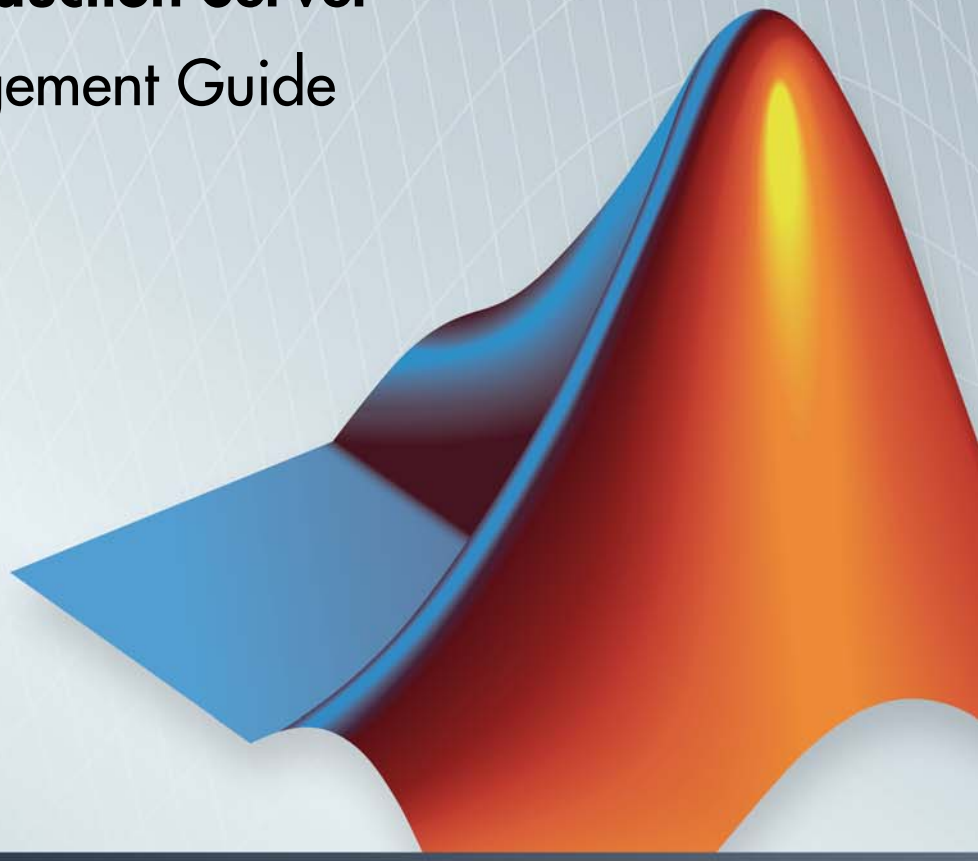
MATLAB® Production Server™

Server Management Guide

R2014**a**



MATLAB®

MathWorks®

How to Contact MathWorks

`www.mathworks.com`	Web
`comp.soft-sys.matlab`	Newsgroup
`www.mathworks.com/contact_TS.html`	Technical Support

`suggest@mathworks.com`	Product enhancement suggestions
`bugs@mathworks.com`	Bug reports
`doc@mathworks.com`	Documentation error reports
`service@mathworks.com`	Order status, license renewals, passcodes
`info@mathworks.com`	Sales, pricing, and general information

508-647-7000 (Phone)

508-647-7001 (Fax)

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

For contact information about worldwide offices, see the MathWorks Web site.

MATLAB® Production Server™ Management Guide

Trademarks

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

Patents

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

Revision History

March 2014	Online only	New for Version 1.2 (Release R2014a)

Contents

1 Server Management

2 Commands — Alphabetical List

1

Server Management

Server Overview

In this section...

What is a Server?

You can create any number of server instances using MATLAB® Production Server™ software. Each server instance can host any number of deployable archives containing MATLAB code. You may find it helpful to create one server for all archives relating to a particular application. You can also create one server to host code strictly for testing, and so on.

A *server instance* is considered to be one unique *configuration* of the MATLAB Production Server product. Each configuration has its own options file (`main_config`) and diagnostic files (`log` files, Process Identification (`pid`) files, and `endpoint` files).

In addition, each server has its own `auto_deploy` folder, which contains the deployable archives you want the server to host for clients.

The server also manages the MATLAB Compiler Runtime (MCR), which enables MATLAB code to execute. The settings in `main_config` determine how each server interacts with the MCR to process clients requests. You can set these parameters according to your performance requirements and other variables in your IT environment.

How Does a Server Manage Work?

A server processes a transaction using these steps:

1. The client sends MATLAB function calls to the master server process (the main process on the server).
2. MATLAB function calls are passed to one or more *MCR workers* (An MCR session).
3. MATLAB functions are executed by the MCR worker.

4. Results of MATLAB function execution are passed back to the master server process.

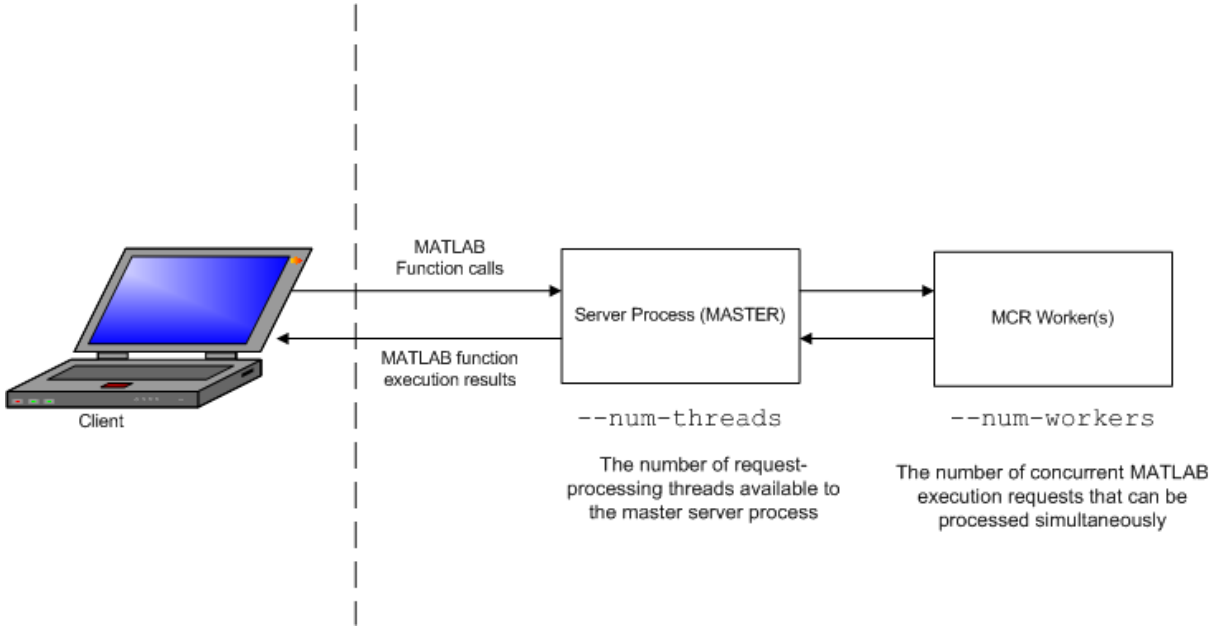
5. Results of MATLAB function execution are passed back for processing by the client.

The server is the middleman in the MATLAB Production Server environment. It simultaneously accepts connections from clients, and then dispatches MCR workers—MATLAB sessions—to process client requests to the MCR. By defining and adjusting the number of workers and threads available to a server, you tune capacity and throughput respectively.

- Workers (capacity management) (`--num-workers`) — The number of MCR workers available to a server.

  Each MCR worker dispatches one MATLAB execution request to the MCR, interacting with one client at a time. By defining and tuning the number of workers available to a server, you set the number of concurrent MATLAB execution requests that can be processed simultaneously. `--num-workers` should roughly correspond to the number of cores available on the local host.

- Threads (throughput management) (`--num-threads`) — The number of threads (units of processing) available to the master server process.



MATLAB® Production Server™ Data Flow from Client to Server and Back

The server does not allocate a unique thread to each client connection. Rather, when data is available on a connection, the required processing is scheduled on a *pool* of threads. `--num-threads` sets the size of that pool (the number of available request-processing threads) in the master server process. The threads in the pool do not execute MATLAB code directly. Instead, there is a single thread within each MCR worker process that executes MATLAB code on the client's behalf. The number of threads you define to a server should roughly correspond to the number of cores available on the local host.

Manage Licenses for MATLAB Production Server

Complete instructions for installing License Manager can be found in the *MATLAB Installation Guide*.

In addition to following instructions in the License Center to obtain and activate your license, do the following in order to set up and manage licensing for MATLAB Production Server:

Specify or Verify License Server Options in Server Configuration File

Specify or verify values for License Server options in the server configuration file (`main_config`). You create a server by using the `mps-new` command.

Edit the configuration file for the server. Open the file *`server_name`*`/config/main_config` and specify or verify parameter values for the following options. See the comments in the server configuration file for complete instructions and default values.

- `--license` — Configuration option to specify the license servers and/or the license files. You can specify multiple license servers including port numbers (*`port_number`*`@`*`license_server_name`*), as well as license files, with one entry in `main_config`. 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:

  1. `27000@hostA:` (`hostA` configured on port `27000`)
  2. `/opt/license/license.dat` (local license data file)
  3. `27001@hostB:` (`hostB` configured on port `27001`)
  4. `./license.dat` (local license data file)

- `--license-grace-period` — The maximum length of time MATLAB Production Server responds to HTTP requests, after license server

heartbeat has been lost. See FLEXlm® documentation for more on heartbeats and related license terminology.

- `--license-poll-interval` — 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 checkout a valid license. Polling occurs at the interval specified by `--license-poll-interval` until license has been successfully checked-out. See FLEXlm documentation for more on heartbeats and related license terminology.

Verify Status of License Server using mps-status

When you enter an `mps-status` command, the status of the server *and* the associated license is returned.

For detailed descriptions of these status messages, see "License Server Status Information".

Forcing a License Checkout Using mps-license-reset

Use the `mps-license-reset`command to force MATLAB Production Server to checkout a license. You can use this command at any time, providing you do not want to wait for MATLAB Production Server to verify and checkout a license at an interval established by a server configuration option such as `--license-grace-period` or `--license-poll-interval`.

Create a Server

In this section...

Prerequisites

Before creating a server, ensure you have:

- Installed MATLAB Production Server software.
- Added the `script` folder to your system `PATH` environment variable. Doing so enables you to run server commands such as `mps-new` from any folder on your system.

Note You can run server commands from the `script` folder. The script folder is located at *`$MPS_INSTALL`*`\script`, where *`$MPS_INSTALL`* is the location where MATLAB Production Server is installed. For example, on Windows, the default location is: `C:\Program Files\MATLAB\MATLAB Production Server\`*`ver`*`\script`. *`ver`* is the version of MATLAB Production Server.

Procedure

Before you can deploy your MATLAB code with MATLAB Production Server, you need to create a server to host your deployable archive.

A server instance is considered to be one unique *configuration* of the MATLAB Production Server product. Each configuration has its own parameter settings file (`main_config`) as well as its own set of diagnostic files.

To create a server configuration or *instance*, do the following:

1. From the system command prompt, navigate to where you want to create your server instance.

2. Enter the following command from the system prompt:

   ```
   mps-new [path/]server_name [-v]
   ```

   where:

   - *path* is the path to the server instance and configuration you want to create for use with the MATLAB Production Server product. When specifying a path, ensure the path ends with the *server_name*.

     If you are creating a server instance in the current folder, you do not need to specify a full path. Only specify the server name.

   - *server_name* — is the name of the server instance and configuration you want to create.

   - `-v` — enables verbose output, giving you information and status about each folder created in the server configuration.

   Upon successful completion of the command, MATLAB Production Server creates a new server instance.

Run mps-setup to Set Location of MATLAB Compiler Runtime (MCR)

Each server that you create with MATLAB Production Server has its own configuration file that defines various server management criteria.

The `mps-setup` command line wizard searches for MCR instances and sets the default path to the MATLAB Compiler Runtime (MCR) for all server instances you create with the product.

To run the command line wizard, do the following after first downloading and performing the “Download and Install the MATLAB Compiler Runtime (MCR)”:

1. Ensure you have logged on with `administrator` privileges.

2. At the system command prompt, run `mps-setup` from the `script` folder. Alternately, add the `script` folder to your system `PATH` environment variable to run `mps-setup` from any folder on your system. The `script`

folder is located at *$MPS_INSTALL*\script, where *$MPS_INSTALL* is the location in which MATLAB Production Server is installed. For example, on Windows, the default location is: `C:\Program Files\MATLAB\MATLAB Production Server\`*`ver`*`\script\mps-setup`.

ver is the version of MATLAB Production Server to use.

3. Follow the instructions in the command line wizard. The wizard will search your system and display installed MCR instances.

4. Enter `y` to confirm or `n` to specify a default MCR for all server configurations created with MATLAB Production Server.

   If `mps-setup` cannot locate an installed MCR on your system, you will be prompted to enter a path name to a valid instance.

Run mps-setup in Non-Interactive Mode for Silent Install

You can also run `mps-setup` without interactive command input for silent installations.

To run `mps-setup`, specify the path name of the MCR as a command line argument. For example, on Windows®:

```
mps-setup "C:\Program Files\MATLAB\MATLAB Compiler
Runtime\mcrver"
```

mcrver is the version of the MCR to use.

Edit the Configuration File

In this section...

About the Server Configuration File

To change any MATLAB Production Server parameters, edit the `main_config` configuration file that corresponds to your specific server instance:

`server_name``/config/main_config`

When editing `main_config`, remember these coding considerations:

- Each server has its own `main_config` configuration file.
- You enter only one configuration file parameter and related options per line. Each configuration file parameter starts with two dashes (`--`).
- Any line beginning with a pound sign (`#`) is ignored as a comment.
- Lines of white space are ignored.

Information about each configuration file parameter is included in the comments of each `main_config` file. The following are critical parameters to set or verify when running a server.

Specify the Installed MCR to the Server Instance

This example shows how to specify the installed location of the MATLAB Compiler Runtime (MCR) to your server instance. See "Download and Install the MATLAB Compiler Runtime (MCR)" for details about how to install the MCR.

1. Navigate to `/tmp/prod_server_1`.

2. Open the folder labeled `prod_server_1`.

3. Open the folder labeled `config`.

4. Open `main_config` with a text editor of your choice.

   Find the configuration file option `--mcr-root` in `main_config`. By default, in a new server instance, the value of `--mcr-root` is:

   ```
   m C R r O O T u N s E T
   ```

5. Modify the `--mcr-root` option default value to point to the installed MCR you want to work with. For example:

   ```
   --mcr-root C:\Program Files\MATLAB\MATLAB Compiler Runtime\vnnn
   ```

   Note You *must* specify the version number of the MCR (v*nnn*) in `--mcr-root`. MCR versions you specify must be compatible with MATLAB Production Server.

6. Save `main_config` and exit.

7. Run `mps-restart` to stop the server instance and start it with your specified options. To restart `prod_server_1` from a system command prompt, enter the following:

   ```
   mps-restart -C /tmp/prod_server_1
   ```

Common Customizations

Setting Default Port Number for Client Requests

Use the `--http` parameter to set the default port number on which the server listens for client requests.

Setting Number of Available Workers

Use the `--num-workers` parameter to set the number of concurrent MATLAB execution requests that can be processed simultaneously.

See "Server Overview" on page 1-2 for more information.

Setting Number of Available Threads

Use the `--num-threads` parameter to set the number of request-processing threads available to the master server process.

See "Server Overview" on page 1-2 for more information.

Note For .NET Clients, the HTTP 1.1 protocol restricts the maximum number of concurrent connections between a client and a server to two.

This restriction only applies when the client and server are connected remotely. A local client/server connection has no such restriction.

To specify a higher number of connections than two for remote connection, use the NET classes `System.Net.ServicePoint` and `System.Net.ServicePointManager` to modify maximum concurrent connections.

For example, to specify four concurrent connections, code the following:

```
ServicePointManager.DefaultConnectionLimit = 4;
MWClient client = new MWHttpClient(new MyConfig());
MPSClient mpsExample = client.CreateProxy(
            new Uri("http://user01:9910/mpsexample"));
```

For More Information

For information about...	See...
Downloading and installing the MCR	"Download and Install the MATLAB Compiler Runtime (MCR)"
Product installation	"Install MATLAB Production Server"

Secure a Server

In this section...

Overview

MATLAB Production Server uses HTTPS to establish secure connections between server instances and clients. The HTTPS layer provides certificate-based authentication for both clients and server instances. It also provides an encrypted data path between the clients and server instances. You can configure the level of security provided by the HTTPS layer and the security protocols it supports.

MATLAB Production Server provides a certificate-based authorization mechanism for restricting access to specific programs. Using this mechanism, you specify the MATLAB programs that a client can access.

Enable Security

To enable security, add the following to the server instance's configuration:

- HTTPS port
- Valid certificate stored in a PEM formatted certificate chain
- Valid private key stored in PEM format

The following configuration excerpt configures a server instance to accept secure connections on port 9920, use the certificate stored in

`./x509/my-cert.pem`, and use the unencrypted private key stored in `./x509/my-key.pem`.

```
...
--https 9920
--x509-cert-chain ./x509/my-cert.pem
--x509-private-key ./x509/my-key.pem
...
```

The default security settings allow all clients to access all programs hosted by the server instance. The server instance does not authenticate the clients, nor does it perform any authorization. The default settings enable all security protocols and enable all cipher suites, save for eNULL.

In production settings that require greater security than that provided by an unencrypted private key, use an encrypted private key. You specify the passphrase for decrypting the private key in a file with owner-read-only access, and use the `--x509-passphrase` property to tell the server instance about it.

```
...
--https 9920
--x509-cert-chain ./x509/my-cert.pem
--x509-private-key ./x509/my-key.pem
--x509-passphrase ./x509/my-passphrase
...
```

Configure Client Authentication

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

1. Set the `--ssl-verify-peer-mode` configuration property to `verify-peer-require-peer-cert`.

2. 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.

---

3 Optionally configure the server instance to respect any certificate revocation lists (CRLs) in the CA store.

Specify this behavior by adding the `--x509-use-crl` property to the server's configuration. If this property is not specified, the server instance ignores the CRLs and potentially authenticate clients using revoked credentials.

Caution You must add a CRL list to the server's CA store before adding the `-x509-use-crl` property. If the the CA store does not include a CRL list, the server will crash.

This configuration excerpt configures a server instance to authenticate clients using the system CA store and to respect CRLs:

```
...
--https 9920
--x509-cert-chain ./x509/my-cert.pem
--x509-private-key ./x509/my-key.pem
--x509-passphrase ./x509/my-passphrase
--ssl-verify-peer-mode verify-peer-require-cert
--x509-use-system-store
--x509-use-crl
...
```

Specify Access to MATLAB Programs

By default, server instances allow all clients to access all hosted MATLAB programs. You control this behavior using the `--ssl-allowed-client` configuration property. The `--ssl-allowed-client` property specifies a

comma separated list of clients, identified by their certificate's common name, that are allowed to access MATLAB programs. You also use the property to list specific MATLAB programs that a client is allowed to access.

If you do not specify the `--ssl-allowed-client` property, the server instance does not restrict access to the hosted MATLAB programs. After you add an entry for the `--ssl-allowed-client` property, the server instance authorizes only the listed clients to access the hosted MATLAB programs.

To only authorize clients with the common names `jim`, `judy`, and `ash` to use the MATLAB programs hosted on a server instance, add this configuration excerpt:

```
--ssl-allowed-client jim,judy,ash
```

You can restrict access further by only authorizing specific clients to have access to specific MATLAB programs. Do this by adding `:`*`allowedPrograms`* to the value of the `--ssl-allowed-client` property. *`allowedPrograms`* is a comma separated list of program names.

To allow clients with the common name `jim` access to all hosted programs, allow clients with the common name `judy` access to the programs `tail` and `zap`, and allow clients with the common name `ash` or `joe` access to the programs `saw` and `travel`, add this configuration excerpt:

```
--ssl-allowed-client jim
--ssl-allowed-client judy:tail,zap
--ssl-allowed-client ash,joe:saw,travel
```

Adjust Security Protocols

By default, MATLAB Production Server instances will allow connections using SSLv2, SSLv3, and TLSv1. You control the enabled protocols using the `--ssl-protocols` property. This property specifies the protocols the server instance can use.

To enable only SSLv3, add this configuration excerpt:

```
--ssl-protocols SSLv3
```

Because SSLv2 and TLSv1 are not included in the list, the server instance does not enable the protocols.

You can restrict the cipher suites used by the server instance with the `--ssl-ciphers` property. After you add the property to a server instance's configuration, the server instance will use only the listed cipher suites.

To enable only high strength cipher suites, you add this configuration excerpt:

```
--ssl-ciphers HIGH
```

Improve Startup Time When Security Is Activated

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 a server instance's startup time.

If you need the server instance to startup 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 using the `--ssl-tmp-dh-param` configuration property. The `--ssl-tmp-dh-param` property specifies the file storing the DH key in PEM format.

Security Configuration Properties

Property	Description	Default
`--https`	Port number used for the secure connection.	
`--x509-private-key`	Path to the private key used to load the server's certificates. The private key must be in PEM format.	
`--x509-passphrase`	Path to the file containing the passphrase used to encrypt the private key. This file must be owner read only.	

Property	Description	Default
`--x509-cert-chain`	Path to the server's certificate chain file. This file contains the server's certificate and any untrusted certificates. This property requires that `--x509-private-key` also be set.	
`--ssl-verify-peer-mode`	Specifies if the server requires the client to provide a certificate for authentication. Valid values are: • `no-verify-peer` • `verify-peer-require-peer-cert`	`no-verify-peer`
`--x509-ca-file-store`	Path to the server's CA store. This file contains trusted certificates that are used to verify client authenticity.	
`--x509-use-system-store`	Specifies that the server should use the host system's CA store. If this property is not set, the server uses the file specified by `--x509-ca-file-store`.	Do not use the system's CA store.
`--x509-use-crl`	Specifies if the server uses the certificate revocation list during client authentication.	Do not use the certificate revocation list.
`--ssl-allowed-client`	List of clients and the programs they can access. The format is `client1,client2,...:program1,program2` where the clients in the list can access the programs in the list. If no programs are specified the listed clients can access all hosted programs.	Allow all clients to access all programs.

Property	Description	Default
`--ssl-protocols`	List of allowed protocols. Supported protocols include: • SSLv2 • SSLv3 • TLSv1	Allow all protocols.
`--ssl-ciphers`	List of enabled cipher suites. Special values include: • `ALL` — Enable all cipher suites except for the eNULL ciphers. • `HIGH` — Enable all high encryption cipher suites.	`ALL`
`--ssl-tmp-dh-param`	Path to a pre-generated ephemeral DH key.	Server auto-generates an ephemeral DH key.

Start a Server

In this section...

Prerequisites

Before attempting to start a server, verify that you have:

- Installed the MATLAB Compiler Runtime (MCR)
- Created a server
- Customized the server configuration file, `main_config` with the location of the MCR or Run mps-setup to set the location of the MATLAB Compiler Runtime (MCR)

Procedure

To start a server, complete the following steps:

1. Open a system command prompt.
2. Enter the following command:

   ```
   mps-start [-C path/]server_name [-f]
   ```

   where:

   - `-C` *`path/`* — Path to the server instance you want to create. *`path`* should end with the server name.
   - *`server_name`* — Name of the server instance you want to start or stop.
   - `-f` — Forces command to succeed, regardless or whether the server is already started or stopped.

   Upon successful completion of the command, the server instance is active.

Note If needed, query the status of the server instance that you started to verify the server is running.

For More Information

For information about...	See...
Downloading and installing the MCR	"Download and Install the MATLAB Compiler Runtime (MCR)"
Solving errors when starting or stopping a server	"Server Troubleshooting" on page 1-27
The `mps-start` command	`mps-start`
Stopping the server with the `mps-stop` command	`mps-stop`
Verifying status of a server with the `mps-status` command	`mps-status`
Product installation	"Install MATLAB Production Server"

Share the Deployable Archive

After you create the deployable archive, share it with clients of MATLAB Production Server by copying it to your server, for hosting.

In order to share the deployable archive, a server must be created and started.

1. Locate your deployable archive in the `for_redistribution_files_only` folder of your compiler project folder.

   It will be named *`project_name`*`.ctf`.

2. Copy *`project_name`*`.ctf` to the `\`*`server_name`*`\auto_deploy` folder in your server instance.

   For example, if your server is named `prod_server_1` and located in `C:\tmp`, copy *`project_name`*`.ctf` to `C:\tmp\prod_server_1\auto_deploy`.

Verify Server Status

In this section...

Procedure

To verify the status of a server instance, complete the following steps:

1. Open a system command prompt.
2. Enter the following command:

   ```
   mps-status [-C path/]server_name
   ```

   where:

   - `-C path/` — Path to the server instance and configuration you want to create. *`path`* should end with the server name.
   - *`server_name`* — Name of the server instance and configuration you want to start or stop.

   Upon successful completion of the command, the server status displays.

License Server Status Information

In addition to the status of the server, `mps-status` also displays the status of the license server associated with the server you are verifying.

Possible statuses and their meanings follow:

This License Server Status Message...	Means...
`License checked out`	The server is operating with a valid license. The server is communicating with the License Manager, and the proper number of license keys are checked out..
`WARNING: lost connection to license server - request processing will be disabled at` *`time`* `unless connection to license server is restored`	The server has lost communication with the License Manager, but the server is still fully operational and will remain operational until the specified *`time`*. At *`time`*, if connectivity to the license server has not been restored, request processing will be disabled until licensing is reestablished.
`ERROR: lost connection to license server - request processing disabled`	The server has lost communication with the License Manager for a period of time exceeding the grace period. Request processing has been suspended, but the server actively attempts to reestablish communication with the License Manager until it succeeds, at which time normal request processing resumes. For information about grace periods, see “Specify or Verify License Server Options in Server Configuration File” on page 1-5.

Verify Status of a Server

This example shows how to verify the status of the server instance you started in the previous example.

In this example, you verify the status of `prod_server_1`, from a location other than the server instance folder (`C:\tmp\prod_server_1`).

1. Open a system command prompt.
2. To verify `prod_server_1` is running, enter this command:

   ```
   mps-status -C \tmp\prod_server_1
   ```

   If `prod_server_1` is running, the following status is displayed:

   ```
   \tmp\prod_server_1 STARTED
   license checked out
   ```

This output confirms `prod_server_1` is running and the server is operating with a valid license.

For more information on the `STOPPED` status, see `mps-stop` and `mps-restart`.

For more information about license status messages, see "License Server Status Information" on page 1-24.

Server Troubleshooting

In this section...

Procedure

To diagnose a problem with a server instance or configuration of MATLAB Production Server, do the following, as needed:

- Check the logs for warnings, errors, or other informational messages.
- Check Process Identification Files (PID files) for information relating to problems with MCR worker processes.
- Check Endpoint Files for information relating to problems relating to the server's bound external interfaces — for example, a problem connecting a client to a server.
- Use server diagnostic tools, such as `mps-which`, as needed.

Diagnose a Server Problem

This example shows a typical diagnostic procedure you might follow to solve a problem starting server `prod_server_x`.

After you issue the command:

```
mps-start prod_server_x
```

from within the server instance folder (`prod_server_x`), you get the following error:

```
Server process exited with return code: 4
(check logs for more information)
Error while waiting for server to start: The I/O operation
has been aborted because of either a thread exit
```

```
or an application request
```

To solve this issue, you might check the `log` files for more detailed messages, as follows:

1. Navigate to the server instance folder (`prod_server_x`) and open the `log` folder.
2. Open `main.err` with any text editor. Note the following message listed under `Server startup error:`

   ```
   Dynamic exception type: class std::runtime_error
   std::exception::what: bad MCR installation:
   C:\Program Files\MATLAB\MATLAB Compiler Runtime\v717
   (C:\Program Files\MATLAB\MATLAB Compiler Runtime\v717\bin\
   win64\mps_worker_app could not be found)
   ```

3. The message indicates the installation of the MATLAB Compiler Runtime (MCR) is incomplete or has been corrupted. To solve this problem, reinstall the MCR.

Server Diagnostic Tools

Each server instance contains three sets of diagnostic files to help you determine and solve problems with the server and associated processes

Log Files

Each server writes a log file containing data from both the main server process, as well as the workers, named *`server_name`*`/log/main.log`. You can change the primary log folder name from the default value (`log`) by setting the option `--log-root` in `main_config`.

The primary log folder contains the `main.log` file, as well as a symbolic link to this file with the auto-generated name of `main_`*`date_fileID`*`.log`.

The `stdout` stream of the main server process is captured as `log/main.out`.

The `stderr` stream of the main server process is captured as `log/main.err`.

Log Retention and Archive Settings. Log data is written to the server's `main.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, located in the archive log folder specified by `--log-archive-root`.

You can set parameters that define when `main.log` is archived using the following options in each server's `main_config` file.

- `--log-rotation-size` — When `main.log` reaches this size, the active log is written to an archive log (located in the folder specified by `--log-archive-root`).
- `--log-archive-max-size` — When the combined size of all files in the archive folder (location defined by `--log-archive-root`) reaches this limit, archive logs are purged until the combined size of all files in the archive folder is less than `--log-archive-max-size`. Oldest archive logs are deleted first.

Specify values for these options using the following units and notations:

Represent these units of measure...	Using this notation...	Example
Byte	`b`	`900b`
Kilobyte (1024 bytes)	`k`	`700k`
Megabytes (1024 kilobytes)	`m`	`40m`
Gigabytes (1024 megabytes)	`g`	`10g`
Terabytes (1024 gigabytes)	`t`	`2t`
Petabytes (1024 terabytes)	`p`	`1p`

Note The minimum value you can specify for `--log-rotation-size` is 1 megabyte.

On Windows 32-bit systems, values larger than 2^{32} bytes are not supported. For example, specifying `5g` is not valid on Windows 32-bit systems.

Best Practices for Log Management. Use these recommendations as a guide when defining values for the options listed in "Log Retention and Archive Settings" on page 1-29.

- Avoid placing `--log-root` and `--log-archive-root` on different physical file systems.
- Place log files on local drives, not on network drives.
- Send MATLAB output to `stdout`. Develop an appropriate, consistent logging strategy following best MATLAB coding practices. See *MATLAB Programming Fundamentals* for guidelines.

Setting Log File Detail Levels. Each log level provides different levels of information for troubleshooting. For complete information on all logging levels and what details they provide, see the comments in the `main_config` file. Before you call support, you should set logging levels to `trace`.

Process Identification Files (PID Files)

Each process that the server runs generates a *Process Identification File (PID File)* in the folder identified as `pid-root` in `main_config`.

The main server PID file is `main.pid`; for each MCR Worker process, it is `worker-`*`n`*`.pid`, where *n* is the unique identifier of the worker.

PID files are automatically deleted when a process exits.

Endpoint Files

Endpoint files are generated to capture information about the server's bound external interfaces. The files are created when you start a server instance and deleted when you stop it.

server_name/endpoint/http contains the IP address and port of the clients connecting to the server. This information can be useful in the event that zero (0) is specified in main_config, indicating that the server bind to a free port.

Common Error Messages and Resolutions

This section lists common troubleshooting scenarios, including error messages and typical resolutions:

(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 MCR Instance

Common causes of this message include:

- You are not properly qualifying the path to the MCR. You must include the version number. For example, you need to specify:

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

  not

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

Error: Server Instance not Specified

MATLAB Production Server can't find the server you are specifying.

Ensure you are either entering commands from the folder containing the server instance, or are using the -C command argument to specify a precise location of the server instance.

For example, if you created server_1 in C:\tmp\server_1, you would issue the mps-start command from within that folder to avoid specifying a path with the -C argument:

```
cd c:\tmp\server_1
mps-start server_1
```

For more information, see “Start a Server” on page 1-21.

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 Files” on page 1-28 for further details on the precise cause of the problem.

2

Commands — Alphabetical List

mps-check

Purpose

Tests and diagnoses MATLAB Production Server instance for problems.

Syntax

`mps-check` [--timeout *`seconds`*] *`host`*:*`port`*

Description

`mps-check` sends a request to a MATLAB Production Server instance and receives a status report that is used to identify issues that cause the product to run less than optimally.

Information reported by `mps-check` to `stdout` include:

- Status of the server instance
- Port the HTTP interface is listening on
- Deployed archives for a server instance

Before using `mps-check`, you must deploy *`mcrroot`*`/bin/`*`arch`*`/mps_check.ctf` to the server instance.

- *`mcrroot`* is the path to the MCR's installation folder.
- *`arch`* is standard abbreviation for the system's operating system and hardware architecture.

Input Arguments

- `--timeout` *`seconds`* — The time, in seconds, to wait for a response from the server before timing out. The default is two minutes.
- *`host`* — The host name of the machine running the server instance.
- *`port`* — The port number on which the server instance listens for requests.

Definitions

Server Instance

An instance of the MATLAB Production Server. The files contained in the folder created by `mps-new`, defined by *`path`*`/`, comprise one configuration of the MATLAB Production Server product.

Examples

Display diagnostic information for the server instance running on port 9910 of the local computer.

```
mps-check localhost:9910

Connecting to localhost:9910
Connected
Sending HTTP request
HTTP request sent
HTTP response received
MPS status check completed successfully
```

mps-license-reset

Purpose

Forces server instance to immediately attempt license checkout

Syntax

```
mps-license-reset [-C path/]server_name
```

Description

`mps-license-reset [-C path/]server_name` triggers the server to checkout a license immediately, regardless of the current license status. License keys that are currently checked out are checked in first.

Input Arguments

`-C path/`

Specify a path to the server instance. If this option is omitted, the current working folder and its parents are searched to find the server instance.

`server_name`

Server checking out license

Examples

Create a new server instance and display the status of each folder in the file hierarchy, as the server instance is created:

```
mps-license-reset -C /tmp/server_2
```

See Also

`mps-status`

Related Examples

- “Forcing a License Checkout Using mps-license-reset” on page 1-6

Concepts

- “Manage Licenses for MATLAB® Production Server™” on page 1-5

Purpose

Create server instance

Syntax

```
mps-new [path/]server_name [-v]
```

Description

`mps-new [path/]server_name [-v]` makes a new folder at *`path`* and populates it with the default folder hierarchy for a server instance

Each server instance can be configured, started, monitored, and stopped independently.

Tips

- Before creating a server instance, ensure that no file or folder with the specified *`path`* currently exists on your system.
- After issuing `mps-new`, you must issue `mps-start` to start the server instance.

Input Arguments

`path/`

Path to server instance.

`server_name`

Name of the server to be created.

If you are creating a server instance in the current working folder, you do not need to specify a full path. Only specify the server name.

`-v`

Displays status of each folder in the file hierarchy, created to form a server instance

Examples

Create a new server instance and display the status of each folder in the file hierarchy, as the server instance is created:

```
mps-new /tmp/server_1 -v
```

Example Output

```
server_1/.mps-version...ok
```

```
server_1/config/...ok
server_1/config/main_config...ok
server_1/endpoint/...ok
server_1/auto_deploy/...ok
server_1/.mps-socket/...ok
server_1/log/...ok
server_1/pid/...ok
```

See Also

`mps-status | mps-start`

Related Examples

- "Create a Server" on page 1-7

Concepts

- "Server Overview" on page 1-2

Purpose

Stop and start server instance

Syntax

```
mps-restart [-C path/]server_name [-f]
```

Description

`mps-restart [-C path/]server_name [-f]` stops a server instance, then restarts the same server instance. Issuing `mps-restart` is equivalent to issuing the `mps-stop` and `mps-start` commands in succession.

Tips

- After issuing `mps-restart`, issue the `mps-status` command to verify the server instance has started.
- If you are restarting a server instance in the current working folder, you do not need to specify a full path. Only specify the server name.

Input Arguments

`-C path/`

Specify a path to the server instance. If this option is omitted, the current working folder and its parents are searched to find the server instance. If you are restarting a server instance in the current working folder, you do not need to specify a full path. Only specify the server name.

`server_name`

Name of the server to be restarted.

`-f`

Force success even if the server instance is stopped. Restarting a stopped instance returns an error.

Examples

Restart a server instance named `server_1`, located in folder `tmp`. Force successful completion of `mps-restart`.

```
mps-restart -f -C /tmp/server_1
```

See Also

`mps-start` | `mps-stop` | `mps-status`

mps-setup

Purpose

Sets up server environment

Syntax

```
mps-setup [mcrroot]
```

Description

`mps-setup [mcrroot]` sets location of MATLAB Compiler Runtime (MCR) and other start-up options.

The `mps-setup` command sets the default path to the MATLAB Compiler Runtime (MCR) for all server instances you create with the product. This is equivalent to presetting the `--mcr-root` option in each server's `main_config` configuration file.

If a default value already exists in *`server_name`*`/config/mcrroot`, it is updated with the value specified when you run the command line wizard.

Tips

- Run `mps-setup` from the script folder. Alternatively, add the `script` folder to your system `PATH` environment variable to run `mps-setup` from any folder on your system.
- Run `mps-setup` without arguments and it will search your system for MCR instances you may want to use with MATLAB Production Server.
- Run `mps-setup` by passing the path to the MATLAB Compiler Runtime (MCR) as an argument. This method is ideal for non-interactive (silent) installations.

Input Arguments

mcrroot

Specify a path to the MATLAB Compiler Runtime if running `mps-setup` in non-interactive, or silent, mode.

Examples

Run `mps-setup` non-interactively, by passing in a path to the MATLAB Compiler Runtime (MCR) instance that you want MATLAB Production Server to use.

```
mps-setup "C:\Program Files\MATLAB\MATLAB
Compiler Runtime\mcrver"
```

mcrver is the version of the MCR to use.

See Also

`mps-start` | `mps-new` | `mps-status`

mps-start

Purpose

Starts server instance

Syntax

```
mps-start [-C path/]server_name [-f]
```

Description

`mps-start [-C path/]server_name [-f]` starts a server instance

Tips

- After issuing `mps-start`, issue the `mps-status` command to verify the server instance has `STARTED`.
- If you are starting a server instance in the current working folder, you do not need to specify a full path. Only specify the server name.

Input Arguments

`-C path/`

Specify a path to the server instance. If this option is omitted, the current working folder and its parents are searched to find the server instance.

`server_name`

Name of the server to be started.

`-f`

Force success even if the server instance is currently running. Starting a running server instance is considered an error.

Examples

Start a server instance named `server_1`, located in folder `tmp`. Force successful completion of `mps-start`.

```
mps-start -f -C /tmp/server_1
```

See Also

`mps-stop` | `mps-restart` | `mps-status`

Concepts

- “Start a Server” on page 1-21
- “Server Overview” on page 1-2

Purpose

Displays status of server instance

Syntax

```
mps-status [-C path/]server_name
```

Description

`mps-status [-C path/]server_name` displays the status of the server (`STARTED`, `STOPPED`), along with a full path to the server instance.

Tips

- If you are creating a server instance in the current working folder, you do not need to specify a full path. Only specify the server name.
- If the server is running, the status of the license associated with that server will also be displayed.

Input Arguments

`-C path/`

Specify a path to the server instance. If this option is omitted, the current working folder and its parents are searched to find the server instance.

`server_name`

Server to be queried for status

Examples

Display status of server instance `server_1`, residing in `tmp` folder.

```
mps-status -C /tmp/server_1
```

Example Output

If server is running and running with a valid license:

```
'/tmp/server_1' STARTED
license checked out
```

If server is not running:

```
'/tmp/server_1' STOPPED
```

See Also

`mps-start` | `mps-stop` | `mps-restart` | `mps-which`

Concepts

- "Start a Server" on page 1-21
- "Server Overview" on page 1-2
- "License Server Status Information" on page 1-24

Purpose

Stop server instance

Syntax

```
mps-stop [-C path/]server_name [-f] [-v]
[--timeout hh:mm:ss]
```

Description

`mps-stop [-C path/]server_name [-f] [-v] [--timeout hh:mm:ss]` closes HTTP server socket and all open client connections immediately. All function requests that were executing when the command was issued are allowed to complete before the server shuts down.

Tips

- After issuing `mps-stop`, issue the `mps-status` command to verify the server instance has `STOPPED`.
- If you are stopping a server instance in the current working folder, you do not need to specify a full path. Only specify the server name.
- Note that the timeout option (`--timeout hh:mm:ss`) is specified with two (2) dashes, not one dash.

Input Arguments

`-C path/`

Specify a path to the server instance. If this option is omitted, the current working folder and its parents are searched to find the server instance.

`server_name`

Name of the server to be stopped.

`-f`

Force success even if the server instance is not currently stopped. Stopping a stopped instance is considered an error.

`-v`

Displays system messages relating to termination of server instance.

`--timeout hh:mm:ss`

Set a limit on how long `mps-stop` will run before returning either success or failure. For example, specifying `--timeout 00:02:00` indicates that `mps-stop` should exit with an error status if the server takes longer than two (2) minutes to shut down. The instance continues to attempt to terminate even if `mps-stop` times out. If this option is not specified, the default behavior is to wait as long as necessary (infinity) for the instance to stop.

Examples

Stop server instance `server_1`, located in `tmp` folder. Force successful completion of `mps-stop`. Timeout with an error status if `mps-stop` takes longer than three (3) minutes to complete.

In this example, the verbose (`-v`) option is specified, which produces an output status message.

```
mps-stop -f -v -C /tmp/server_1 --timeout 00:03:00
```

Example Output

```
waiting for stop... (timeout = 00:03:00)
```

See Also

`mps-start` | `mps-restart` | `mps-new` | `mps-status`

Purpose

Displays licensing and configuration information of a MATLAB Production Server instance

Syntax

`mps-support-info` [-C [[*instance_path*] | [*server_name*]]]

Description

`mps-support-info` displays licensing and configuration information of a MATLAB Production Server instance.

Input Arguments

- `-C` *`instance_path`* — The path to where the server instance is installed.
- `-C` *`server_name`* — The name of the server instance to locate in the current folder.

Examples

Display licensing and configuration information of server instance `fred`, residing in `/` folder.

```
mps-support-info -C /fred
```

```
Instance Version:           1.0
License Number:             UNKNOWN -- MPS stopped
MPS Version:                UNKNOWN -- MPS stopped
Available License Number:   857812
Client Version:             1.0.1 R2013a
Operating System:           Microsoft Windows 7 Enterprise Edition (build 7601), 64-bit
Number of CPU cores:        8
CPU Info:                   Intel(R) Xeon(R) CPU           W3550  @ 3.07GHz 64-bit Compatible
Memory:                     11.9915 GB ( 1.2574e+007 KB )
```

mps-which

Purpose

Display path to server instance that is currently using the configured port.

Syntax

```
mps-which [-C path/]server_name
```

Description

`mps-which [-C path/]server_name` is useful when running multiple server instances on the same machine. If you accidently leaves a server instance running and try to start another which is configured to use the same port number, the latter server instance will fail to start, displaying an `address-in-use` error. `mps-which` can be used to identify which server instance is using the port.

Tips

- If you are creating a server instance in the current working folder, you do not need to specify a full path. Only specify the server name.

Input Arguments

`-C path/`

Specify a path to the server instance. If this option is omitted, the current working folder and its parents are searched to find the server instance.

`server_name`

Server to be queried for path.

Examples

`server_1` and `server_2`, both residing in folder `tmp`, are configured to use to same port, defined by `--http` in the `main_config` configuration files. However, the port can only be allocated to one server.

Run `mps-which` for both servers:

```
mps-which -C /tmp/server_1

mps-which -C /tmp/server_2
```

Example Output

In both cases, the server that has allocated the configured port displays (`server_1`):

```
/tmp/server_1
```

See Also

mps-status