five seconds for changes.

Parameters

`path`

Path to the folder scanned for deployable archives relative to the server instance's root folder.

Examples

Scan the `auto_deploy` folder for deployable archives to hot deploy.

```
--auto-deploy-root ./auto_deploy
```

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 domains 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 `cors-allowed-origins` must be enabled 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.

See Also

See Also

`http`

disable-control-c

Disable keyboard interruptions for server instance

Description

`disable-control-c` disables keyboard interruption for the server instance. The server instance does not respond to **CTRL-C**.

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 that server instance uses 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 not specified, the server binds to any available interface.

`port`

Port number the server instance uses to accept connections. `0` specifies bind to any available port.

Examples

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

```
localhost: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 that server instance uses for secure connections

Description

`https` 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 not specified, the server binds to any available interface.

`port`

Port number the server instance uses to accept connections. `0` specifies bind to any available port.

Examples

Restrict access to the HTTP interface to local clients only, port 9920.

```
localhost: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
```

license

Locations searched for valid licenses

Description

`license` specifies the license servers or the license files used by the server instance. You can specify multiple license sources with this option.

If this option is not specified, the server searches in the default locations for the license files.

Parameters

`pathList`

Path to one or more license servers or license files. Multiple entries are separated by the appropriate path separator for the platform.

Examples

A Unix 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
```

mcr-root

Location of a MATLAB Runtime installation

Description

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

Note:

- A server instance should only be configured to use MATLAB Runtime roots on a local file system. Otherwise, a network partition may cause worker processes to fail.

Parameters

`path`

Path to the root folder of the MATLAB Runtime installation.

Examples

Use the v80 version of the MATLAB Runtime.

```
/usr/local/MCR/v80
```

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
```

num-workers

Maximum number of workers allowed to process work simultaneously

Description

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

Parameters

`count`

Number of workers available 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
```

profile

Turn profiling on or off

Description

`profile` turns profiling on or off.

Note: Activating profiling has a negative impact on performance.

---

ssl-allowed-clients

MATLAB programs a client can access

Description

`ssl-allowed-clients` 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 to use

Description

`ssl-ciphers` provides a list of cipher suites the server instance can use.

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
```

ssl-protocols

List of allowed SSL protocols

Description

`ssl-protocols` lists the allowed protocols. The default behavior is to attempt to use TLSv1.2. If this property is not set, the server allows all protocols.

Parameters

`protocols`

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

- `TLSv1`
- `SSLv2`
- `SSLv3`

Examples

Allow only TLSv1.

```
TLSv1
```

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

Elliptical curve used in ECDHE ciphers

Syntax

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

Description

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

If this property is not specified, all ECDHE ciphers will be removed from the list of ciphers available for secure communication.

Parameters

`elliptic_curve_name`

Named 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 the server instance requires

Description

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

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
```

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

See Also

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

Topics

"Control Worker Restarts" on page 1-15

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 1-15

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

See Also

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

Topics

“Control Worker Restarts” on page 1-15

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

See Also

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

Topics

"Control Worker Restarts" on page 1-15

x509-ca-file-store

File containing the server certificate authority file

Description

`x509-ca-file-store` specifies the 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
```

x509-cert-chain

File containing the server certificate chain

Description

`x509-cert-chain` specifies the file storing the server certificate chain file. It contains one or more PEM formatted certificates. The chain begins with the server's certificate. The server's certificate is followed by the chain of untrusted certificates. To use the certificate chain file, specify `x509-private-key`.

Note: Trusted certificates should not be put into this file.

Parameters

`path`

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

Examples

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

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

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 PEM formatted private key

Description

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

If it is not configured, the server instance does not load the private key and the server-side certificates.

Parameters

`path`

Path to the PEM formatted 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
```

x509-use-crl

Use the certificate revocation list

Description

`x509-use-crl` specifies that the server instance uses the certificate revocation list. By default, instances do not use any certificate revocation lists. In case, the CRLs in the CA 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.

x509-use-system-store

Use the CA store provided by the system

Description

`x509-use-system-store` specifies that the server instance uses the system provided 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`.

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.

See Also

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.

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

See Also

See Also

`server-memory-threshold-overflow-action`

server-memory-threshold-overflow-action

Action to be taken when memory size threshold of server process has been 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` has been breached.

Parameters

ACTION

`archive_responses`

`purge_responses`

Examples

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

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

See Also

See Also

`server-memory-threshold`

response-archive-root

Path to location where responses are archived

Description

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

Parameters

PATH

Location specified as a string.

Examples

Set the archive root.

See Also

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

See Also

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.

See Also

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.

Use quotes.

See Also

Introduced in R2016a