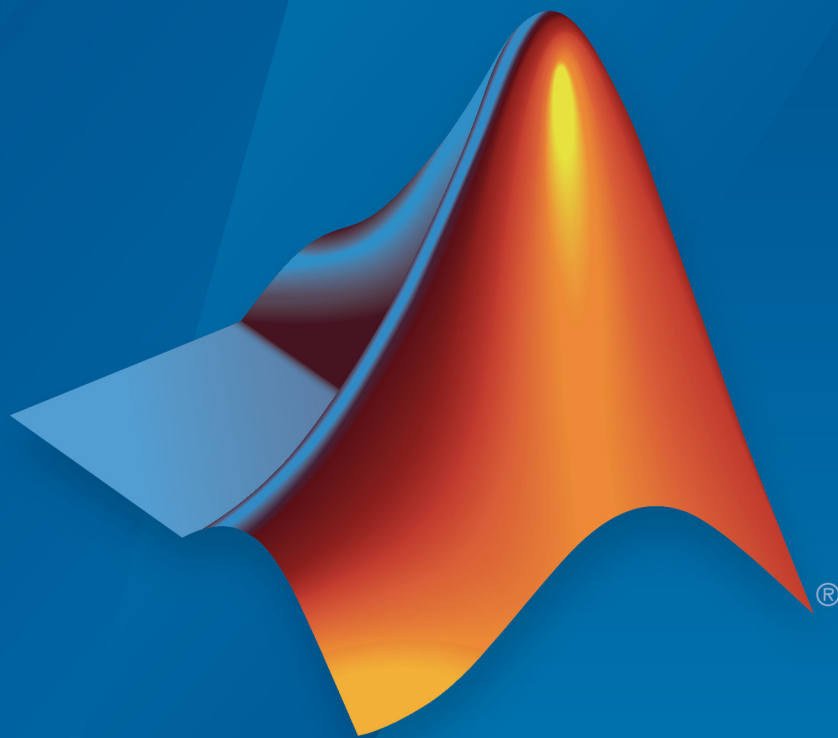


MATLAB® Parallel Server™

System Administrator's Guide



MATLAB®

R2019a



How to Contact MathWorks



Latest news: www.mathworks.com
Sales and services: www.mathworks.com/sales_and_services
User community: www.mathworks.com/matlabcentral
Technical support: www.mathworks.com/support/contact_us



Phone: 508-647-7000



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

MATLAB® Parallel Server™ System Administrator's Guide

© COPYRIGHT 2005–2019 by The MathWorks, Inc.

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

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

Trademarks

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

Patents

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

Revision History

November 2005	Online only	New for Version 2.0 (Release 14SP3+)
December 2005	Online only	Revised for Version 2.0 (Release 14SP3+)
March 2006	Online only	Revised for Version 2.0.1 (Release 2006a)
September 2006	Online only	Revised for Version 3.0 (Release 2006b)
March 2007	Online only	Revised for Version 3.1 (Release 2007a)
September 2007	Online only	Revised for Version 3.2 (Release 2007b)
March 2008	Online only	Revised for Version 3.3 (Release 2008a)
October 2008	Online only	Revised for Version 4.0 (Release 2008b)
March 2009	Online only	Revised for Version 4.1 (Release 2009a)
September 2009	Online only	Revised for Version 4.2 (Release 2009b)
March 2010	Online only	Revised for Version 4.3 (Release 2010a)
September 2010	Online only	Revised for Version 5.0 (Release 2010b)
April 2011	Online only	Revised for Version 5.1 (Release 2011a)
September 2011	Online only	Revised for Version 5.2 (Release 2011b)
March 2012	Online only	Revised for Version 6.0 (Release 2012a)
September 2012	Online only	Revised for Version 6.1 (Release 2012b)
March 2013	Online only	Revised for Version 6.2 (Release 2013a)
September 2013	Online only	Revised for Version 6.3 (Release 2013b)
March 2014	Online only	Revised for Version 6.4 (Release 2014a)
October 2014	Online only	Revised for Version 6.5 (Release 2014b)
March 2015	Online only	Revised for Version 6.6 (Release 2015a)
September 2015	Online only	Revised for Version 6.7 (Release 2015b)
March 2016	Online only	Revised for Version 6.8 (Release 2016a)
September 2016	Online only	Revised for Version 6.9 (Release 2016b)
March 2017	Online only	Revised for Version 6.10 (Release 2017a)
September 2017	Online only	Revised for Version 6.11 (Release 2017b)
March 2018	Online only	Revised for Version 6.12 (Release 2018a)
September 2018	Online only	Revised for Version 6.13 (Release 2018b)
March 2019	Online only	Revised for Version 7.0 (Release 2019a)

Introduction

1

MATLAB Parallel Server Product Description	1-2
Product Overview	1-3
Parallel Computing Concepts	1-3
Determining Product Installation and Versions	1-4
Toolbox and Server Components	1-5
Schedulers, Workers, and Clients	1-5
Third-Party Schedulers	1-7
Components on Mixed Platforms or Heterogeneous Clusters	1-8
mjs Service	1-8
Using Parallel Computing Toolbox Software	1-9

Network Administration

2

Requirements and Ports for MATLAB Parallel Server	2-2
Port Configuration	2-2
Fully Qualified Domain Names	2-2
Security Considerations	2-2
Use Different MPI Builds on UNIX Systems	2-5
Build MPI	2-5
Use Your MPI Build	2-5
Shut Down a Job Manager Cluster	2-8
Linux and Macintosh Operating Systems	2-8

Microsoft Windows Operating Systems	2-9
Customize Startup Parameters	2-11
Define Script Defaults	2-11
Override Script Defaults	2-13
Access Service Record Files	2-15
Locate Log Files	2-15
Locate Checkpoint Folders	2-15
Set MATLAB Job Scheduler Cluster Security	2-17
Set the Security Level	2-17
Local, MATLAB Job Scheduler, and Network Passwords	2-19
Set Secure Communication	2-20
Troubleshoot Common Problems	2-22
License Errors	2-22
Memory Errors on UNIX Operating Systems	2-24
Run Server Processes on Windows Network Installation	2-24
Required Ports	2-24
Ephemeral TCP Ports with Job Manager	2-25
Host Communications Problems	2-25
Verify Network Communications for Cluster Discovery	2-26

Product Installation

3

Integrate MATLAB with Third-Party Schedulers	3-2
Activate Your MATLAB Parallel Server License	3-2
Get the Installation Files	3-2
Install License Manager on the Head Node	3-3
Install Software on Worker Nodes	3-4
Install Software on Local Desktop	3-4
Configure Your Cluster	3-5
Integrate MATLAB Job Scheduler for Network License Manager	3-6
Activate Your MATLAB Parallel Server License	3-6
Get the Installation Files	3-6
Install Software on the Head Node	3-7

Install Software on Worker Nodes	3-8
Configure the MATLAB Job Scheduler with Admin Center	3-8
Connect the MATLAB Client to the MATLAB Parallel Server Cluster	3-11
Integrate MATLAB Job Scheduler for Online Licensing	3-13
Set Up License and Users	3-13
Get the Installation Files	3-14
Install Software on All Nodes	3-15
Configure the MATLAB Job Scheduler with Admin Center	3-16
Connect the MATLAB Client to the MATLAB Parallel Server Cluster	3-18
Configure Advanced Options for MATLAB Job Scheduler	
Integration	3-19
Run Multiple MATLAB Parallel Server Versions	3-19
Set Up Windows Cluster Hosts	3-20
Configure Firewalls on Server	3-21
Stop mjs Services of Old Installation	3-22
Set the MATLAB Job Scheduler Security Level	3-23
Start the mjs Service, MATLAB Job Scheduler, and Workers (Command-Line)	3-23
Install the mjs Service to Start Automatically at Boot Time (UNIX)	3-28
Validate Installation with MATLAB Job Scheduler	3-30
Configure for HPC Pack	3-31
Configure Cluster for Microsoft HPC Pack	3-31
Configure Client Computer for HPC Pack	3-32
Validate Installation Using Microsoft HPC Pack	3-33
Configure for Slurm, PBS Pro, Platform LSF, TORQUE	3-37
Configure Platform LSF Scheduler on Windows Cluster	3-37
Configure Windows Firewalls on Client	3-39
Validate Installation Using a Slurm, LSF, PBS Pro, or TORQUE Scheduler	3-39
Configure Using the Generic Scheduler Interface	3-44
Interfacing with Generic Schedulers	3-44
Creating a Generic Cluster Profile	3-45
Special Configurations	3-52

Distribute a Generic Cluster Profile and Integration Scripts	3-54
.....	
Decide How Users Access the Integration Scripts	3-54
Shared IntegrationScriptsLocation Folder	3-54
Distribute Copies of the IntegrationScriptsLocation Folder ..	3-55
Further Considerations for Clusters with Shared File Systems	
.....	3-56
Configure a Hadoop Cluster	3-57
Cluster Configuration	3-57
Client Configuration	3-57
Kerberos Authentication	3-58
Hadoop Version Support	3-59

Admin Center

4

Start Admin Center	4-2
Set Up Resources	4-3
Add Hosts	4-3
Start mjs Service	4-4
Start a MATLAB Job Scheduler	4-5
Start Workers	4-7
Stop, Destroy, Resume, Restart Processes	4-9
Move a Worker	4-10
Update the Display	4-10
Test Connectivity	4-11
Export and Import Sessions	4-14
Prepare for Cluster Profiles	4-15

Control Scripts – Alphabetical List

5

Glossary

Introduction

- “MATLAB Parallel Server Product Description” on page 1-2
- “Product Overview” on page 1-3
- “Toolbox and Server Components” on page 1-5
- “Using Parallel Computing Toolbox Software” on page 1-9

MATLAB Parallel Server Product Description

Perform MATLAB and Simulink computations on clusters and clouds

MATLAB Parallel Server lets you scale MATLAB programs and Simulink® simulations to clusters and clouds. You can prototype your programs and simulations on the desktop and then run them on clusters and clouds without recoding. MATLAB Parallel Server supports batch jobs, interactive parallel computations, and distributed computations with large matrices.

All cluster-side licensing is handled by MATLAB Parallel Server. Your desktop license profile is dynamically enabled on the cluster, so you do not need to supply MATLAB licenses for the cluster. The licensing model includes features to support unlimited scaling.

MATLAB Parallel Server runs your programs and simulations as scheduled applications on your cluster. You can use the MATLAB optimized scheduler provided with MATLAB Parallel Server or your own scheduler. A plugin framework enables direct communication with popular cluster scheduler submission clients.

Product Overview

In this section...
“Parallel Computing Concepts” on page 1-3
“Determining Product Installation and Versions” on page 1-4

Parallel Computing Concepts

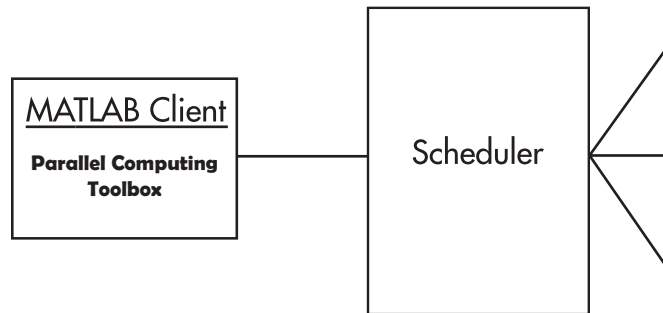
A *job* is some large operation that you need to perform in your MATLAB session. A job is broken down into segments called *tasks*. You decide how best to divide your job into tasks. You could divide your job into identical tasks, but tasks do not have to be identical.

The MATLAB session in which the job and its tasks are defined is called the *client* session. Often, this is on the machine where you program MATLAB. The client uses Parallel Computing Toolbox software to perform the definition of jobs and tasks. The MATLAB Parallel Server product performs the execution of your job by evaluating each of its tasks and returning the result to your client session.

Parallel Computing Toolbox software allows you to run a cluster of MATLAB workers on your local machine in addition to your MATLAB client session. MATLAB Parallel Server software allows you to run as many MATLAB workers on a remote cluster of computers as your licensing allows.

The MATLAB Job Scheduler is the part of the server software that coordinates the execution of jobs and the evaluation of their tasks. The MATLAB Job Scheduler distributes the tasks for evaluation to the server's individual MATLAB sessions called *workers*. Use of the MATLAB Job Scheduler is optional; the distribution of tasks to workers can also be performed by a third-party scheduler, such as Window HPC Server (including CCS), a Platform LSF[®] scheduler, or a PBS Pro[®] scheduler.

See the Glossary for definitions of the parallel computing terms used in this manual.



Basic Parallel Computing Configuration

Determining Product Installation and Versions

To determine if Parallel Computing Toolbox software is installed on your system, type this command at the MATLAB prompt:

```
ver
```

When you enter this command, MATLAB displays information about the version of MATLAB you are running, including a list of all toolboxes installed on your system and their version numbers.

You can run the `ver` command as part of a task in a distributed or parallel application to determine what version of MATLAB Parallel Server software is installed on a worker machine. Note that the toolbox and server software must be the same version.

Toolbox and Server Components

In this section...

“Schedulers, Workers, and Clients” on page 1-5

“Third-Party Schedulers” on page 1-7

“Components on Mixed Platforms or Heterogeneous Clusters” on page 1-8

“mjs Service” on page 1-8

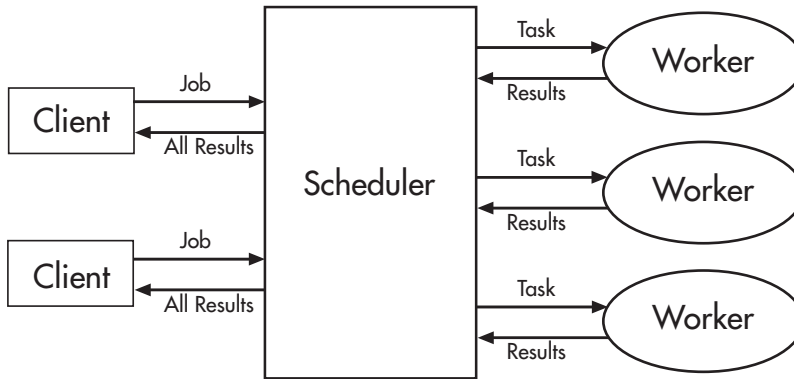
Schedulers, Workers, and Clients

The optional MATLAB Job Scheduler can run on any machine on the network. The MATLAB Job Scheduler runs jobs in the order in which they are submitted, unless any jobs in its queue are promoted, demoted, canceled, or destroyed.

Each worker receives a task of the running job from the MATLAB Job Scheduler, executes the task, returns the result to the MATLAB Job Scheduler, and then receives another task. When all tasks for a running job have been assigned to workers, the MATLAB Job Scheduler starts running the next job with the next available worker.

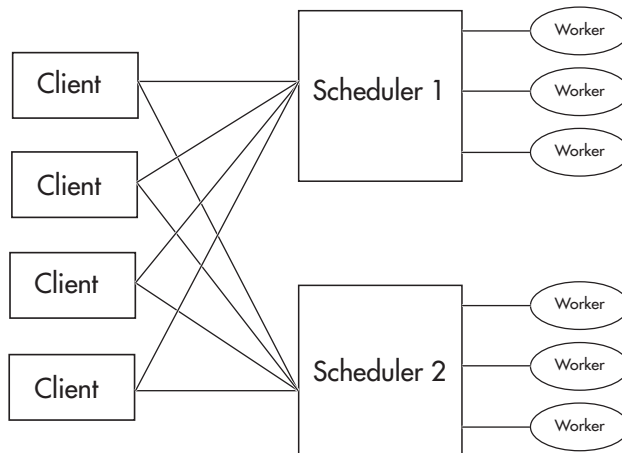
A MATLAB Parallel Server network configuration usually includes many workers that can all execute tasks simultaneously, speeding up execution of large MATLAB jobs. It is generally not important which worker executes a specific task. Each worker evaluates tasks one at a time, returning the results to the MATLAB Job Scheduler. The MATLAB Job Scheduler then returns the results of all the tasks in the job to the client session.

Note For testing your application locally or other purposes, you can configure a single computer to host the client, workers, and MATLAB Job Scheduler. You can also have more than one worker session or more than one MATLAB Job Scheduler session on a machine.



Interactions of Parallel Computing Sessions

A large network might include several MATLAB Job Scheduler sessions as well as several client sessions. Any client session can create, run, and access jobs on any MATLAB Job Scheduler, but a worker session is registered with and dedicated to only one MATLAB Job Scheduler at a time. The following figure shows a configuration with multiple MATLAB Job Scheduler processes.



Configuration with Multiple Clients and MATLAB Job Scheduler Processes

Third-Party Schedulers

As an alternative to using the MATLAB Job Scheduler, you can use a third-party scheduler. This could be a Microsoft® Windows HPC Server (including CCS), Platform LSF scheduler, PBS Pro scheduler, TORQUE scheduler, or a generic scheduler.

Choosing Between a Scheduler and MATLAB Job Scheduler

You should consider the following when deciding to use a scheduler or the MATLAB Job Scheduler for distributing your tasks:

- Does your cluster already have a scheduler?

If you already have a scheduler, you may be required to use it as a means of controlling access to the cluster. Your existing scheduler might be just as easy to use as a MATLAB Job Scheduler, so there might be no need for the extra administration involved.

- Is the handling of parallel computing jobs the only cluster scheduling management you need?

The MATLAB Job Scheduler is designed specifically for MathWorks® parallel computing applications. If other scheduling tasks are not needed, a third-party scheduler might not offer any advantages.

- Is there a file sharing configuration on your cluster already?

The MATLAB Job Scheduler can handle all file and data sharing necessary for your parallel computing applications. This might be helpful in configurations where shared access is limited.

- Are you interested in batch or interactive processing?

When you use a MATLAB Job Scheduler, worker processes usually remain running at all times, dedicated to their MATLAB Job Scheduler. With a third-party scheduler, workers are run as applications that are started for the evaluation of tasks, and stopped when their tasks are complete. If tasks are small or take little time, starting a worker for each one might involve too much overhead time.

- Are there security concerns?

Your scheduler may be configured to accommodate your particular security requirements.

- How many nodes are on your cluster?

If you have a large cluster, you probably already have a scheduler. Consult your MathWorks representative if you have questions about cluster size and the MATLAB Job Scheduler.

- Who administers your cluster?

The person administering your cluster might have a preference for how jobs are scheduled.

Components on Mixed Platforms or Heterogeneous Clusters

Parallel Computing Toolbox software and MATLAB Parallel Server software are supported on Windows®, UNIX® (including Linux®), and Macintosh operating systems. Mixed platforms are supported, so that the clients, MATLAB Job Scheduler, and workers do not have to be on the same platform.

For a complete listing of all network requirements, including those for heterogeneous environments, see the System Requirements page for MATLAB Parallel Server software at

<https://www.mathworks.com/products/matlab-parallel-server/requirements.html>

In a mixed platform environment, be sure to follow the proper installation instructions for each local machine on which you are installing the software.

mjs Service

If you are using the MATLAB Job Scheduler, every machine that hosts a worker or MATLAB Job Scheduler session must also run the mjs service.

The mjs service recovers worker and MATLAB Job Scheduler sessions when their host machines crash. If a worker or MATLAB Job Scheduler machine crashes, when mjs starts up again (usually configured to start at machine boot time), it automatically restarts the MATLAB Job Scheduler and worker sessions to resume their sessions from before the system crash.

Using Parallel Computing Toolbox Software

A typical Parallel Computing Toolbox client session includes the following steps:

- 1** Find a cluster — Your network may have one or more MATLAB Job Scheduler available (but usually only one scheduler). The function you use to find a MATLAB Job Scheduler or scheduler creates an object in your current MATLAB session to represent the MATLAB Job Scheduler or scheduler that will run your job.
- 2** Create a Job — You create a job to hold a collection of tasks. The job exists on the MATLAB Job Scheduler (or scheduler's data location), but a job object in the local MATLAB session represents that job.
- 3** Create Tasks — You create tasks to add to the job. Each task of a job can be represented by a task object in your local MATLAB session.
- 4** Submit a Job to the Job Queue for Execution — When your job has all its tasks defined, you submit it to the queue in the MATLAB Job Scheduler or scheduler. The MATLAB Job Scheduler or scheduler distributes your job's tasks to the worker sessions for evaluation. When all of the workers are completed with the job's tasks, the job moves to the finished state.
- 5** Retrieve the Job's Results — The resulting data from the evaluation of the job is available as a property value of each task object.
- 6** Destroy the Job — When the job is complete and all its results are gathered, you can destroy the job to free memory resources.

Network Administration

This chapter provides information useful for network administration of Parallel Computing Toolbox software and MATLAB Parallel Server software.

- “Requirements and Ports for MATLAB Parallel Server” on page 2-2
- “Use Different MPI Builds on UNIX Systems” on page 2-5
- “Shut Down a Job Manager Cluster” on page 2-8
- “Customize Startup Parameters” on page 2-11
- “Access Service Record Files” on page 2-15
- “Set MATLAB Job Scheduler Cluster Security” on page 2-17
- “Troubleshoot Common Problems” on page 2-22

Requirements and Ports for MATLAB Parallel Server

In this section...

“Port Configuration” on page 2-2

“Fully Qualified Domain Names” on page 2-2

“Security Considerations” on page 2-2

This section discusses the requirements and port configuration for your network to support parallel computing.

Port Configuration

Before you can use MATLAB Parallel Server, you must configure required ports. For details, see “Required Ports” on page 2-24. If you need more help during configuration, see this information from MathWorks Support Team on MATLAB Answers: MATLAB Job Scheduler, or Third-Party Scheduler.

Fully Qualified Domain Names

MATLAB Parallel Server software and Parallel Computing Toolbox software support both short host names and fully qualified domain names. The default usage is short host names. Check the following considerations depending on your scheduler type:

Scheduler	Consideration
MATLAB Job Scheduler	<ul style="list-style-type: none"> If your network requires fully qualified host names, you can use the <code>mjs_def</code> file to identify the worker nodes by their full names. See “Customize Startup Parameters” on page 2-11. To set the host name used for a MATLAB client session, see the <code>pctconfig</code> reference page.
Third-Party scheduler	<ul style="list-style-type: none"> To set the host name used for a MATLAB client session, see the <code>pctconfig</code> reference page.

Security Considerations

Check the following table for security considerations when using MATLAB Parallel Server:

Scheduler	Security Consideration
MATLAB Job Scheduler	<ul style="list-style-type: none"> • MATLAB workers run as whatever user the administrator starts the node's mjs service under. By default, the mjs service starts as root on UNIX operating systems, and as LocalSystem on Microsoft Windows operating systems. Because MATLAB provides system calls, users can submit jobs that execute shell commands. • The mjs service does not enforce any access control or authentication. Anyone with local or remote access to the mjs services can start and stop their workers and job managers, and query for their status. • The job manager does not restrict access to the cluster, nor to job and task data. For information on security options, see "Set MATLAB Job Scheduler Cluster Security" on page 2-17. Using a third-party scheduler instead of the MathWorks job manager could allow you to take advantage of the security measures it provides. • The parallel computing processes must all be on the same side of a firewall, or you must take measures to enable them to communicate with each other through the firewall. Workers running tasks of the same communicating job cannot be firewalled off from each other, because their MPI-based communication will not work. • If certain ports are restricted, you can specify the ports used for parallel computing. See "Define Script Defaults" on page 2-11. • If your organization is a member of the Internet Multicast Backbone (MBone), make sure that your parallel computing cluster is isolated from MBone access if you are using multicast for parallel computing. Isolation is generally the default condition. If you have any questions about MBone membership, contact your network administrator.
Third-Party scheduler	<ul style="list-style-type: none"> • The parallel computing processes must all be on the same side of a firewall, or you must take measures to enable them to communicate with each other through the firewall. Workers running tasks of the same communicating job cannot be firewalled off from each other, because their MPI-based communication will not work.

See Also

“Integrate MATLAB Job Scheduler for Network License Manager” on page 3-6 |

“Integrate MATLAB with Third-Party Schedulers” on page 3-2

Use Different MPI Builds on UNIX Systems

In this section...

“Build MPI” on page 2-5

“Use Your MPI Build” on page 2-5

Build MPI

On Linux and Macintosh operating systems, you can use an MPI build that differs from the one provided with Parallel Computing Toolbox. This topic outlines the steps for creating an MPI build for use with the generic scheduler interface. If you already have an alternative MPI build, proceed to “Use Your MPI Build” on page 2-5.

- 1 Unpack the MPI sources into the target file system on your machine. For example, suppose you have downloaded `mpich2-distro.tgz` and want to unpack it into `/opt` for building:

```
# cd /opt
# mkdir mpich2 && cd mpich2
# tar zxvf path/to/mpich2-distro.tgz
# cd mpich2-1.4.1p1
```

- 2 Build your MPI using the `enable-shared` option (this is vital, as you must build a shared library MPI, binary compatible with `MPICH2-1.4.1p1` for R2013b to R2018b, or `MPICH3.2.1` for R2019a and later). For example, the following commands build an MPI with the `nemesis` channel device and the `gforker` launcher.

```
#!/configure -prefix=/opt/mpich2/mpich2-1.4.1p1 \
--enable-shared --with-device=ch3:nemesis \
--with-pm=gforker 2>&1 | tee log
# make 2>&1 | tee -a log
# make install 2>&1 | tee -a log
```

Use Your MPI Build

When your MPI build is ready, this stage highlights the steps to use it with a generic scheduler. To get your cluster working with a different MPI build, follow these steps.

- 1 Test your build by running the `mpiexec` executable. The build should be ready to test if its `bin/mpiexec` and `lib/libmpich.so` are available in the MPI installation location.

Following the example in “Build MPI” on page 2-5, `/opt/mpich2/mpich2-1.4.1p1/bin/mpiexec` and `/opt/mpich2/mpich2-1.4.1p1/lib/libmpich.so` are ready to use, so you can test the build with:

```
$ /opt/mpich2/mpich2-1.4.1p1/bin/mpiexec -n 4 hostname
```

- 2 Create an `mpiLibConf` function to direct Parallel Computing Toolbox to use your new MPI. Write your `mpiLibConf.m` to return the appropriate information for your build. For example:

```
function [primary, extras] = mpiLibConf
primary = '/opt/mpich2/mpich2-1.4.1p1/lib/libmpich.so';
extras = {};
```

The primary path *must* be valid on the cluster; and your `mpiLibConf.m` file must be higher on the cluster workers' path than `matlabroot/toolbox/distcomp/mpi`. (Sending `mpiLibConf.m` as an attached file for this purpose does not work. You can get the `mpiLibConf.m` function on the worker path by either moving the file into a folder on the path, or by having the scheduler use `cd` in its command so that it starts the MATLAB worker from within the folder that contains the function.)

- 3 Determine necessary daemons and command-line options.
 - Determine all necessary daemons (often something like `mpdboot` or `smpd`). The `gforker` build example in this section uses an MPI that needs no services or daemons running on the cluster, but it can use only the local machine.
 - Determine the correct command-line options to pass to `mpiexec`.
- 4 To set up your cluster to use your new MPI build, modify your communicating job wrapper script to pick up the correct `mpiexec`. Additionally, there might be a stage in the wrapper script where the MPI process manager daemons are launched.

The communicating job wrapper script must:

- Determine which nodes are allocated by the scheduler.
- Start required daemon processes. For example, for the MPD process manager this means calling `"mpdboot -f <nodefile>"`.
- Define which `mpiexec` executable to use for starting workers.
- Stop the daemon processes. For example, for the MPD process manager this means calling `"mpdallexit"`.

For examples of communicating job wrapper scripts, see the subfolders of `matlabroot/toolbox/distcomp/examples/integration/`; specifically, for an

example for Sun Grid Engine, look in the subfolder `sgc/shared` for `communicatingJobWrapper.sh`. Wrapper scripts are available for various schedulers and file sharing configurations. Copy and modify the appropriate script for your particular cluster usage.

Shut Down a Job Manager Cluster

In this section...

“Linux and Macintosh Operating Systems” on page 2-8

“Microsoft Windows Operating Systems” on page 2-9

If you are done using the job manager and its workers, you might want to shut down the server software processes so that they are not consuming network resources. You do not need to be at the computer running the processes that you are shutting down. You can run these commands from any machine with network access to the processes. The following sections explain shutting down the processes for different platforms.

Linux and Macintosh Operating Systems

Enter the commands of this section at the prompt in a shell.

Stopping the Job Manager and Workers

- 1 To shut down the job manager, enter the commands

```
cd matlabroot/toolbox/distcomp/bin
```

(Enter the following command on a single line.)

```
stopjobmanager -remotehost <job manager hostname> -name  
<MyJobManager> -v
```

If you have more than one job manager running, stop each of them individually by host and name.

For a list of all options to the script, type

```
stopjobmanager -help
```

- 2 For each MATLAB worker you want to shut down, enter the commands

```
cd matlabroot/toolbox/distcomp/bin  
stopworker -remotehost <worker hostname> -v
```

If you have more than one worker session running, you can stop each of them individually by host and name.

```
stopworker -name worker1 -remotehost <worker hostname>  
stopworker -name worker2 -remotehost <worker hostname>
```

For a list of all options to the script, type

```
stopworker -help
```

Stop and Uninstall the mjs Daemon

Normally, you configure the mjs daemon to start at system boot time and continue running until the machine shuts down. However, if you plan to uninstall the MATLAB Parallel Server product from a machine, you might want to uninstall the mjs daemon also, because you no longer need it.

Note You must have root privileges to stop or uninstall the mjs daemon.

- 1 Use the following command to stop the mjs daemon:

```
/etc/init.d/mjs stop
```

- 2 Remove the installed link to prevent the daemon from starting up again at system reboot:

```
cd /etc/init.d/  
rm mjs
```

Stop the Daemon Manually

If you used the alternative manual startup of the mjs daemon, use the following commands to stop it manually:

```
cd matlabroot/toolbox/distcomp/bin  
mjs stop
```

Microsoft Windows Operating Systems

Stop the Job Manager and Workers

Enter the commands of this section at the prompt in a DOS command window.

- 1 To shut down the job manager, enter the commands

```
cd matlabroot\toolbox\distcomp\bin
```

(Enter the following command on a single line.)

```
stopjobmanager -remotehost <job manager hostname> -name  
<MyJobManager> -v
```

If you have more than one job manager running, stop each of them individually by host and name.

For a list of all options to the script, type

```
stopjobmanager -help
```

- 2 For each MATLAB worker you want to shut down, enter the commands

```
cd matlabroot\toolbox\distcomp\bin  
stopworker -remotehost <worker hostname> -name <worker name> -v
```

If you have more than one worker session running, you can stop each of them individually by host and name.

```
stopworker -remotehost <worker hostname> -name <worker1 name>  
stopworker -remotehost <worker hostname> -name <worker2 name>
```

For a list of all options to the script, type

```
stopworker -help
```

Stop and Uninstall the mjs Service

Normally, you configure the mjs service to start at system boot time and continue running until the machine shuts down. If you need to stop the mjs service while leaving the machine on, enter the following commands at a DOS command prompt:

```
cd matlabroot\toolbox\distcomp\bin  
mjs stop
```

If you plan to uninstall the MATLAB Parallel Server product from a machine, you might want to uninstall the mjs service also, because you no longer need it.

You do not need to stop the service before uninstalling it.

To uninstall the mjs service, enter the following commands at a DOS command prompt:

```
cd matlabroot\toolbox\distcomp\bin  
mjs uninstall
```

Customize Startup Parameters

In this section...

“Define Script Defaults” on page 2-11

“Override Script Defaults” on page 2-13

The MATLAB Parallel Server scripts run using several default parameters. You can customize the scripts, as described in this section.

Define Script Defaults

The scripts for the server services require values for several parameters. These parameters set the process name, the user name, log file location, ports, etc. Some of these can be set using flags on the command lines, but the full set of user-configurable parameters are in the `mjs_def` file.

Note The startup script flags take precedence over the settings in the `mjs_def` file.

The default parameters used by the server service scripts are defined in the file:

- `matlabroot\toolbox\distcomp\bin\mjs_def.bat` (on Microsoft Windows operating systems)
- `matlabroot/toolbox/distcomp/bin/mjs_def.sh` (on Linux or Macintosh operating systems)

To set the default parameters, edit this file before installing or starting the `mjs` service.

The `mjs_def` file is self-documented, and includes explanations of all its parameters.

Note If you want to run more than one job manager on the same machine, they must all have unique names. Specify the names using flags with the startup commands.

Set the User

By default, the job manager and worker services run as the user who starts them. You can run the services as a different user with the following settings in the `mjs_def` file.

Parameter	Description
MJSUSER	Set this parameter to run the mjs services as a user different from the user who starts the service. On a UNIX operating system, set the value before starting the service; on a Windows operating system, set it before installing the service.
MJSPASS	On a Windows operating system, set this parameter to specify the password for the user identified in the MJSUSER parameter; otherwise, the system prompts you for the password when the service is installed.

On UNIX operating systems, MJSUSER requires that the current machine has the `sudo` utility installed, and that the current user be allowed to use `sudo` to execute commands as the user identified by MJSUSER. For further information, refer to your system documentation on the `sudo` and `sudors` utilities (for example, `man sudo` and `man sudors`).

The MJSUSER is granted these permissions on Windows systems:

Privilege	Purpose	Local Security Settings Policy
SeServiceLogonRight	Required to log on using the service logon type.	Log on as a service
SeAssignPrimaryTokenPrivilege	Required to start a process under a different user account.	Replace a process level token
SeIncreaseQuotaPrivilege	Required to start a process under a different user account.	Adjust memory quotas for a process

To modify or remove these privileges,

- 1 Select the Windows menu **Start > Settings > Control Panel**.
- 2 Double-click **Administrative Tools**, then **Local Security Policy**.
- 3 In the tree, select **Local Policies**, then in the right pane, double-click **User Rights Assignment**.

The table above indicates which policies are affected by MJSUSER. Double-click any of the listed policies in the Local Security Settings GUI to alter its setting or remove a user from that policy.

Override Script Defaults

Specify an Alternative Defaults File

The default parameters used by the mjs service, job managers, and workers are defined in the file:

- *matlabroot\toolbox\distcomp\bin\mjs_def.bat* (on Windows operating systems)
- *matlabroot/toolbox/distcomp/bin/mjs_def.sh* (on Linux or Macintosh operating systems)

Before installing and starting the mjs service, you can edit this file to set the default parameters with values you require.

Alternatively, you can make a copy of this file, modify the copy, and specify that this copy be used for the default parameters.

On Linux or Macintosh operating systems, enter the command

```
mjs start -mjsdef my_mjs_def.sh
```

On Windows operating systems, enter the command

```
mjs install -mjsdef my_mjs_def.bat  
mjs start -mjsdef my_mjs_def.bat
```

If you specify a new *mjs_def* file instead of the default file for the service on one computer, the new file is not automatically used by the mjs service on other computers. If you want to use the same alternative file for all your mjs services, you must specify it for each mjs service you install or start.

For more information, see “Define Script Defaults” on page 2-11.

Note The startup script flags take precedence over the settings in the *mjs_def* file.

Start in a Clean State

When a job manager or worker starts up, it normally resumes its session from the past. This way, a job queue is not destroyed or lost if the job manager machine crashes or if the

job manager is inadvertently shut down. To start up a job manager or worker from a clean state, with all history deleted, use the `-clean` flag on the `start` command:

```
startjobmanager -clean -name MyJobManager  
startworker -clean -jobmanager MyJobManager
```

Access Service Record Files

In this section...

“Locate Log Files” on page 2-15

“Locate Checkpoint Folders” on page 2-15

The MATLAB Parallel Server services generate various record files in the normal course of their operations. The mjs service, job manager, and worker sessions all generate such files. This section describes the types of information stored by the services.

Locate Log Files

Log files for each service contain entries for the service’s operations. These might be of particular interest to the network administrator in cases when problems arise.

Operating System	File Location
Windows	<p>The default location of the log files is <TEMP>\MJS\Log, where <TEMP> is the value of the system TEMP variable. For example, if TEMP is set to C:\TEMP, the log files are placed in C:\TEMP\MJS\Log.</p> <p>You can set alternative locations for the log files by modifying the LOGBASE setting in the mjs_def.bat file before starting the mjs service.</p>
Linux and Macintosh	<p>The default location of the log files is /var/log/mjs/.</p> <p>You can set alternative locations for the log files by modifying the LOGBASE setting in the mjs_def.sh file before starting the mjs service.</p>

Locate Checkpoint Folders

Checkpoint folders contain information related to persistence data, which the server services use to create continuity from one instance of a session to another. For example, if you stop and restart a job manager, the new session continues the old session, using all the same data.

A primary feature offered by the checkpoint folders is in crash recovery. This allows server services to automatically resume their sessions after a system goes down and comes back up, minimizing the loss of data. However, if a MATLAB worker goes down during the evaluation of a task, that task is neither reevaluated nor reassigned to another worker. In this case, a finished job may not have a complete set of output data, because data from any unfinished tasks might be missing.

Note If a job manager crashes and restarts, its workers can take up to 2 minutes to reregister with it.

Platform	File Location
Windows	<p>The default location of the checkpoint folders is <TEMP>\MJS\Checkpoint, where <TEMP> is the value of the system TEMP variable. For example, if TEMP is set to C:\TEMP, the checkpoint folders are placed in C:\TEMP\MJS\Checkpoint.</p> <p>You can set alternative locations for the checkpoint folders by modifying the CHECKPOINTBASE setting in the <code>mjs_def.bat</code> file before starting the mjs service.</p>
Linux and Macintosh	<p>The checkpoint folders are placed by default in <code>/var/lib/mjs/</code>.</p> <p>You can set alternative locations for the checkpoint folder by modifying the CHECKPOINTBASE setting in the <code>mjs_def.sh</code> file before starting the mjs service.</p>

Set MATLAB Job Scheduler Cluster Security

In this section...

“Set the Security Level” on page 2-17

“Local, MATLAB Job Scheduler, and Network Passwords” on page 2-19

“Set Secure Communication” on page 2-20

Set the Security Level

You set the MATLAB Job Scheduler security level with the `SECURITY_LEVEL` parameter in the `mjs_def` file before starting the `mjs` service on your cluster nodes. The `mjs_def` file indicates what values are allowed, and briefly describes each security level.

The following table describes the available security levels for accessing a MATLAB Job Scheduler and its jobs.

Security Level	Description	User Requirements
0	<p>No security.</p> <ul style="list-style-type: none"> Any user can access any job. Tasks run as the user who started the <code>mjs</code> process on the worker machines (typically root or Local System). This is the default, and is the behavior in all releases prior to R2010b. 	<ul style="list-style-type: none"> Jobs are associated with the default user name of the programmer, but no protection is provided.
1	<p>Jobs are identified with the submitting user.</p> <ul style="list-style-type: none"> Any user can access any job; a dialog warns if the accessed job belongs to another user. Tasks run as the user who started the <code>mjs</code> process on the worker machines (typically root or Local System). 	<ul style="list-style-type: none"> A dialog requires you to establish a user name when you first access the job manager. Your MATLAB Job Scheduler user name does not have to match your system/network user name. No passwords are used.

Security Level	Description	User Requirements
2	<p>Job manager MATLAB Job Scheduler password protection on jobs.</p> <ul style="list-style-type: none">• Jobs and tasks are identified with the submitting user, and are password protected. Other users cannot access your jobs.• Tasks run as the user who started the mjs process on the worker machines (typically root or Local System).	<ul style="list-style-type: none">• When you start the MATLAB Job Scheduler, it prompts you to provide a new password for that job manager's admin account, which can be used for accessing all users' jobs and tasks.• A dialog box requires you to establish a user name and password when you first access the MATLAB Job Scheduler from the MATLAB client.• Your MATLAB Job Scheduler user name and password do not have to match your system/network user name and password.

Security Level	Description	User Requirements
3	<p>In addition to the security of level 2, tasks run as the submitting user on worker machines.</p> <ul style="list-style-type: none"> Jobs and tasks are identified with the submitting user, and are password protected. Other users cannot access your jobs. Tasks run as the user who submitted the job. 	<ul style="list-style-type: none"> On UNIX systems, the mjs process on the cluster nodes must be started by the root user. The MATLAB Job Scheduler must use secure communication with the workers (set in the mjs_def file). When you start the MATLAB Job Scheduler, it prompts you to provide a new password for that job manager's admin account, which can be used for accessing all users' jobs and tasks. A dialog box requires you to establish a user name and password when you first access the MATLAB Job Scheduler from the MATLAB client. Your job manager MATLAB Job Scheduler user name and password must be the same as your system/network user name and password, because the worker must log you in to run the task as you. All users that tasks run as, require read and write permissions to the CHECKPOINTBASE folder and all its subfolders.

The job manager and the workers should run at the same security level. A worker running at too low a security level will fail to register with the job manager, because the job manager does not trust it.

Local, MATLAB Job Scheduler, and Network Passwords

For any security above level 0, when you start the MATLAB Job Scheduler (for example, with the `startjobmanager` command), a cluster user account named `admin` is created for this cluster, and you are prompted to provide a password for this new account. The

cluster `admin` account has all the necessary permissions for accessing the cluster and all its jobs.

For any security level, the MATLAB Job Scheduler identifies every job with the user who submits the job. Therefore, whenever you access the MATLAB Job Scheduler or a job, the MATLAB Job Scheduler must be aware of who you are.

At security level 0, the MATLAB Job Scheduler and job objects' `UserName` property is set to the login name of the person who creates the job; this setting can be changed at any time. For all higher security levels, the first access to the MATLAB Job Scheduler causes a dialog box to open which asks for your username; if the security level is 2 or 3, you must also provide a password. The username and password you provide for the MATLAB Job Scheduler needs to match your network username and password *only* if you are using security level 3; otherwise, you can create a new username and password unique for the MATLAB Job Scheduler. For your convenience, you can choose how long to save your username and password on the local computer, so that you do not need to enter them every time you access your job.

For information about changing a password and logging out of a MATLAB Job Scheduler, see `changePassword` and `logout`.

Set Secure Communication

To establish secure encrypted communication between MATLAB Job Scheduler, client and workers, set

- `USE_SECURE_COMMUNICATION = true`
- `ALL_SERVER_SOCKETS_IN_CLUSTER = true` (default)

in the `mjs_def` file. Secure encrypted communication is provided via `SSLSocket` using `TLSv1.2` only.

Note If `ALL_SERVER_SOCKETS_IN_CLUSTER = false` in the `mjs_def` file, then secure encrypted communication is established between MATLAB Job Scheduler and workers only.

You must also provide a value for the `SHARED_SECRET_FILE` parameter in the `mjs_def` file, identifying where the file can be found from the MATLAB Job Scheduler perspective. To create this file, run either script:

- `matlabroot/toolbox/distcomp/bin/createSharedSecret` (UNIX)
- `matlabroot\toolbox\distcomp\bin\createSharedSecret.bat` (Windows)

The secret file establishes trust between the processes on different machines.

- In a shared file system, all the nodes can point to the same secret file, and they can even all share the same `mjs_def` file.
- In a nonshared file system, create a secret file with the provided script, then copy the file to each node and make sure each node's `mjs_def` file indicates where its particular secret file is located.

Note Secure communication is required when using MATLAB Job Scheduler security level 3.

Troubleshoot Common Problems

In this section...
“License Errors” on page 2-22
“Memory Errors on UNIX Operating Systems” on page 2-24
“Run Server Processes on Windows Network Installation” on page 2-24
“Required Ports” on page 2-24
“Ephemeral TCP Ports with Job Manager” on page 2-25
“Host Communications Problems” on page 2-25
“Verify Network Communications for Cluster Discovery” on page 2-26

This section offers advice on solving problems you might encounter with MATLAB Parallel Server software.

License Errors

When starting a MATLAB worker, a licensing problem might result in the message

```
License checkout failed. No such FEATURE exists.  
License Manager Error -5
```

There are many reasons why you might receive this error:

- This message usually indicates that you are trying to use a product for which you are not licensed. Look at your `license.dat` file located within your MATLAB installation to see if you are licensed to use this product.
- If you are licensed for this product, this error may be the result of having extra carriage returns or tabs in your license file. To avoid this, ensure that each line begins with either `#`, `SERVER`, `DAEMON`, or `INCREMENT`.

After fixing your `license.dat` file, restart the network license manager and MATLAB should work properly.

- This error may also be the result of an incorrect system date. If your system date is before the date that your license was made, you will get this error.
- If you receive this error when starting a worker with MATLAB Parallel Server software:

- You may be calling the `startworker` command from an installation that does not have access to a worker license. For example, starting a worker from a client installation of the Parallel Computing Toolbox product causes the following error:

The `mjs` service on the host `hostname` returned the following error:

Problem starting the MATLAB worker.

The cause of this problem is:

```
=====
Most likely, the MATLAB worker failed to start due to a
licensing problem, or MATLAB crashed during startup. Check
the worker log file
/tmp/mjs_user/node_node_worker_05-11-01_16-52-03_953.log
for more detailed information. The mjs log file
/tmp/mjs_user/mjs-service.log
may also contain some additional information.
=====
```

In the worker log files, you see the following information:

```
License checkout failed.
License Manager Error -15
MATLAB is unable to connect to the license server.
Check that the license manager has been started, and that the
MATLAB client machine can communicate with the license server.
```

Troubleshoot this issue by visiting:
<https://www.mathworks.com/support/lme/R2009a/15>

```
Diagnostic Information:
Feature: MATLAB_Distrib_Comp_Engine
License path: /apps/matlab/etc/license.dat
FLEXnet Licensing error: -15,570. System Error: 115
```

- If you installed only the Parallel Computing Toolbox product, and you are attempting to run a worker on the same machine, you will receive this error because the MATLAB Parallel Server product is not installed, and therefore the worker cannot obtain a license.

Memory Errors on UNIX Operating Systems

If the number of threads created by the server services on a machine running a UNIX operating system (Linux or Macintosh) exceeds the limitation set by the `maxproc` value, the services fail and generate an out-of-memory error. Check your `maxproc` value on a UNIX operating system with the `limit` command. (Different versions of UNIX software might have different names for this property.)

Run Server Processes on Windows Network Installation

Many networks are configured not to allow `LocalSystem` to have access to UNC or mapped network shares. In this case, run the `mjs` process under a different user with rights to log on as a service. See “Set the User” on page 2-11.

Required Ports

With Job Manager

BASE_PORT

The `mjs_def` file specifies and describes the ports required by the job manager and all workers. See the following file in the MATLAB installation used for each cluster process:

- `matlabroot/toolbox/distcomp/bin/mjs_def.sh` (on UNIX operating systems)
- `matlabroot\toolbox\distcomp\bin\mjs_def.bat` (on Windows operating systems)

Communicating Jobs

On worker machines running a UNIX operating system, the number of ports required by MPICH for the running of communicating jobs ranges from `BASE_PORT + 1000` to `BASE_PORT + 2000`.

With Third-Party Scheduler

Before the worker processes start, you can control the range of ports used by the workers for communicating jobs by defining the environment variable `MPICH_PORT_RANGE` with the value `minport:maxport`.

Client Ports

With the `pctconfig` function, you specify the ports used by the client. If the default ports cannot be used, this function allows you to configure ports separately for communication with the job scheduler and communication with `pmode` or a parallel pool.

Ephemeral TCP Ports with Job Manager

If you use the job manager on a cluster of nodes running Windows operating systems, you must make sure that a large number of ephemeral TCP ports are available on the job manager machine. By default, the maximum valid ephemeral TCP port number on a Windows operating system is 5000, but transfers of large data sets might fail if this setting is not increased. In particular, if your cluster has 32 or more workers, you should increase the maximum valid ephemeral TCP port number using the following procedure:

- 1 Start the Registry Editor.
- 2 Locate the following subkey in the registry, and click **Parameters**:
`HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters`
- 3 On the Registry Editor window, select **Edit > New > DWORD Value**.
- 4 In the list of entries on the right, change the new value name to `MaxUserPort` and press **Enter**.
- 5 Right-click on the `MaxUserPort` entry name and select **Modify**.
- 6 In the Edit DWORD Value dialog, enter `65534` in the **Value data** field. Select **Decimal** for the **Base** value. Click **OK**.

This parameter controls the maximum port number that is used when a program requests any available user port from the system. Typically, ephemeral (short-lived) ports are allocated between the values of 1024 and 5000 inclusive. This action allows allocation for port numbers up to 65534.

- 7 Quit the Registry Editor.
- 8 Reboot your machine.

Host Communications Problems

If a worker is not able to make a connection with its MATLAB Job Scheduler, or if a client session cannot validate a profile that uses that scheduler, this might indicate communications problems between nodes.

With Command-Line Interface

First, be sure that the machines in question agree on their IP resolutions. The IP address for a particular host should be the same for itself as it is from the perspective of another host. For example, if a process on `hostB` cannot connect to one on `hostA`, find out the `hostA` IP address for itself, then see what the IP address for `hostA` is from `hostB`. They should be the same.

If the machines can identify each other, the `nodestatus` command can be useful for diagnosing problems between their processes. Use the function to determine what MATLAB Parallel Server processes are running on the local host, and which are accessible from remote hosts. If a worker on `hostA` cannot register with its job manager on `hostB`, run `nodestatus` on both hosts to see what each can see on `hostB`.

On `hostB`, execute:

```
nodestatus -remotehost hostB
```

Then on `hostA`, run exactly the same command:

```
nodestatus -remotehost hostB
```

The results should be the same, showing the same listing of job managers and workers.

If the output indicates problems, run the command again with a higher information level to receive more detailed information:

```
nodestatus -remotehost hostB -infolevel 3
```

With Admin Center GUI

You can diagnose some communications problems using Admin Center.

If you cannot successfully add hosts to the listing by specifying host name, you can use their IP addresses instead (see “Add Hosts” on page 4-3). If you suspect any communications problems, in the Admin Center GUI click **Test Connectivity** (see “Test Connectivity” on page 4-11). This testing verifies that the nodes can identify each other and allow their processes to communicate with each other.

Verify Network Communications for Cluster Discovery

If you want to use the discover cluster capabilities in Parallel Computing Toolbox, your network must be configured with at least one of the following:

- “DNS SRV Record” on page 2-27
- “Multicast” on page 2-28

DNS SRV Record

Using DNS for cluster discovery requires that you have a DNS SRV record of the following general form:

```
_mdcs._tcp.domainname.com SSSS IN SRV PPPP WWW MJS_PORT MJS_FQDN_HOSTNAME
```

The parts of this record are:

- `_mdcs._tcp`. The record must start with this text, followed by your domain name (like `company.com` or `university.edu`) that the client machine searches.
- `SSSS` indicates how long (in seconds) the DNS record can be cached; 3600 is recommended.
- `IN SRV` is required as shown, indicating that this is a service record.
- `PPPP` and `WWW` indicate priority and wait values. These are not used, so 0 is recommended for each.
- `MJS_PORT` is the port on which you connect to the MATLAB Job Scheduler server. The default is 27350, but if you change it for the server you must change it here accordingly.
- `MJS_FQDN_HOSTNAME` is the fully qualified domain name for the host serving the MATLAB Job Scheduler. For example, `mjs-1.company.com`.

A valid DNS SRV record for the `company.com` network running a MATLAB Job Scheduler on machine `mjs-1` might look like this:

```
_mdcs._tcp.company.com 3600 IN SRV 0 0 27350 mjs-1.company.com
```

For your network, create the appropriate DNS SRV record using the standard procedure for your DNS system. Then you can verify that your network is configured with the necessary DNS SRV records by using standard utilities, such as the `nslookup` command. For example, this system command indicates the existence of the applicable DNS SRV records:

```
nslookup -type=SRV _mdcs._tcp.company.com
```

Multicast

To use multicast, it is required on the head node running the MATLAB Job Scheduler and on the client system.

Multicast, unlike TCP/IP or UDP, is a subscription-based protocol where a number of machines on a network indicate to the network their interest in particular packets originating somewhere on that network. By contrast, both UDP and TCP packets are always bound for a single machine, usually indicated by its IP address.

The main tools for investigating this type of packet are:

- `tcpdump` for UNIX operating systems
- `wireshark` and `ethereal` for Microsoft Windows operating systems
- A Java® class included with the parallel computing products.

The Java class is called `com.mathworks.toolbox.distcomp.test.MulticastTester`. Both its static main method and its constructor take two input arguments: the multicast group to join and the port number to use.

This Java class has a number of simple methods to attempt to join a specified multicast group. Once the class has successfully joined the group, it has methods to send messages to the group, listen for messages from the group, and display what it receives. You can use this class both from a command-line call to Java software and inside MATLAB.

From a shell prompt (assuming that `java` is on your path), type

```
java -cp distcomp.jar com.mathworks.toolbox.distcomp.test.MulticastTester
```

You should see an output something like this:

```
0 : host1name : 0
1 : host2name : 0
```

The following example shows how to use the Java class inside MATLAB.

Start MATLAB on two machines (e.g., `host1name` and `host2name`) for which you want to test multicast. In each MATLAB session, enter the following commands:

```
m = com.mathworks.toolbox.distcomp.test.MulticastTester('239.1.1.1', 9999);
m.startSendingThread;
m.startListeningThread;
```


These instructions cause each MATLAB session to issue a stream of multicast test packets, and to listen for test packets. If multicast is working between the machines, you see a stream of lines like the following:

```
0 : host1name : 0  
1 : host2name : 0  
2 : host2name : 1  
3 : host2name : 2
```

The number on the left in each character vector is the line number for the received packet. The text in the center is the host from which the packet is received. The number on the right is the packet number sent by the sending host. It is normal for a host to report a test packet from itself.

If either machine does not receive a stream of test packets, or if the remote host is not included in either stream, then multicast communication is not operating properly.

To terminate the test stream, execute the following in both MATLAB sessions:

```
m.stopSendingThread;  
m.stopListeningThread;
```


Product Installation

Integrate MATLAB with Third-Party Schedulers

If you already have a cluster with a scheduler, follow these instructions to integrate MATLAB with your scheduler using MATLAB Parallel Server. If you do not have an existing scheduler in your cluster, see: “Integrate MATLAB Job Scheduler for Network License Manager” on page 3-6.

After you integrate MATLAB with a scheduler, you can access workers in your cluster from a desktop MATLAB client session with Parallel Computing Toolbox.

The setup in these steps uses the network license manager.

Activate Your MATLAB Parallel Server License

To install MATLAB Parallel Server, you must activate your license. To activate your MATLAB Parallel Server license:

- 1 Navigate to <https://www.mathworks.com/licensecenter>.
- 2 Log into the Administrator’s MathWorks Account.
- 3 Select your MATLAB Parallel Server license, and click the **Install and Activate** tab.
- 4 At the rightmost side, under **RELATED TASKS**, select **Activate to Retrieve License File**.
- 5 Fill in the requested information. This information must refer to the machine that hosts the license manager. In these instructions, it is the head node. For more information, see “Install License Manager on the Head Node” on page 3-3.
- 6 After filling in the information, download or email the License File and copy the File Installation Key. These are used later in the process.

Note Activation is not necessary for trials. Contact your sales representative to obtain the License File and the File Installation Key.

Get the Installation Files

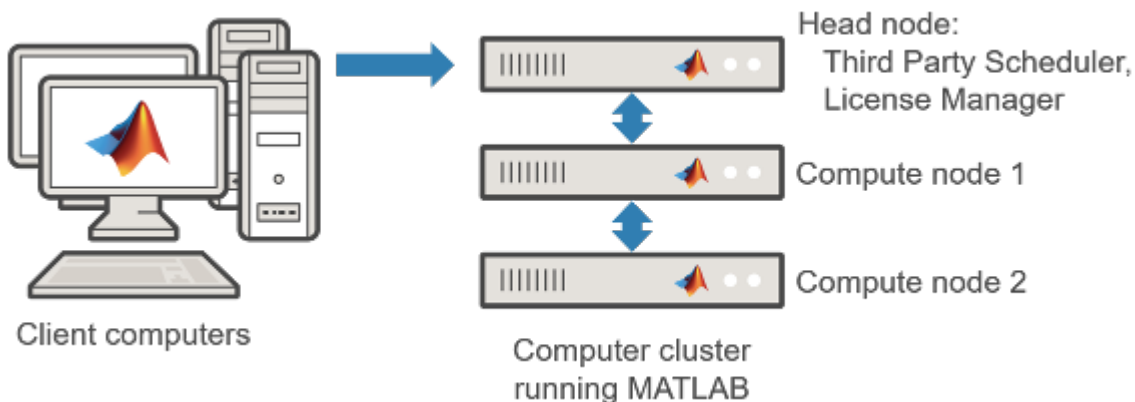
To save time and eliminate the need for the installer-based download process for each computer in your cluster, download the installation files prior to installation. Doing so facilitates installation in a large number of machines. If you have access to an Administrator’s account for your license, you can use the installer to download files

without installing them. If not, contact the administrator of your license to obtain a copy of the installation files. For more information, see “Download Products Without Installation” (Installation, Licensing, and Activation). When using the installer to download the files, choose the following options:

- Select the appropriate operating system for the cluster nodes.
- Select all products for download. MATLAB Parallel Server cannot run jobs requiring products that are not installed.

There are three server-side components of MATLAB Parallel Server:

- 1 The license manager, which hosts the MATLAB Parallel Server license used by each worker. For more information, see “Install License Manager on the Head Node” on page 3-3.
- 2 Your third-party job scheduler, which runs on the head node and manages jobs on your cluster. You integrate this scheduler with MATLAB Parallel Server. For more information, see “Install License Manager on the Head Node” on page 3-3 and “Configure Your Cluster” on page 3-5.
- 3 MATLAB Parallel Server, which runs on the compute nodes. For more information, see “Install Software on Worker Nodes” on page 3-4.



Install License Manager on the Head Node

The head node is the computer node that hosts your third-party job scheduler. For the installation, use the offline installer from the previous step. For more information on the

offline installation procedure, see “Install Products Offline” (Installation, Licensing, and Activation).

- 1 Start the MATLAB installer from the installation files acquired in “Get the Installation Files” on page 3-2.
- 2 Select **Use a File Installation Key**, and continue through the prompts.
- 3 In the product selection, select only the license manager.
- 4 In the License File step, browse to your `license.lic` file (obtained from “Activate Your MATLAB Parallel Server License” on page 3-2).
- 5 Start the license manager.

Install Software on Worker Nodes

This procedure is similar to “Install License Manager on the Head Node” on page 3-3. For more information on the offline installation procedure, see “Install Products Offline” (Installation, Licensing, and Activation).

- 1 Start the MATLAB installer from the installation files acquired in “Get the Installation Files” on page 3-2.
- 2 Select **Use a File Installation Key** and continue through the prompts.
- 3 Select all products. However, do not install the license manager.
- 4 Use the `license.dat` file from the head node. You can obtain this file from the `matlabroot/etc` folder, where `matlabroot` is the MATLAB installation folder.

For best performance, install locally on each node. However, you can also install in a network share location. To save time, you can perform a noninteractive installation (silent installation) on the worker nodes. For instructions, see “Install Noninteractively (Silent Installation)” (Installation, Licensing, and Activation).

This license does not allow you to run MATLAB from the worker nodes. If you want to test the installation, complete the following setup.

Install Software on Local Desktop

To use MATLAB Parallel Server, you must use a local desktop running MATLAB and Parallel Computing Toolbox. Install the MathWorks products for which you are licensed, including Parallel Computing Toolbox, on the local desktops from which you want to

submit jobs to the cluster. For help with this step, see “Installation, Licensing, and Activation”.

Configure Your Cluster

When the cluster and client installations are complete, you can proceed to configure the products for the job scheduler of your choice. Use this table to choose a suitable guide to complete your configuration.

Next step	More information
<p>If your scheduler is supported, use direct integration. Schedulers supported by direct integration include SLURM, PBS Pro, Torque, LSF, and HPC Server.</p>	<ul style="list-style-type: none"> • “Configure for HPC Pack” on page 3-31 • “Configure for Slurm, PBS Pro, Platform LSF, TORQUE” on page 3-37 • “Configure a Hadoop Cluster” on page 3-57
<p>Use the generic scheduler interface if:</p> <ul style="list-style-type: none"> • Interfacing MATLAB with third-party schedulers not already supported by direct integration. Schedulers supported by direct integration include SLURM, PBS Pro, Torque, LSF, and HPC Server. • Interfacing MATLAB and third-party schedulers that do not have a shared file system between the MATLAB client and the cluster nodes. • Using a MATLAB client machine that does not have the third-party scheduler’s utilities installed. 	<ul style="list-style-type: none"> • “Configure Using the Generic Scheduler Interface” on page 3-44

See Also

“Running Code on Clusters and Clouds”

Integrate MATLAB Job Scheduler for Network License Manager

If you do not have an existing scheduler in your cluster, follow these instructions to integrate the MATLAB Job Scheduler, which is provided with MATLAB Parallel Server. If you already have a cluster with a scheduler, see: “Integrate MATLAB with Third-Party Schedulers” on page 3-2.

After you integrate MATLAB with a scheduler, you can access workers in your cluster from a desktop MATLAB client session with Parallel Computing Toolbox.

The setup in these steps uses the network license manager.

Activate Your MATLAB Parallel Server License

To install MATLAB Parallel Server, you must activate your license. To activate your MATLAB Parallel Server license:

- 1 Navigate to <https://www.mathworks.com/licensecenter>.
- 2 Log into the Administrator’s MathWorks Account.
- 3 Select your MATLAB Parallel Server license, and click the **Install and Activate** tab.
- 4 At the rightmost side, under **RELATED TASKS**, select **Activate to Retrieve License File**.
- 5 Fill in the requested information. This information must refer to the machine that hosts the license manager. In these instructions, it is the head node. For more information, see “Install Software on the Head Node” on page 3-7.
- 6 After filling in the information, download or email the License File and copy the File Installation Key. These are used later in the process.

Note Activation is not necessary for trials. Contact your sales representative to obtain the License File and the File Installation Key.

Get the Installation Files

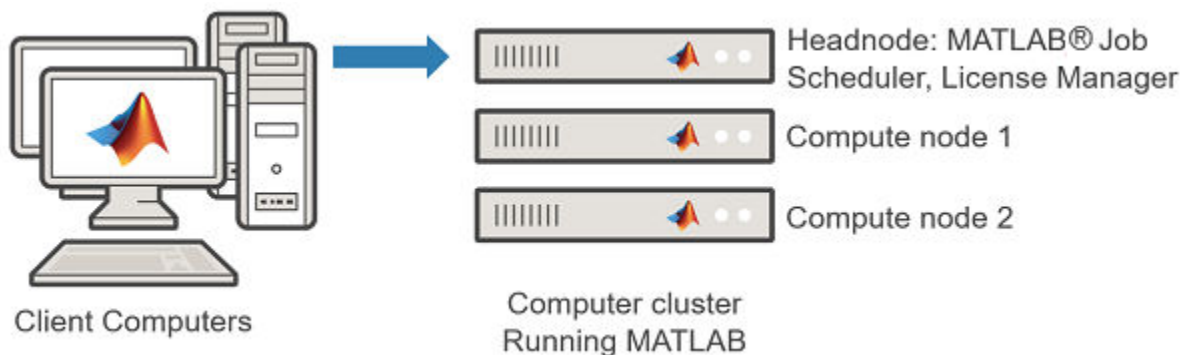
To save time and eliminate the need for the installer-based download process for each computer in your cluster, download the installation files prior to installation. Doing so

facilitates installation in a large number of machines. If you have access to an Administrator's account for your license, you can use the installer to download files without installing them. If not, contact the administrator of your license to obtain a copy of the installation files. For more information, see "Download Products Without Installation" (Installation, Licensing, and Activation). When using the installer to download the files, choose the following options:

- Select the appropriate operating system for the cluster machines.
- Select all products for download. MATLAB Parallel Server cannot run jobs requiring products that are not installed.

There are three server-side components of MATLAB Parallel Server:

- 1 The license manager, which hosts the MATLAB Parallel Server license used by each worker. For more information, see "Install Software on the Head Node" on page 3-7.
- 2 The MATLAB Job Scheduler, which runs on the head node and manages jobs on your cluster. For more information, see "Install Software on the Head Node" on page 3-7.
- 3 MATLAB Parallel Server, which runs on the compute nodes. For more information, see "Install Software on Worker Nodes" on page 3-8.



Install Software on the Head Node

Use the offline installer from the previous step. For more information on the offline installation procedure, see "Install Products Offline" (Installation, Licensing, and Activation).

- 1 Choose a computer to host the license manager and the MATLAB Job Scheduler. This computer is your head node.
- 2 Start the MATLAB installer from the installation files acquired in “Get the Installation Files” on page 3-6.
- 3 Select **Use a File Installation Key**, and continue through the prompts.
- 4 Select all products, including the license manager.
- 5 In the License File step, browse to your `license.lic` file (obtained from “Activate Your MATLAB Parallel Server License” on page 3-6).
- 6 Start the license manager.

This license does not allow you to run MATLAB from the worker nodes or the head node. If you want to test the installation, complete the following setup, and follow the steps in “Connect the MATLAB Client to the MATLAB Parallel Server Cluster” on page 3-11.

Install Software on Worker Nodes

This procedure is similar to “Install Software on the Head Node” on page 3-7. For more information on the offline installation procedure, see “Install Products Offline” (Installation, Licensing, and Activation).

- 1 Start the MATLAB installer from the installation files acquired in “Get the Installation Files” on page 3-6.
- 2 Select **Use a File Installation Key** and continue through the prompts.
- 3 Select all products. However, do not install the license manager.
- 4 Use the `license.dat` file from the head node. You can obtain this file from the `matlabroot/etc` folder, where `matlabroot` is the MATLAB installation folder.

For best performance, install locally on each node. However, you can also install in a network share location. To save time, you can perform a noninteractive installation (silent installation) on the worker nodes. For instructions, see “Install Noninteractively (Silent Installation)” (Installation, Licensing, and Activation).

Configure the MATLAB Job Scheduler with Admin Center

The MATLAB Job Scheduler is a scheduler that ships with MATLAB Parallel Server. The MATLAB Job Scheduler is intended primarily for small-to-medium-sized clusters that run only MATLAB jobs. The scheduler interface is a high-level abstraction that lets you submit

jobs to your computation resources, so you do not have to deal with differences in operating systems and environments.

- 1** On the head node, start Admin Center. Browse to `matlabroot/toolbox/distcomp/bin` and execute the file named `admincenter`. `matlabroot` is the MATLAB installation folder.
- 2** Click **Add or Find**, and specify the computers that you are using as your head node and worker nodes.
- 3** Progress through the prompts, and confirm to start the mjs service. If necessary, manually start the mjs service using the command-line interface. For more information, see “Use the Command-Line Interface (Windows)” on page 3-23 or “Use the Command-Line Interface (UNIX)” on page 3-26.
- 4** In the MATLAB Job Scheduler section, click **Start**. Specify a name for your MATLAB Job Scheduler, and select the head node from the dropdown list.
- 5** To add the MATLAB Parallel Server workers, click **Start** in the **Workers** section of the Admin Center.
 - a** Select the computers to host the workers.
 - b** Select the number of workers per computer.
- 6** To verify your configuration, review worker status in the **Workers** section.
- 7** To troubleshoot issues, click **Test Connectivity** in the **Host** section.
- 8** If you are using UNIX, configure the mjs service to start automatically at start time. For instructions, see “Start the mjs Service, MATLAB Job Scheduler, and Workers (Command-Line)” on page 3-23.

The following screenshot shows a final setup in Admin Center:

3 Product Installation

Admin Center
_ □ ×

File Hosts Scheduler Workers Help

Hosts ?

	Host			MJS Service		MATLAB Job Sche...	Workers
Add or Find...	Hostname	Reachable	Cores	Status	Up Since	Name	Count
Start MJS Service...	ComputeNodeHostname	● yes	6	● running	2019-01-03 15:55		4
Stop MJS Service...	HeadNodeHostname	● yes	6	● running	2019-01-03 15:53	myJobManager	0
Test Connectivity...							

MATLAB Job Scheduler ?

	Name	Hostname	Status	Up Since	Workers
Start...	myJobManager	HeadNodeHostname	● running	2019-01-03 16:03	4
Stop...					
Resume					

Workers ?

	Worker				MATLAB Job Scheduler		
Start...	Name	Hostname	Status	Up Since	Connection	Name	Hostname
Stop...	Worker1	ComputeNode...	● idle	2019-01-03 16:03	● connected	myJobManager	HeadNodeHos...
Resume	Worker2	ComputeNode...	● idle	2019-01-03 16:04	● connected	myJobManager	HeadNodeHos...
	Worker3	ComputeNode...	● idle	2019-01-03 16:04	● connected	myJobManager	HeadNodeHos...
	Worker4	ComputeNode...	● idle	2019-01-03 16:04	● connected	myJobManager	HeadNodeHos...

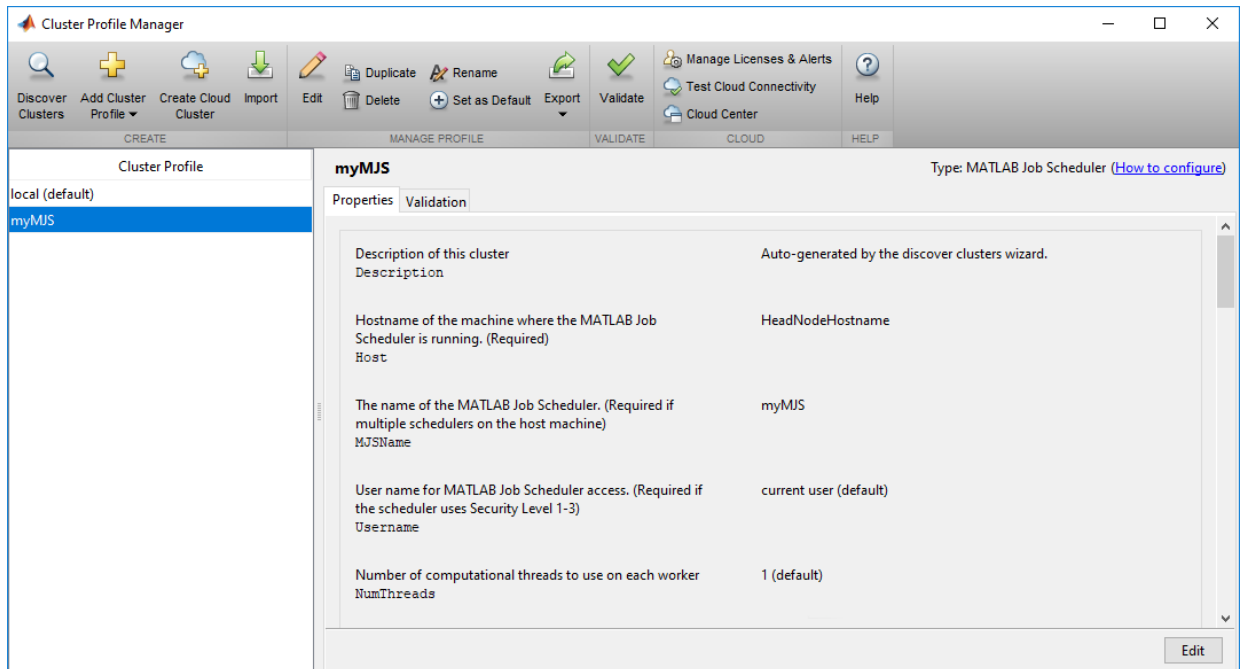
Last updated: 03/01/19 16:04 Update Update Now

Note If you need more help during the configuration, such as your cluster requires firewall configuration or you want to set up multiple mjs installations, see this more detailed guide: “Configure Advanced Options for MATLAB Job Scheduler Integration” on page 3-19.

Connect the MATLAB Client to the MATLAB Parallel Server Cluster

To use MATLAB Parallel Server, you must use a client computer running MATLAB and Parallel Computing Toolbox. In the MATLAB toolstrip, use **Parallel > Discover Clusters** and follow the instructions to automatically discover and set up your cluster. Alternatively, you can configure it manually as follows:

- 1 In MATLAB, on the **Home** tab, select the **Parallel** menu. Select **Create and Manage Clusters**.
- 2 Click **Add Cluster Profile > MATLAB Job Scheduler**.
- 3 To modify the name of the MATLAB Job Scheduler profile, double click the profile name.
- 4 To edit the profile, select it and click **Edit** in the toolbar.
- 5 In the **Host** field, enter the host name of the head node.
- 6 Click **Done**. The following image shows an MATLAB Job Scheduler cluster profile after configuration:



7 To make this profile the default, select **Set as Default**.

8 **Validate** the cluster profile.

If validation of your cluster is successful, your MATLAB session can now submit jobs to the MATLAB Parallel Server cluster.

Note If your validation does not pass, contact the MathWorks install support team.

To configure more advanced options for your cluster, see “MATLAB Job Scheduler Cluster Customization”. For example, you can set the security of the cluster in “Set MATLAB Job Scheduler Cluster Security” on page 2-17. After you finish your configuration, try some examples of cluster workflows in “Running Code on Clusters and Clouds”.

See Also

Related Examples

- “MATLAB Job Scheduler Cluster Customization”
- “Set MATLAB Job Scheduler Cluster Security” on page 2-17

Integrate MATLAB Job Scheduler for Online Licensing

If you do not have an existing scheduler in your cluster, follow these instructions to integrate the MATLAB Job Scheduler, which is provided with MATLAB Parallel Server. If you already have a cluster with a scheduler, see: “Integrate MATLAB with Third-Party Schedulers” on page 3-2.

After you integrate MATLAB with a scheduler, you can access workers in your cluster from a desktop MATLAB client session with Parallel Computing Toolbox.

The setup in these steps uses online licensing.

Set Up License and Users

Set License Type

To install MATLAB Parallel Server using online licensing, you must verify your license type.

- 1 In your browser, go to License Center and log in with your MathWorks Administrator Account.
- 2 In your MathWorks Account, click on the MATLAB Parallel Server license that you plan to use
- 3 On the **Install and Activate** tab, look for **License Manager**: followed by the license manager type currently assigned to this license.
 - If the license manager is already the one you want, then you do not need to do anything. Go to “Add Licensed End Users” on page 3-13.
 - To change the license manager, click the pencil icon and follow the onscreen instructions. When you have finished, go to “Add Licensed End Users” on page 3-13.

Add Licensed End Users

For online licensing to know which users have permission to check out a license, you have to define a list of allowed users for that license. Follow these steps to control which users can access worker licenses.

- 1 If you are not already logged in as an administrator, go to License Center and log in with your MathWorks Administrator Account.

- 2 Click your MATLAB Parallel Server license, and then click **Manage Users**. Any users already associated with this license appear in the list.
- 3 Click **Add User** to add a user to the list.

Add User

A MathWorks Account will be automatically created for the user, if one does not already exist.

Email Address:

First Name:

Last Name:

Country:

Download Software

- 4 Fill in the requested information. Provide the user's email address, first and last names, and their country. Click **Add User**. Note that if the specified email address does not correspond to an existing MathWorks Account, a new account is created for that user.
- 5 Add as many end users as necessary. When you have finished adding users, go to "Get the Installation Files" on page 3-14.

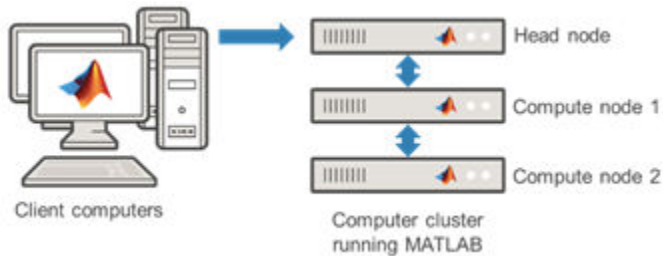
Get the Installation Files

To save time and eliminate the need for the installer-based download process for each computer in your cluster, download the installation files prior to installation. Doing so facilitates installation in a large number of machines. If you have access to an Administrator's account for your license, you can use the installer to download files without installing them. If not, contact the administrator of your license to obtain a copy of the installation files. For more information, see "Download Products Without Installation" (Installation, Licensing, and Activation). When using the installer to download the files, choose the following options:

- Select the appropriate operating system for the cluster machines.
- Select all products for download. MATLAB Parallel Server cannot run jobs requiring products that are not installed.

There are two server-side components of MATLAB Parallel Server:

- 1 The MATLAB Job Scheduler, which runs on the head node and manages jobs on your cluster. For more information, see “Install Software on All Nodes” on page 3-15.
- 2 MATLAB Parallel Server, which runs on the compute nodes. For more information, see “Install Software on All Nodes” on page 3-15.



Install Software on All Nodes

For each computer in the cluster:

- 1 Start the MATLAB installer from the installation files acquired in “Get the Installation Files” on page 3-14.
- 2 Select **Log in with a MathWorks Account** and continue through the prompts.
- 3 Select all products, and exclude the license manager.
- 4 After the installation completes, update the `mjs_def` file in `matlabroot/toolbox/distcomp/bin`. Uncomment and set:
 - Unix: `USE_ONLINE_LICENSING="true"`
 - Windows: `USE_ONLINE_LICENSING=true`

For best performance, install locally on each node. However, you can also install in a network share location. For your convenience, you can perform a noninteractive installation (silent installation) on the worker nodes. For instructions, see “Install Noninteractively (Silent Installation)” (Installation, Licensing, and Activation).

Configure the MATLAB Job Scheduler with Admin Center

The MATLAB Job Scheduler is a scheduler that ships with MATLAB Parallel Server. The MATLAB Job Scheduler is intended primarily for small-to-medium-sized clusters that run only MATLAB jobs. The scheduler interface is a high-level abstraction that lets you submit jobs to your computation resources, so you do not have to deal with differences in operating systems and environments.

- 1** On the head node, start Admin Center. Browse to `matlabroot/toolbox/distcomp/bin` and execute the file named `admincenter`. `matlabroot` is the MATLAB installation folder.
- 2** Click **Add or Find**, and specify the computers that you are using as your head node and worker nodes.
- 3** Progress through the prompts, and confirm to start the mjs service. If necessary, manually start the mjs service using the command-line interface. For more information, see “Use the Command-Line Interface (Windows)” on page 3-23 or “Use the Command-Line Interface (UNIX)” on page 3-26.
- 4** In the MATLAB Job Scheduler section, click **Start**. Specify a name for your MATLAB Job Scheduler, and select the head node from the dropdown list.
- 5** To add the MATLAB Parallel Server workers, click **Start** in the **Workers** section of the Admin Center.
 - a** Select the computers to host the workers.
 - b** Select the number of workers per computer.
- 6** To verify your configuration, review worker status in the **Workers** section.
- 7** To troubleshoot issues, click **Test Connectivity** in the **Host** section.
- 8** If you are using UNIX, configure the mjs service to start automatically at start time. For instructions, see “Start the mjs Service, MATLAB Job Scheduler, and Workers (Command-Line)” on page 3-23.

The following screenshot shows a final setup in Admin Center:

Admin Center

File Hosts Scheduler Workers Help

Hosts

Add or Find...	Host			MJS Service		MATLAB Job Sche...	Workers
	Hostname	Reachable	Cores	Status	Up Since	Name	Count
Start MJS Service...	ComputeNodeHostname	yes	6	running	2019-01-03 15:55		4
Stop MJS Service...	HeadNodeHostname	yes	6	running	2019-01-03 15:53	myJobManager	0
Test Connectivity...							

MATLAB Job Scheduler

Start...	Name	Hostname	Status	Up Since	Workers
Stop...	myJobManager	HeadNodeHostname	running	2019-01-03 16:03	4
Resume					

Workers

Start...	Worker				MATLAB Job Scheduler		
	Name	Hostname	Status	Up Since	Connection	Name	Hostname
Stop...	Worker1	ComputeNode...	idle	2019-01-03 16:03	connected	myJobManager	HeadNodeHos...
Resume	Worker2	ComputeNode...	idle	2019-01-03 16:04	connected	myJobManager	HeadNodeHos...
	Worker3	ComputeNode...	idle	2019-01-03 16:04	connected	myJobManager	HeadNodeHos...
	Worker4	ComputeNode...	idle	2019-01-03 16:04	connected	myJobManager	HeadNodeHos...

Last updated: 03/01/19 16:04

Update every 2 minutes

Note If you need more help during the configuration, such as your cluster requires firewall configuration or you want to set up multiple mjs installations, see this more detailed guide: “Configure Advanced Options for MATLAB Job Scheduler Integration” on page 3-19.

Connect the MATLAB Client to the MATLAB Parallel Server Cluster

To use MATLAB Parallel Server, you must use a client computer running MATLAB and Parallel Computing Toolbox. In the MATLAB toolstrip, use **Parallel > Discover Clusters** and follow the instructions to automatically discover and set up your cluster. Alternatively, you can configure it manually as follows:

- 1 In MATLAB, on the **Home** tab, select the **Parallel** menu. Select **Create and Manage Clusters**.
- 2 Click **Add Cluster Profile > MATLAB Job Scheduler**.
 - After the MATLAB Job Scheduler profile has been created, click **Edit**.
 - Update the hostname of the head node.
 - Update the License number.
 - Click **Done**, and select **Set as Default** (optional) .

If validation of your cluster is successful, your MATLAB session can now submit jobs to the MATLAB Parallel Server cluster.

Note If your validation does not pass, contact the MathWorks install support team.

To configure more advanced options for your cluster, see “MATLAB Job Scheduler Cluster Customization”. For example, you can set the security of the cluster in “Set MATLAB Job Scheduler Cluster Security” on page 2-17. After you finish your configuration, try some examples of cluster workflows in “Running Code on Clusters and Clouds”.

See Also

Related Examples

- “MATLAB Job Scheduler Cluster Customization”
- “Set MATLAB Job Scheduler Cluster Security” on page 2-17

Configure Advanced Options for MATLAB Job Scheduler Integration

In this section...

“Run Multiple MATLAB Parallel Server Versions” on page 3-19

“Set Up Windows Cluster Hosts” on page 3-20

“Configure Firewalls on Server” on page 3-21

“Stop mjs Services of Old Installation” on page 3-22

“Set the MATLAB Job Scheduler Security Level” on page 3-23

“Start the mjs Service, MATLAB Job Scheduler, and Workers (Command-Line)” on page 3-23

“Install the mjs Service to Start Automatically at Boot Time (UNIX)” on page 3-28

“Validate Installation with MATLAB Job Scheduler” on page 3-30

Follow these instructions to configure advanced options during integration of MATLAB Job Scheduler with your cluster.

Note If this is the first time you integrate MATLAB Job Scheduler, see the following for the most common configuration options: “Integrate MATLAB Job Scheduler for Network License Manager” on page 3-6.

In the following instructions, *matlabroot* refers to the location of your installed MATLAB Parallel Server software. Where you see this term used in the instructions that follow, substitute the path to your location.

Run Multiple MATLAB Parallel Server Versions

You can upgrade your MATLAB Job Scheduler clusters and continue to use the R2016a release onwards of Parallel Computing Toolbox on your MATLAB desktop client to connect to it. To take advantage of this backward compatibility feature:

- 1 Install the latest version of MATLAB Parallel Server on your cluster. You must use this version to run MATLAB Job Scheduler on your cluster.

- 2 Install MATLAB Parallel Server for each release that you want to support in the cluster. For example, to use R2016a and R2016b with your cluster, install both the R2016a and R2016b releases of MATLAB Parallel Server.
- 3 Configure MATLAB Job Scheduler with the location of these installations. In the `mjs_def` configuration file, specify the location of each installation of MATLAB Parallel Server in the `MJS_ADDITIONAL_MATLABROOTS` variable. You can find this file in `matlabroot/toolbox/distcomp/bin` for Linux (`mjs_def.sh`) and Windows (`mjs_def.bat`). For more information, see `mjs`.

With this configuration, the MATLAB Job Scheduler allows MATLAB clients from the installed releases to submit jobs to the cluster. The MATLAB Job Scheduler dynamically starts the right version of the MATLAB worker to run the job.

Set Up Windows Cluster Hosts

If this is the first installation of MATLAB Parallel Server on a cluster of Windows machines, you need to configure these hosts for job communications.

Note If you do not have a Windows cluster, or if you have already installed a previous version of MATLAB Parallel Server on your Windows cluster, you can skip this step.

Configure Windows Firewalls on the Client

If you are using Windows firewalls on your cluster nodes,

- 1 Log in as a user with administrator privileges.
- 2 Execute the following in a DOS command window.

```
matlabroot\toolbox\distcomp\bin\addMatlabToWindowsFirewall.bat
```

This command adds MATLAB as an allowed program. If you are using other firewalls, you must configure them for similar accommodation.

Configure Windows User Access for mjs

The user that `mjs` runs as requires access to the cluster MATLAB installation location. By default, `mjs` runs as the user `LocalSystem`. If your network allows `LocalSystem` to access the install location, you can skip this step. (If you are not sure of your network configuration and the access provided for `LocalSystem`, contact the MathWorks install support team.)

Note If LocalSystem cannot access the install location, you must run mjs as a different user.

You can set a different user with these steps:

- 1 With any standard text editor (such as WordPad) open the `mjs_def` file found at:

```
matlabroot\toolbox\distcomp\bin\mjs_def.bat
```

- 2 Find the line for setting the MJSUSER parameter, and provide a value in the form `domain\username`:

```
set MJSUSER=mydomain\myusername
```

- 3 Provide the user password by setting the MJSPASS parameter:

```
set MJSPASS=password
```

- 4 Save the file.

Configure Firewalls on Server

The `mjs` service uses as many ports as required, starting with `BASE_PORT`. By default, `BASE_PORT` is 27350.

If you use a machine that runs a total of `nJ` job managers and `nW` workers, the `mjs` service reserves a total of $6+2*nJ+4*nW$ consecutive ports for its own use. All job managers and workers, even those on different hosts, that are going to work together must use the same base port. Otherwise the job managers and workers will not be able to contact each other. In addition, MPI communication occurs on ports starting at `BASE_PORT+1000` and use $2*nW$ consecutive ports.

For example, if you use a machine with 1 job manager and 16 workers, then you need the following ranges of ports to be open:

- 27350 – 27422 for the `mjs` service.
- 28350 – 28382 for MPI communication.

To connect from MATLAB to a cluster with a non-default `BASE_PORT`, you must append the value of `BASE_PORT` to the 'Host' property in the MATLAB Job Scheduler cluster profile. You must do this in the form `Hostname:BASE_PORT`, for example `myMJSHost:44001`.

Stop mjs Services of Old Installation

If you have an older version of MATLAB Parallel Server running on your cluster nodes, you should stop the mjs services before starting the services of the new installation.

Stop mjs on Windows

- 1 Open a DOS command window with the necessary privileges:

- a If you are using Windows 7 or Windows Vista™, you must run the command window with administrator privileges. Click the Windows menu **Start > (All) Programs > Accessories**; then right-click **Command Window**, and select **Run as Administrator**. This option is available only if you are running User Account Control (UAC).
- b If you are using Windows XP, open a DOS command window by selecting the Windows menu **Start > Run**, then in the **Open** field, type

```
cmd
```

- 2 In the command window, navigate to the folder of the old installation that contains the control scripts.

```
cd oldmatlabroot\toolbox\distcomp\bin
```

- 3 Stop and uninstall the old service and remove its associated files by typing the following command.

```
mjs uninstall -clean
```

In releases before R2019a, the service is called mdce. Type the following command instead.

```
mdce uninstall -clean
```

Note Using the `-clean` flag permanently removes all existing job data. Be sure this data is no longer needed before removing it.

- 4 Repeat the instructions of this step on all worker nodes.

Stop mjs on UNIX

- 1 Log in as root. If you cannot log in as root, you must alter the following parameters in the `oldmatlabroot/toolbox/distcomp/bin/mjs_def.sh` file to point to a folder for which you have write privileges: CHECKPOINTBASE, LOGBASE, PIDBASE, and

LOCKBASE if applicable. In releases before R2019a, this file is *oldmatlabroot/toolbox/distcomp/bin/mdce_def.sh* instead.

- 2 On each cluster node**, stop the mjs service and remove its associated files by typing the commands:

```
cd oldmatlabroot/toolbox/distcomp/bin
./mjs stop -clean
```

In releases before R2019a, the service is called mdce. Type the following command instead.

```
cd oldmatlabroot/toolbox/distcomp/bin
./mdce stop -clean
```

Note Using the `-clean` flag permanently removes all existing job data. Be sure this data is no longer needed before removing it.

Set the MATLAB Job Scheduler Security Level

Before starting the mjs service on your cluster nodes, set a security level. For instructions, see “Set the Security Level” on page 2-17. For additional security considerations, see “Set MATLAB Job Scheduler Cluster Security” on page 2-17.

Start the mjs Service, MATLAB Job Scheduler, and Workers (Command-Line)

You can start MATLAB Job Scheduler using a graphical interface or the command line. For instructions on how to use the graphical interface, see “Configure the MATLAB Job Scheduler with Admin Center” on page 3-8. To use the graphical interface, Admin Center, you must run it on a computer that has direct network connectivity to all the nodes of your cluster. If you cannot run Admin Center on such a computer, you must use the command-line interface. For instructions on how to use the command-line interface, follow the next steps.

Use the Command-Line Interface (Windows)

1 Start the mjs Service

You must install the mjs service on all nodes (head node and worker nodes). Begin on the head node.

- a Open a DOS command window with the necessary privileges:
 - i If you are using Windows or Windows Vista, you must run the command window with administrator privileges. Click the Windows menu **Start > (All) Programs > Accessories**; then right-click **Command Window**, and select **Run as Administrator**. This option is available only if you are running User Account Control (UAC).
 - ii If you are using Windows XP, open a DOS command window by selecting the Windows menu **Start > Run**, then in the **Open** field, type:

```
cmd
```

- b In the DOS command window, navigate to the folder with the control scripts:

```
cd matlabroot\toolbox\distcomp\bin
```

- c Install the mjs service by typing the command:

```
mjs install
```

- d Start the mjs service by typing the command:

```
mjs start
```

- e Repeat the instructions of this step on all worker nodes.

As an alternative to items 3-5, you can install and start the mjs service on several nodes remotely from one machine by typing:

```
cd matlabroot\toolbox\distcomp\bin
remotemjs install -remotehost hostA,hostB,hostC . . .
remotemjs start -remotehost hostA,hostB,hostC . . .
```

where *hostA*, *hostB*, *hostC* refers to a list of your host names. Note that there are no spaces between host names, only a comma. If you need to indicate protocol, platform (such as in a mixed environment), or other information, see the help for `remotemjs` by typing:

```
remotemjs -help
```

Once installed, the mjs service starts running each time the machine reboots. The mjs service continues to run until explicitly stopped or uninstalled, regardless of whether a MATLAB Job Scheduler or worker session is running.

2 Start the MATLAB Job Scheduler

To start the MATLAB Job Scheduler, enter the following commands in a DOS command window. You do not have to be at the machine on which the MATLAB Job Scheduler runs, as long as you have access to the MATLAB Parallel Server installation.

- a** In your DOS command window, navigate to the folder with the startup scripts:

```
cd matlabroot\toolbox\distcomp\bin
```
- b** Start the MATLAB Job Scheduler, using any unique text you want for the name <MyMJS>:

```
startjobmanager -name <MyMJS> -remotehost <MATLAB Job Scheduler host name> -v
```
- c** Verify that the MATLAB Job Scheduler is running on the intended host.

```
nodestatus -remotehost <MATLAB Job Scheduler host name>
```

Note If you are executing `startjobmanager` on the host where the MATLAB Job Scheduler runs, you do not need to specify the `-remotehost` flag.

If you have more than one MATLAB Job Scheduler on your cluster, each must have a unique name.

3 Start the Workers

Note Before you can start a worker on a machine, the `mjs` service must already be running on that machine. If you are using the network license manager, it must be running on the network.

For each node used as a worker, enter the following commands in a DOS command window. You do not have to be at the machines where the MATLAB workers will run, as long as you have access to the MATLAB Parallel Server installation.

- a** Navigate to the folder with the startup scripts:

```
cd matlabroot\toolbox\distcomp\bin
```
- b** Start the workers on each node, using the text for <MyMJS> that identifies the name of the MATLAB Job Scheduler you want this worker registered with. Enter this text on a single line:

```
startworker -jobmanagerhost <MATLAB Job Scheduler host name>  
-jobmanager <MyMJS> -remotehost <worker host name> -v
```

To run more than one worker session on the same node, give each worker a unique name by including the `-name` option on the `startworker` command, and run it for each worker on that node:

```
startworker ... -name <worker1 name>
startworker ... -name <worker2 name>
```

- c Verify that the workers are running.

```
nodestatus -remotehost <worker host name>
```

- d Repeat items 2-3 for all worker nodes.

For more information about `mjs`, MATLAB Job Scheduler, and worker processes, such as how to shut them down or customize them, see “MATLAB Job Scheduler Cluster Customization”.

Use the Command-Line Interface (UNIX)

1 Start the `mjs` Service

On each cluster node, start the `mjs` service by typing the commands:

```
cd matlabroot/toolbox/distcomp/bin
./mjs start
```

Alternatively (on Linux, but not Macintosh), you can start the `mjs` service on several nodes remotely from one machine by typing

```
cd matlabroot/toolbox/distcomp/bin
./remotemjs start -remotehost hostA,hostB,hostC . . .
```

where `hostA,hostB,hostC` refers to a list of your host names. Note that there are no spaces between host names, only a comma. If you need to indicate protocol, platform (such as in a mixed environment), or other information, see the help for `remotemjs` by typing

```
./remotemjs -help
```

2 Start the MATLAB Job Scheduler

To start the MATLAB Job Scheduler, enter the following commands. You do not have to be at the machine on which the MATLAB Job Scheduler runs, as long as you have access to the MATLAB Parallel Server installation.

- a** Navigate to the folder with the startup scripts:

```
cd matlabroot/toolbox/distcomp/bin
```

- b** Start the MATLAB Job Scheduler, using any unique text you want for the name <MyMJS>. Enter this text on a single line.

```
./startjobmanager -name <MyMJS> -remotehost <MATLAB Job Scheduler host name> -v
```

- c** Verify that the MATLAB Job Scheduler is running on the intended host:

```
./nodestatus -remotehost <MATLAB Job Scheduler host name>
```

Note If you have more than one MATLAB Job Scheduler on your cluster, each must have a unique name.

3 Start the Workers

Note Before you can start a worker on a machine, the mjs service must already be running on that machine. If you are using the network license manager, it must be running on the network.

For each computer hosting a MATLAB worker, enter the following commands. You do not have to be at the machines where the MATLAB workers run, as long as you have access to the MATLAB Parallel Server installation.

- a** Navigate to the folder with the startup scripts:

```
cd matlabroot/toolbox/distcomp/bin
```

- b** Start the workers on each node, using the text for <MyMJS> that identifies the name of the MATLAB Job Scheduler you want this worker registered with. Enter this text on a single line:

```
./startworker -jobmanagerhost <MATLAB Job Scheduler host name>  
-jobmanager <MyMJS> -remotehost <worker host name> -v
```

To run more than one worker session on the same machine, give each worker a unique name with the `-name` option:

```
./startworker ... -name <worker1>  
./startworker ... -name <worker2>
```

- c** Verify that the workers are running. Repeat this command for each worker node:

```
./nodestatus -remotehost <worker host name>
```

For more information about mjs, MATLAB Job Scheduler, and worker processes, such as how to shut them down or customize them, see “MATLAB Job Scheduler Cluster Customization”.

Install the mjs Service to Start Automatically at Boot Time (UNIX)

Although this step is not required, it is helpful in case of a system crash. Once configured for this, the mjs service starts running each time the machine reboots. The mjs service continues to run until explicitly stopped, regardless of whether a MATLAB Job Scheduler or worker session is running.

You must have root privileges to do this step.

Choose your platform:

- “Debian, Fedora, SUSE, and Red Hat (non-Fedora) Platforms” on page 3-28
- “Macintosh Platform” on page 3-29

Debian, Fedora, SUSE, and Red Hat (non-Fedora) Platforms

On each cluster node, register the mjs service as a known service and configure it to start automatically at system boot time by following these steps:

- 1 Create the following link, if it does not already exist:

```
ln -s matlabroot/toolbox/distcomp/bin/mjs /etc/mjs
```

- 2 Create the following link to the boot script file:

```
ln -s matlabroot/toolbox/distcomp/bin/mjs /etc/init.d/mjs
```

- 3 Set the boot script file permissions:

```
chmod 555 /etc/init.d/mjs
```

- 4 Find your default run level. If you have a SysV linux machine, you can determine the default run level by booting your machine and immediately executing the `$runlevel` command. The second number output is the default run level of your system. If your Linux machine does not support SysV, look in `/etc/inittab` for the default run level.

- 5 When you have determined the run level, create a link in the `rc` folder associated with that run level. For example, if the run level is 5, execute one of the following sets of platform-specific commands.

- Debian and Fedora platforms:

```
cd /etc/rc5.d;
ln -s ../init.d/mjs S99MJS
```

- SUSE platform:

```
cd /etc/init.d/rc5.d;
ln -s ../mjs S99MJS
```

- Red Hat platform (non-Fedora):

```
cd /etc/rc.d/rc5.d;
ln -s ../../init.d/mjs S99MJS
```

Macintosh Platform

On each cluster node, register the `mjs` service as a known service with `launchd`, and configure it to start automatically at system boot time by following these steps:

- 1 Navigate to the toolbox folder and stop the running `mjs` service:

```
cd matlabroot/toolbox/distcomp/bin
sudo ./mjs stop
```

- 2 Create the following link if it does not already exist:

```
sudo mkdir -p /usr/local/sbin/
sudo ln -s matlabroot/toolbox/distcomp/bin/mjs /usr/local/sbin/mjs
```

- 3 Copy the `launchd.plist` file for `mjs` to `/Library/LaunchDaemons`:

```
sudo cp ./util/com.mathworks.mjs.plist /Library/LaunchDaemons
```

- 4 Open the copied `.plist` file in a text editor. Ensure that the leading part of the `StandardOutPath` and `StandardErrorPath` fields match the `LOGBASE` value as defined in the `mjs_def.sh` file. For example, if `LOGBASE` is `/var/log/mjs`, then you must define `StandardOutPath` and `StandardErrorPath` as follows:

```
<key>StandardOutPath</key>
<string>/var/log/mjs/launchctl.stdout</string>
<key>StandardErrorPath</key>
<string>/var/log/mjs/launchctl.stderr</string>
```

- 5 Restart your machine and observe that mjs is running using `nodestatus`:

```
cd matlabroot/toolbox/distcomp/bin
./nodestatus
```

Validate Installation with MATLAB Job Scheduler

To verify that your MATLAB Parallel Server products are installed and configured correctly, create a cluster profile and validate it. For instructions, see “Connect the MATLAB Client to the MATLAB Parallel Server Cluster” on page 3-11. You can specify the number of workers to use when validating your profile, to avoid occupying the whole cluster. If your validation does not pass, contact the MathWorks Install Support Team, or see “Troubleshoot Common Problems” on page 2-22.

After you create a cluster profile, you can make any modifications appropriate for your applications, such as `NumWorkersRange`, `AttachedFiles`, or `AdditionalPaths`. To save your profile for other users, in the Cluster Profile Manager, select the profile and click **Export**, then save your profile to a file in a convenient location. Later, when running the Cluster Profile Manager, other users can import your profile by clicking **Import**. For more information about cluster profiles, see “Discover Clusters and Use Cluster Profiles” (Parallel Computing Toolbox).

See Also

Related Examples

- “Integrate MATLAB Job Scheduler for Network License Manager” on page 3-6

More About

- “MATLAB Job Scheduler Cluster Customization”
- “Troubleshooting in MATLAB Parallel Server”
- “Getting Started with MATLAB Parallel Server”

Configure for HPC Pack

In this section...

“Configure Cluster for Microsoft HPC Pack” on page 3-31

“Configure Client Computer for HPC Pack” on page 3-32

“Validate Installation Using Microsoft HPC Pack” on page 3-33

Configure Cluster for Microsoft HPC Pack

Follow these instructions to configure your MATLAB Parallel Server installation to work with Microsoft HPC Pack or Compute Cluster Server (CCS). In the following instructions, *matlabroot* refers to the MATLAB installation location.

Supported versions: Windows Compute Cluster Server 2003, Windows HPC Server 2008, Windows HPC Server 2008 R2, Microsoft HPC Pack 2012, Microsoft HPC Pack 2012 R2, and Microsoft HPC Pack 2016.

Note If you are using an HPC Pack in a network share installation, the network share location must be in the “Intranet” zone. You might need to adjust the Internet Options for your cluster nodes and add the network share location to the list of Intranet sites.

- 1 Log in on the cluster head node as a user with administrator privileges.
- 2 Open a command window with administrator privileges and run the following file command

```
matlabroot\toolbox\distcomp\bin\MicrosoftHPCServerSetup.bat -cluster
```

This command performs some of the setup required for all machines in the cluster. The location of the MATLAB installation must be the same on every cluster node.

Note If you need to override the script default values, modify the values defined in `MicrosoftHPCServerSetup.xml` before running `MicrosoftHPCServerSetup.bat`. Use the `-def_file` argument to the script when using a `MicrosoftHPCServerSetup.xml` file in a custom location. For example:

```
MicrosoftHPCServerSetup.bat -cluster -def_file <filename>
```

You modify the file only on the node where you actually run the script.

An example of one of the values you might set is for `CLUSTER_NAME`. If you provide a friendly name for the cluster in this parameter, it is recognized by MATLAB's discover clusters feature and displayed in the resulting cluster list.

Configure Client Computer for HPC Pack

This configuring applies to all versions of HPC Pack.

Note If you are using HPC Pack in a network share installation, the network share location must be in the "Intranet" zone. You might need to adjust the Internet Options for your cluster nodes and add the network share location to the list of Intranet sites.

- 1 Open a command window with administrator privileges and run the following file command

```
matlabroot\toolbox\distcomp\bin\MicrosoftHPCServerSetup.bat -client
```

This command performs some of the setup required for a client machine.

Note If you need to override the default values the script, modify the values defined in `MicrosoftHPCServerSetup.xml` before running `MicrosoftHPCServerSetup.bat`. Use the `-def_file` argument to the script when using a `MicrosoftHPCServerSetup.xml` file in a custom location. For example:

```
MicrosoftHPCServerSetup.bat -client -def_file <filename>
```

- 2 To submit jobs or discover the cluster from MATLAB, the Microsoft HPC Pack client utilities must be installed on your MATLAB client machine. If they are not already installed and up to date, ask your system administrator for the correct client utilities to install. The utilities are available from Microsoft download center.

If you have installed multiple versions of the Microsoft HPC Pack client utilities, MATLAB uses the most recent install. To configure MATLAB to use a specific install, set the environment variable 'MATLAB_HPC_SERVER_HOME' to the install location of the client utilities you want to use.

Validate Installation Using Microsoft HPC Pack

This procedure verifies that your parallel computing products are installed and configured correctly for using Microsoft Windows HPC Pack or Compute Cluster Server (CCS).

Step 1: Define a Cluster Profile

In this step you define a cluster profile to use in subsequent steps.

- 1 Start the Cluster Profile Manager from the MATLAB desktop by selecting on the **Home** tab in the **Environment** area **Parallel > Manage Cluster Profiles**.
- 2 Create a new profile in the Cluster Profile Manager by selecting **New > HPC Server**.
- 3 With the new profile selected in the list, click **Rename** and edit the profile name to be HPCtest. Press **Enter**.
- 4 In the Properties tab, provide text for the following fields:
 - a Set the **Description** field to For testing installation with HPC Server.
 - b Set the **NumWorkers** field to the number of workers you want to run the validation tests on, within the limitation of your licensing.
 - c Set the **Host** field to the name of the host on which your scheduler is running. Depending on your network, this might be a simple host name, or it might have to be a fully qualified domain name.

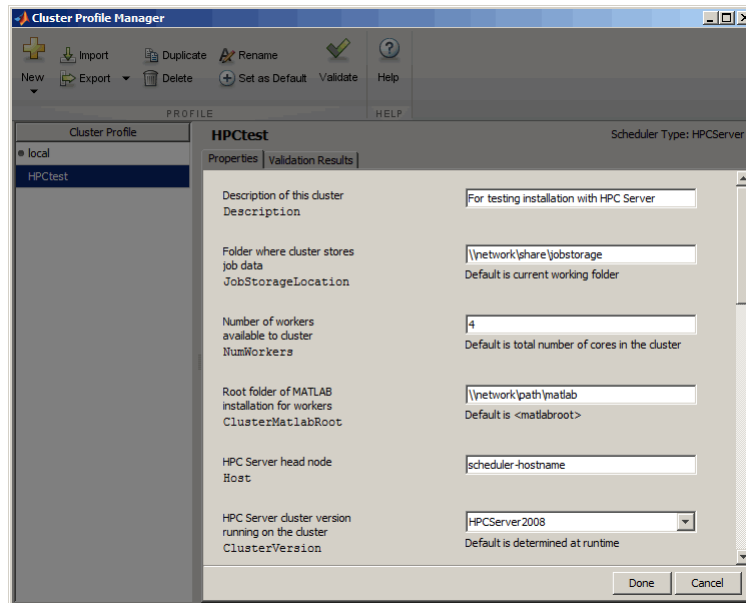
Note: The following four property settings (`JobStorageLocation`, `ClusterMatlabRoot`, `ClusterVersion`, and `UseSOAJobSubmission`) are optional, and need to be set in the profile here only if you did not run `MicrosoftHPCServerSetup.bat` as described in “Configure Cluster for Microsoft HPC Pack” on page 3-31, or if you want to override the setting established by that script.

- d Set the **JobStorageLocation** to the location where you want job and task data to be stored. This must be accessible to all the worker machines.

Note `JobStorageLocation` should not be shared by parallel computing products running different versions; each version on your cluster should have its own `JobStorageLocation`.

- e Set the **ClusterMatlabRoot** to the installation location of the MATLAB to be executed by the worker machines, as determined in Chapter 1 of the installation instructions.
- f Set the **ClusterVersion** field to HPCServer or CCS.
- g If you want to test SOA job submissions on an HPC Server cluster, set **UseSOAJobSubmission** to `true`. If you plan to use SOA job submissions with your cluster, you should test this first without SOA submission, then later return and test it with SOA job submission. The default value is determined at runtime based on your scheduler.

So far, the dialog box should look like the following figure:



- 5 Click **Done** to save your cluster profile.

Step 2: Validate the Configuration

In this step you validate your cluster profile, and thereby your installation. You can specify the number of workers to use when validating your profile. If you do not specify the number of workers in the **Validation** tab, then the validation will attempt to use as many workers as the value specified by the `NumWorkers` property on the **Properties** tab. You

can specify a smaller number of workers to validate your configuration without occupying the whole cluster.

- 1 If it is not already open, start the Cluster Profile Manager from the MATLAB desktop by selecting on the **Home** tab in the **Environment** area **Parallel > Manage Cluster Profiles**.
- 2 Select your cluster profile in the listing.
- 3 Click **Validation** tab.
- 4 Use the checkboxes to choose all tests, or a subset of the validation stages, and specify the number of workers to use when validating your profile.
- 5 Click **Validate**.

The Validation Results tab shows the output. The following figure shows the results of a profile that passed all validation tests.

The screenshot shows the Cluster Profile Manager window. The 'Validation' tab is active, displaying a table of test results. All tests passed. Below the table, the number of workers is set to 'Use default'. The stage details show the validation completed in 0 min 2 sec.

Stage	Status	Description
<input checked="" type="checkbox"/> Cluster connection test (parcluster)	Passed	
<input checked="" type="checkbox"/> Job test (createJob)	Passed	
<input checked="" type="checkbox"/> SPMD job test (createCommunicatingJob)	Passed	Job ran with 6 workers.
<input checked="" type="checkbox"/> Pool job test (createCommunicatingJob)	Passed	Job ran with 6 workers.
<input checked="" type="checkbox"/> Parallel pool test (parpool)	Passed	Parallel pool ran with 6 workers.

Number of workers to use: Use default

STAGE DETAILS

Stage started at 16:40:32. Completed in 0 min 2 sec.

Note If your validation does not pass, contact the MathWorks install support team.

If your validation passed, you now have a valid profile that you can use in other parallel applications. You can make any modifications to your profile appropriate for your applications, such as `NumWorkersRange`, `AttachedFiles`, `AdditionalPaths`, etc.

To save your profile for other users, select the profile and click **Export**, then save your profile to a file in a convenient location. Later, when running the Cluster Profile Manager, other users can import your profile by clicking **Import**.

Configure for Slurm, PBS Pro, Platform LSF, TORQUE

In this section...

“Configure Platform LSF Scheduler on Windows Cluster” on page 3-37

“Configure Windows Firewalls on Client” on page 3-39

“Validate Installation Using a Slurm, LSF, PBS Pro, or TORQUE Scheduler” on page 3-39

Follow these instructions to configure your MATLAB Parallel Server installation to work with Slurm, PBS Pro, Platform LSF, TORQUE.

Note Instead, use the generic scheduler interface for any of the following:

- Any third-party scheduler not listed above (e.g., Sun Grid Engine, GridMP, etc.)
- PBS other than PBS Pro
- A nonshared file system when the client cannot directly submit to the scheduler (e.g., TORQUE on Windows)
- When the MATLAB client machine does not have the third-party scheduler’s utilities installed (for example `sinfo`, `sbatch`, `squeue` and `sacct` for Slurm)
- If you want to use Slurm from a Windows client, or if you do not have a shared file system with your Slurm cluster, or if the Slurm utilities are not accessible on your client, then use the generic scheduler interface.

Configure Platform LSF Scheduler on Windows Cluster

If your cluster is already set up to use `mpiexec` and `smpd`, you can use Parallel Computing Toolbox software with your existing configuration if you are using a compatible MPI implementation library (as defined in `matlabroot\toolbox\distcomp\mpi\mpiLibConf.m`). However, if you do not have `mpiexec` on your cluster and you want to use it, you can use the `mpiexec` software shipped with the parallel computing products.

For further information about `mpiexec` and `smpd`, see the MPICH home page. For user’s guides and installation instructions on that page, select **Documentation > User Docs**.

In the following instructions, `matlabroot` refers to the MATLAB installation location.

To use `mpiexec` to distribute a job, the `smpd` service must be running on all nodes that will be used for running MATLAB workers.

Note The `smpd` executable does not support running from a mapped drive. Use either a local installation, or the full UNC path name to the executable. Microsoft Windows Vista does not support the `smpd` executable on network share installations, so with Vista the installation must be local.

- 1 Log in as a user with administrator privileges.
- 2 Start `smpd` by typing in a DOS command window:

```
matlabroot\bin\win64\smpd -install
```

This command installs the service and starts it. As long as the service remains installed, it will start each time the node boots.

- 3 If this is a worker machine and you did not run the installer on it to install MATLAB Parallel Server software (for example, if you are running MATLAB Parallel Server software from a shared installation), execute the following command in a DOS command window.

```
matlabroot\bin\matlab.bat -install_vcrt
```

This command installs the Microsoft run-time libraries needed for running jobs with your scheduler.

- 4 If you are using Windows firewalls on your cluster nodes, execute the following in a DOS command window.

```
matlabroot\toolbox\distcomp\bin\addMatlabToWindowsFirewall.bat
```

This command adds MATLAB as an allowed program. If you are using other firewalls, you must configure them to make similar accommodation.

- 5 Log in as the user who will be submitting jobs for execution on this node.
- 6 Register this user to use `mpiexec` by typing:

```
matlabroot\bin\win64\mpiexec -register
```

- 7 Repeat steps 5-6 for all users who will run jobs on this machine.
- 8 Repeat all these steps on all Windows nodes in your cluster.

Configure Windows Firewalls on Client

If you are using Windows firewalls on your cluster nodes,

- 1 Log in as a user with administrative privileges.
- 2 Execute the following in a DOS command window.

```
matlabroot\toolbox\distcomp\bin\addMatlabToWindowsFirewall.bat
```

This command adds MATLAB as an allowed program. If you are using other firewalls, you must configure them for similar accommodation.

Validate Installation Using a Slurm, LSF, PBS Pro, or TORQUE Scheduler

This procedure verifies that the parallel computing products are installed and configured correctly on your cluster.

Step 1: Define a Cluster Profile

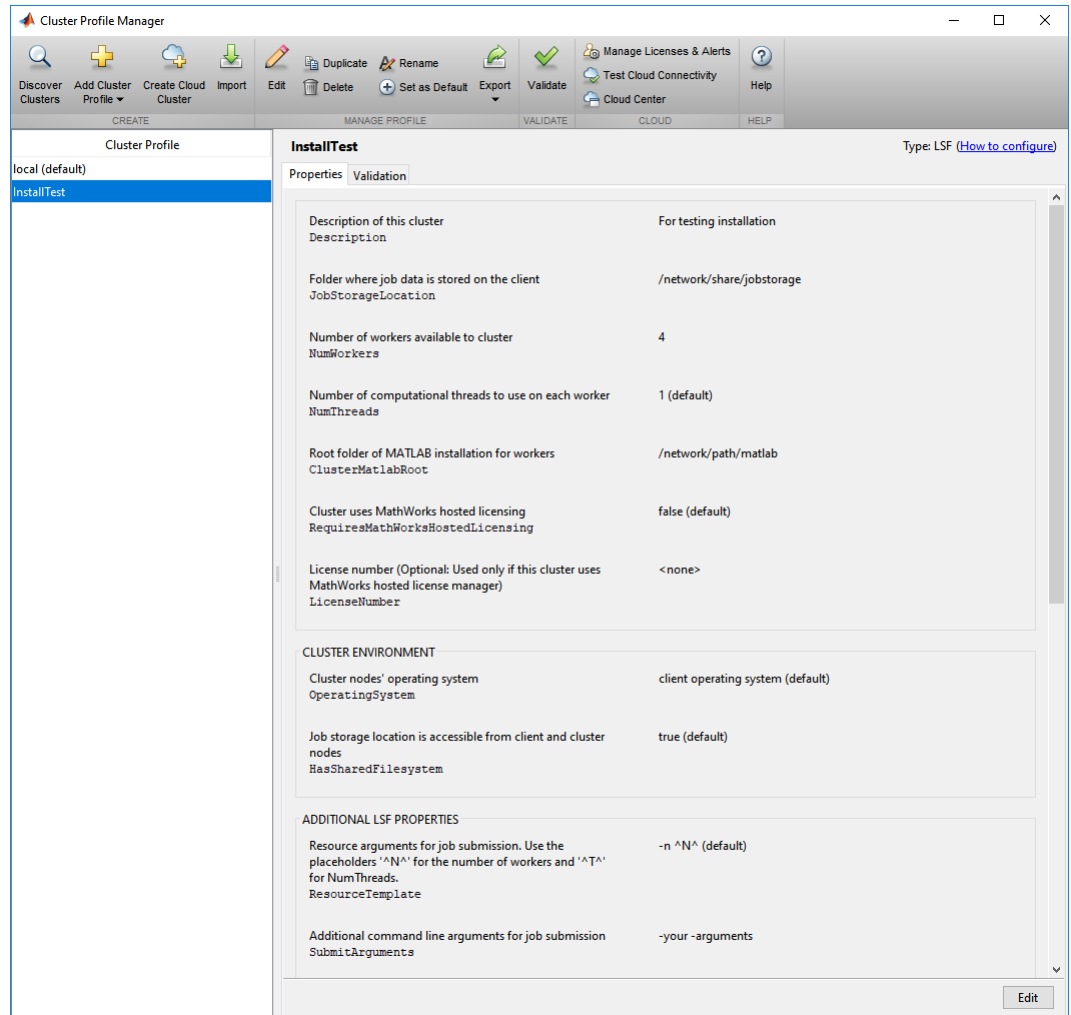
In this step you define a cluster profile to use in subsequent steps.

- 1 Start the Cluster Profile Manager from the MATLAB desktop by selecting on the **Home** tab in the **Environment** area **Parallel > Manage Cluster Profiles**.
- 2 Create a new profile in the Cluster Profile Manager by selecting **New > LSF** (or **Slurm, PBS Pro** or **Torque**, as appropriate).
- 3 With the new profile selected in the list, click **Rename** and edit the profile name to be `InstallTest`. Press **Enter**.
- 4 In the Properties tab, provide settings for the following fields:
 - a Set the **Description** field to `For testing installation`.
 - b Set the **JobStorageLocation** to the location where you want job and task data to be stored (accessible to all the worker machines if you have a shared file system).

Note `JobStorageLocation` should not be shared by parallel computing products running different versions; each version on your cluster should have its own `JobStorageLocation`.

- c** Set the **NumWorkers** field to the number of workers you want to run the validation tests on, within the limitation of your licensing.
- d** Set the **ClusterMatlabRoot** to the installation location of the MATLAB to be executed by the worker machines, as determined in Chapter 1 of the installation instructions.
- e** Set the **SubmitArguments** to include any additional command arguments required by your particular cluster and scheduler.
- f** If you are using LSF[®], set the **OperatingSystem** to the operating system of your worker machines.
- g** Set **HasSharedFilesystem** to indicate if client and workers can share the same data location.

The dialog box should look something like this, or slightly different for PBS Pro or TORQUE schedulers.



5 Click **Done** to save your cluster profile.

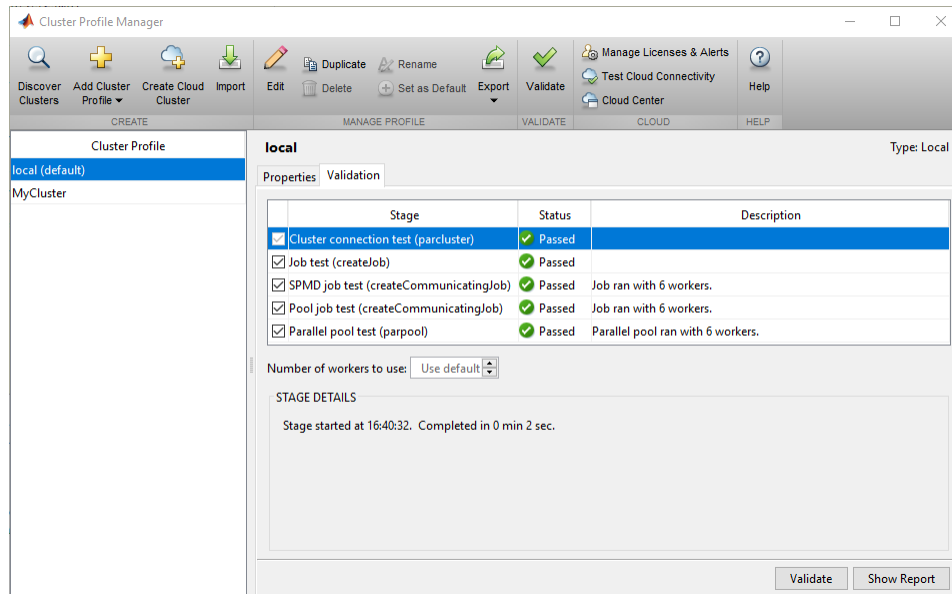
Step 2: Validate the Cluster Profile

In this step you verify your cluster profile, and thereby your installation. You can specify the number of workers to use when validating your profile. If you do not specify the number of workers in the **Validation** tab, then the validation will attempt to use as many workers as the value specified by the NumWorkers property on the **Properties** tab. You

can specify a smaller number of workers to validate your configuration without occupying the whole cluster.

- 1 If it is not already open, start the Cluster Profile Manager from the MATLAB desktop by selecting on the **Home** tab in the **Environment** area **Parallel > Manage Cluster Profiles**.
- 2 Select your cluster profile in the listing.
- 3 Click **Validation** tab.
- 4 Use the checkboxes to choose all tests, or a subset of the validation stages, and specify the number of workers to use when validating your profile.
- 5 Click **Validate**.

The Validation Results tab shows the output. The following figure shows the results of a profile that passed all validation tests.



Note If your validation does not pass, contact the MathWorks install support team.

If your validation passed, you now have a valid profile that you can use in other parallel applications. You can make any modifications to your profile appropriate for your applications, such as `NumWorkersRange`, `AttachedFiles`, `AdditionalPaths`, etc.

To save your profile for other users, select the profile and click **Export**, then save your profile to a file in a convenient location. Later, when running the Cluster Profile Manager, other users can import your profile by clicking **Import**.

Configure Using the Generic Scheduler Interface

The generic scheduler interface provides flexibility to configure the interaction of the MATLAB client, MATLAB workers, and a third-party scheduler. Use the generic scheduler interface when you want complete customization for interfacing MATLAB with your scheduler setup.

You must use the generic scheduler interface when:

- Interfacing MATLAB with third-party schedulers not already supported by direct integration. Schedulers supported by direct integration include SLURM, PBS Pro, Torque, LSF, and HPC Server.
- Interfacing MATLAB and third-party schedulers that do not have a shared file system between the MATLAB client and the cluster nodes.
- Using a MATLAB client machine that does not have the third-party scheduler utilities installed.

Interfacing with Generic Schedulers

The generic scheduler interface provides a means of getting tasks from your Parallel Computing Toolbox client session to your scheduler and cluster nodes. To achieve this, you must provide the generic scheduler with a set of integration scripts. The scripts contain instructions specific to your cluster infrastructure, such as how to communicate with the job scheduler, and how to transfer job and task data to cluster nodes.

Support Scripts

To support usage of the generic scheduler interface, the following third-party schedulers provide integration scripts:

- IBM Platform LSF
- Grid Engine family
- PBS family
- SLURM

Each installer provides scripts for three possible submission modes:

- Shared - When the client can submit directly to the scheduler, and the client and the cluster machines have a shared file system.

- Remote - When the client and cluster machines have a shared file system, but the client machine cannot submit directly to the scheduler, such as when the client utilities of the scheduler are not installed. In this case, a remote host submits commands to the scheduler using the `ssh` protocol.
- Nonshared - When the client and cluster machines do not have a shared file system. This mode uses the `ssh` protocol to submit commands to the scheduler using a remote host, and it uses the `sftp` protocol to copy job and task files to the cluster file system.

Each submission mode has its own subfolder within the installation folder. These subfolders contain a `README` file that provides specific instructions on how to use the scripts. Before using the scripts, decide which submission mode describes your network setup.

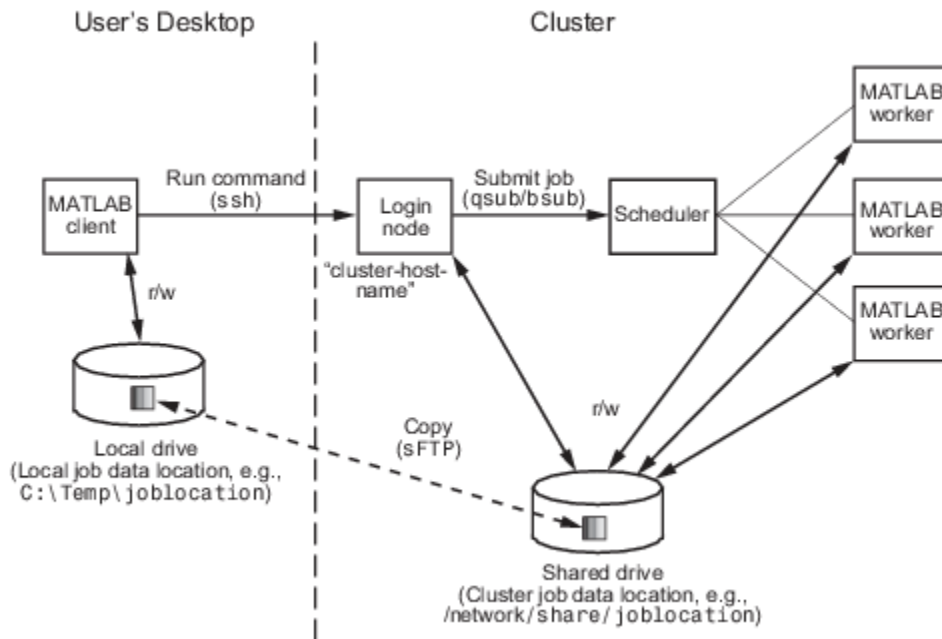
To run the installer, download the appropriate support package for your scheduler, and open it in your MATLAB client. The installer includes a wizard to guide you through creating a cluster profile for your cluster configuration.

If your scheduler or cluster configuration is not supported by one of the support packages, it is recommended that you modify the scripts of one of these packages. For more information on how to write a set of integration scripts for generic schedulers, see “Integration Scripts for Generic Schedulers” (Parallel Computing Toolbox).

Creating a Generic Cluster Profile

Sample Setup for LSF

This example shows how to set up your cluster profile to use the generic scheduler interface. It shows the set up of an LSF scheduler in a network without a shared file system between the client and the cluster machines. The following diagram illustrates the cluster setup:



In this type of configuration, job data is copied from the client host running a Windows operating system to a host on the cluster (cluster login node) running a UNIX® operating system. From the cluster login node, the LSF `bsub` command submits the job to the scheduler. When the job finishes, its output is copied back to the client host.

Requirements

The setup must meet the following conditions:

- The client node and cluster login node must support `ssh` and `sftp`.
- The cluster login node must be able to call the `bsub` command to submit a job to an LSF scheduler. You can find more about this in the `README` file in the `nonshared` subfolder within the installation folder.

Run the LSF Installer

- 1 Download the installer for LSF from [here](#).
- 2 Run the installer by opening the file from within your MATLAB client.

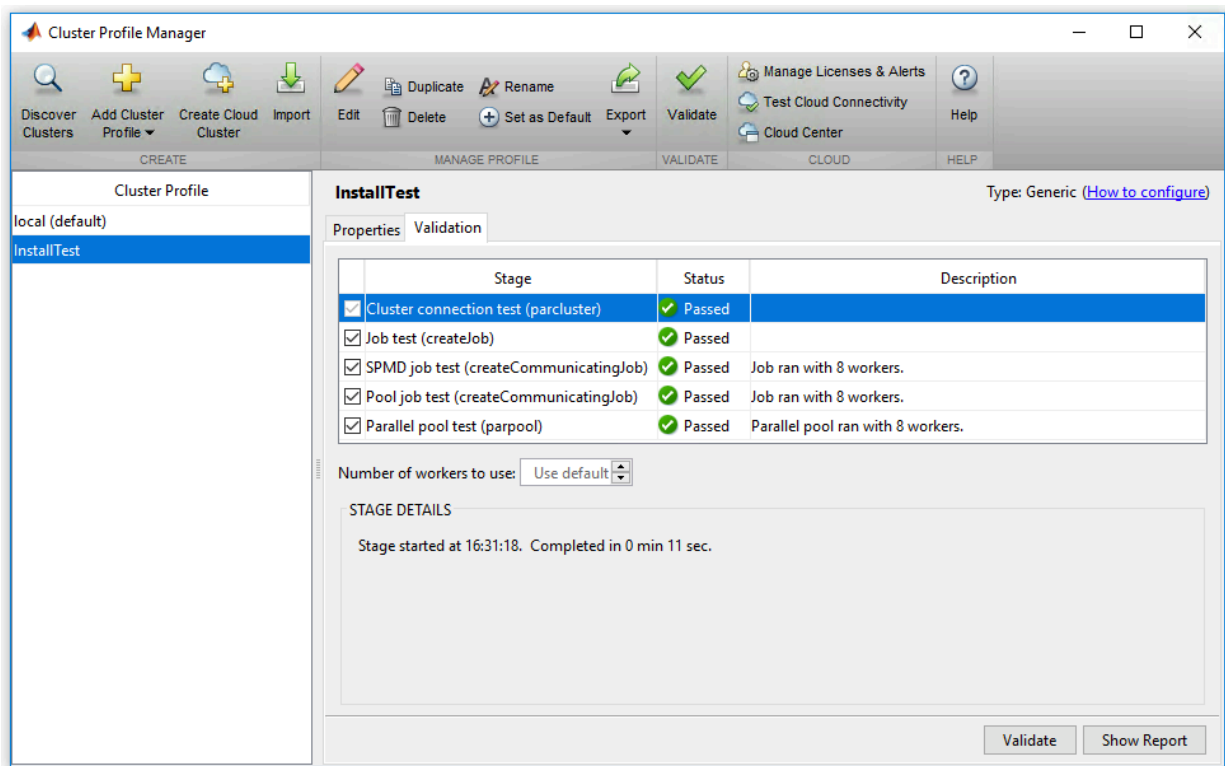
The installer downloads the integration scripts to the `nonshared` subfolder within the installation folder. The installer wizard guides you through the steps to create and validate a cluster profile.

Validate Cluster Profile and Installation

You can specify the number of workers to use when validating your profile. If you do not specify the number of workers in the **Validation** tab, then the validation process attempts to use as many workers as the value specified by the **NumWorkers** property on the **Properties** tab. You can specify a smaller number of workers to validate your configuration without occupying the whole cluster.

- 1** Start the Cluster Profile Manager from the MATLAB desktop. On the **Home** tab, in the **Environment** area, select **Parallel > Manage Cluster Profiles**.
- 2** Select your cluster profile in the listing.
- 3** Click the **Validation** tab.
- 4** Use the checkboxes to choose all tests, or a subset of the validation stages, and specify the number of workers to use when validating your profile.
- 5** Click **Validate**.

The **Validation** results tab shows the output. The following figure shows the results of a profile that passed all validation tests.



Note If your validation fails any stage, contact the MathWorks install support team.

If your validation passes, you have a valid profile that you can use in other parallel applications. You can make any modifications to your profile that are appropriate for your applications, such as **NumWorkersRange**, **AttachedFiles**, or **AdditionalPaths**.

To save your profile for other users, select the profile, and click **Export**. Then save your profile to a file in a convenient location. When running the Cluster Profile Manager, other users can import your profile by clicking **Import**.

To learn how to distribute a generic cluster profile and integration scripts for others to use, see “Distribute a Generic Cluster Profile and Integration Scripts” on page 3-54.

Manually Configure a Cluster Profile

If you want to modify an existing generic cluster profile, you can configure the profile manually. If you are creating the profile for the first time and you are using one of the provided support scripts, use the installer wizard instead: “Run the LSF Installer” on page 3-46. The following steps reproduce manually the configuration performed by the installer. You can modify any of these options depending on your setup.

- 1 Start a MATLAB session on the client host.
- 2 Start the Cluster Profile Manager from the MATLAB desktop. On the **Home** tab, in the **Environment** area, select **Parallel > Manage Cluster Profiles**.
- 3 Create a new profile in the Cluster Profile Manager by selecting **Add > Custom > Generic**.
- 4 With the new profile selected in the list, select **Rename** and change the profile name to `InstallTest`. Press **Enter**.
- 5 In the **Properties** tab, select **Edit** and provide settings for the following fields:
 - a Set the **Description** field to *For testing installation*.
 - b Set the **JobStorageLocation** to the location where you want job and task data to be stored on the client machine (not the cluster location), for example, `C:\Temp\joblocation`.

You must not share **JobStorageLocation** among parallel computing products running different versions. Each version on your cluster must have its own **JobStorageLocation**.

- c Set **NumWorkers** to the number of workers for which you want to test your installation.
- d Set **NumThreads** to the number of threads to use on each worker.
- e Set **ClusterMatlabRoot** to the installation location of MATLAB to run on the worker machines.
- f If the cluster uses online licensing, set **RequiresOnlineLicensing** to true.
- g If you set **RequiresOnlineLicensing** to true, enter your **LicenseNumber**.
- h Set **OperatingSystem** to the operating system of your cluster worker machines.
- i Set **HasSharedFilesystem** to false. This setting indicates that the client node and worker nodes cannot share the same data location.
- j Set the **IntegrationScriptsLocation** to the location of your integration scripts. In this example, the location is the nonshared subfolder within the LSF

installation folder. As part of using the example scripts in nonshared submission mode, set the properties in steps k and l.

- k In the **AdditionalProperties** table, select **Add**. Specify a new property with name `ClusterHost`, value `cluster-host-name`, and type `String`.
 - l In the **AdditionalProperties** table, select **Add**. Specify a new property with name `RemoteJobStorageLocation`, value `/network/share/joblocation`, and type `String`.
- 6 Click **Done** to save your cluster profile changes. The dialog box looks as follows:

InstallTest Type: Generic ([How to configure](#))

Properties Validation

Description of this cluster Description	For testing installation
Folder where job data is stored on the client JobStorageLocation	C:\Temp\joblocation
Number of workers available to cluster NumWorkers	4
Number of computational threads to use on each worker NumThreads	1
Root folder of MATLAB installation for workers ClusterMatlabRoot	/network/installs/MATLAB/R2019a
License number (Optional: Used only if this cluster uses online licensing) LicenseNumber	<none>
Cluster uses online licensing RequiresOnlineLicensing	<none>

CLUSTER ENVIRONMENT

Cluster nodes' operating system OperatingSystem	unix
Job storage location is accessible from client and cluster nodes HasSharedFilesystem	false

SCHEDULER INTEGRATION

Folder containing scheduler integration scripts IntegrationScriptsLocation	C:\ProgramData\MATLAB\SupportPackages\R2019a\parallel\sl...									
Additional properties for integration scripts AdditionalProperties	<table border="1"> <thead> <tr> <th>Name</th> <th>Value</th> <th>Type</th> </tr> </thead> <tbody> <tr> <td>ClusterHost</td> <td>cluster-host-name</td> <td>String</td> </tr> <tr> <td>RemoteJobStorageLoc...</td> <td>/network/share/jobloc...</td> <td>String</td> </tr> </tbody> </table>	Name	Value	Type	ClusterHost	cluster-host-name	String	RemoteJobStorageLoc...	/network/share/jobloc...	String
Name	Value	Type								
ClusterHost	cluster-host-name	String								
RemoteJobStorageLoc...	/network/share/jobloc...	String								

To check that the profile works, perform a validation following the steps in “Validate Cluster Profile and Installation” on page 3-47.

Special Configurations

Depending on your cluster architecture, you might need to perform additional tasks before you connect to your generic scheduler.

Custom MPI builds

You can use an MPI build that differs from the one provided with Parallel Computing Toolbox. For more information about using this option with the generic scheduler interface, see “Use Different MPI Builds on UNIX Systems” on page 2-5.

Run Communicating Jobs with the Grid Engine Family

The sample scripts for Grid Engine family rely on the presence of a `matlab` parallel environment. Parallel environments (PE) are programming environments designed for parallel computing in clusters. To run communicating jobs with MATLAB Parallel Server and a Grid Engine family cluster, you must establish a `matlab` parallel environment.

Create the Parallel Environment

The following steps create the parallel environment, and then make it runnable on all queues. As a best practice, perform these steps on the head node of your cluster. Some steps require administrator access.

- 1 Download and run the installer for Grid Engine from Grid Engine family.
- 2 Navigate to the location of the relevant integration scripts for your submission mode in the installation folder.
- 3 Modify the contents of `matlabpe.template` to use the number of slots you want and the correct location of the `startmatlabpe.sh` and `stopmatlabpe.sh` files. These files can exist in a shared location accessible by all hosts, or you can copy them to the same location on each host. You can also change other values or add additional values to `matlabpe.template` to suit your cluster. For more information, refer to the `sge_pe` documentation provided with your scheduler.
- 4 Add the `matlab` parallel environment, using a shell command such as:

```
qconf -Ap matlabpe.template
```
- 5 Make the `matlab` parallel environment runnable on all queues:

```
qconf -mq all.q
```

This command brings up a text editor for you to make changes. Search for the line `pe_list`, and add `matlab`.

- 6 Ensure you can submit a trivial job to the PE:

```
$ echo "hostname" | qsub -pe matlab 1
```

- 7 Use `qstat` to check that the job runs correctly, and check that the output file contains the name of the host that ran the job. The default file name for the output file is `~/STDIN.o###`, where `###` is the Grid Engine job number.

Note If you change the name of the parallel environment to something other than `matlab`, also change the submit functions.

Configure Firewalls on Windows Cluster

If you are using Windows firewalls on your cluster nodes, you can add MATLAB as an allowed program.

In the following instructions, *matlabroot* refers to the MATLAB installation location.

- 1 Log in as a user with administrative privileges.
- 2 Execute the following script in a DOS command window:

```
matlabroot\toolbox\distcomp\bin\addMatlabToWindowsFirewall.bat
```

If you are using other firewalls, you must configure these separately to add MATLAB as an allowed program.

See Also

Related Examples

- “Integration Scripts for Generic Schedulers” (Parallel Computing Toolbox)

Distribute a Generic Cluster Profile and Integration Scripts

You can distribute a Generic cluster profile and integration scripts for others to use. If necessary, you can create a Generic cluster profile and integration scripts as follows:

- 1 Install the appropriate support package for your third-party scheduler (see “Support Scripts” on page 3-44).
- 2 Use the Generic Profile Wizard to create a Generic cluster profile with the default MATLAB integration scripts.

Decide How Users Access the Integration Scripts

The **IntegrationScriptsLocation** property of your Generic cluster profile specifies the folder containing the integration scripts that your cluster profile uses to submit MATLAB jobs to the cluster. Other users must have access to these integration scripts or a copy of them in order to submit jobs to the cluster. As the one distributing a Generic cluster profile and integration scripts, you must decide how other users will access these scripts.

- If you prefer to put the integration scripts in a read-only shared location, follow the steps in “Shared IntegrationScriptsLocation Folder” on page 3-54. This option simplifies subsequent steps and allows any changes you make to the integration scripts to take effect immediately for all users.
- If you prefer to give other users their own copy of your integration scripts, follow the steps in “Distribute Copies of the IntegrationScriptsLocation Folder” on page 3-55.

Shared IntegrationScriptsLocation Folder

If other users have read access to the **IntegrationScriptsLocation** folder specified in your cluster profile, they need only a copy of your profile to submit MATLAB jobs to the cluster.

Note If you have moved your IntegrationScriptsLocation folder to a shared location, remember to update the **IntegrationScriptsLocation** property of your cluster profile before continuing.

- To distribute a copy of your profile, you must:
 - 1 Open MATLAB and navigate to **Home > Parallel > Manage Cluster Profiles** to open the Cluster Profile Manager.
 - 2 Select your profile in the list and click **Export**.
 - 3 Choose a name for the `.settings` file, which contains your profile, and click **Save**.
 - 4 Send a copy of the `.settings` file to other users.
- To import your profile, other users must:
 - 1 Save the `.settings` file to a location of their choice.
 - 2 Open MATLAB and navigate to **Home > Parallel > Manage Cluster Profiles**.
 - 3 Open the Cluster Profile Manager and click **Import**.
 - 4 Select the `.settings` file and click **Open**. A copy of the profile appears in their profile list.
 - 5 Check that the profile works by selecting the profile in the Cluster Profile Manager and clicking **Validate**.

Distribute Copies of the IntegrationScriptsLocation Folder

If you cannot or prefer not to put your integration scripts in a shared location, you can give users a copy of your integration scripts in addition to your profile.

Note If you make changes to your integration scripts, you will have to distribute copies of your updated integration scripts for the changes to take effect for other users.

- To distribute a copy of your profile and integration scripts, you must:
 - 1 Open MATLAB and navigate to **Home > Parallel > Manage Cluster Profiles** to open the Cluster Profile Manager.
 - 2 Select your profile in the list and click **Export**.
 - 3 Choose a name for the `.settings` file, which contains the exported profile, and click **Save**.
 - 4 Send other users a copy of

- The `.settings` file.
 - The **IntegrationScriptsLocation** folder and all files therein.
- To import your profile and integration scripts, other users must:
 - 1 Save all files to a location of their choice.
 - 2 Open MATLAB and navigate to **Home > Parallel > Manage Cluster Profiles**.
 - 3 Open the Cluster Profile Manager and click **Import**.
 - 4 Select the `.settings` file and click **Open**. A copy of the profile appears in their profile list.
 - 5 Select the profile in the Cluster Profile Manager and click **Edit**. Scroll down to the Scheduler Integration section of the profile and change the **IntegrationScriptsLocation** property to point to their copy of the **IntegrationScriptsLocation** folder.
 - 6 Check that the profile works by selecting the profile in the Cluster Profile Manager and clicking **Validate**.

Further Considerations for Clusters with Shared File Systems

If you are distributing a Generic cluster profile with the **HasSharedFileSystem** property set to true (e.g. shared or remote submission modes, see “Interfacing with Generic Schedulers” on page 3-44), the cluster machines must have read and write access to the folder specified in the **JobStorageLocation** property of the profile. Remind other users receiving the profile that they must either:

- 1 Set the **JobStorageLocation** property of their profile to a shared location, preferably one that is unique to their user name and MATLAB version.
- 2 Only create or submit jobs to the cluster when the current working folder is a shared location.

See Also

“Interfacing with Generic Schedulers” on page 3-44 | “Support Scripts” on page 3-44

Configure a Hadoop Cluster

In this section...

“Cluster Configuration” on page 3-57

“Client Configuration” on page 3-57

“Kerberos Authentication” on page 3-58

“Hadoop Version Support” on page 3-59

Parallel MATLAB code that contains `tal` arrays and `mapreduce` functions can be submitted to the Hadoop cluster from suitably configured MATLAB clients.

To configure the client to run MATLAB code on the cluster, you must already be able to submit to the cluster from the intended client machine. The client machine must have a Hadoop installation that can access the cluster outside of MATLAB.

Many Hadoop distributions do not support direct access of Linux based clusters from Windows clients. Users of Windows clients typically need to set up a Linux gateway node that can be accessed from the Windows client via SSH or VNC. The cluster can then be accessed from this gateway node.

Cluster Configuration

- 1 Integrate MATLAB Parallel Server with your cluster infrastructure. For instructions, see “Integrate MATLAB with Third-Party Schedulers” on page 3-2.
- 2 If your cluster requires Kerberos authentication, ensure your MATLAB Parallel Server installations have been configured correctly. For instructions, see “Kerberos Authentication” on page 3-58.

Client Configuration

- 1 Ensure your client can access the Hadoop cluster outside MATLAB.
- 2 Ensure your client MATLAB installation has been configured for Kerberos authentication if your cluster requires it. For instructions, see “Kerberos Authentication” on page 3-58.

To access the cluster from within MATLAB, set up a `parallel.cluster.Hadoop` object using the following statements.

```
setenv('HADOOP_HOME', '/path/to/hadoop/install')
cluster = parallel_cluster.Hadoop;
```

Use `mapreducer` to specify `mapreduce` to run on the Hadoop cluster object.

For examples of how to run parallel MATLAB code on your Hadoop cluster, see “Run mapreduce on a Hadoop Cluster” (Parallel Computing Toolbox) and “Use Tall Arrays on a Spark Enabled Hadoop Cluster” (Parallel Computing Toolbox).

Kerberos Authentication

If the cluster uses Kerberos authentication that requires the Oracle® Java Cryptography Extension, you must configure all installations of MATLAB and MATLAB Parallel Server. If you are using Hortonworks® or Cloudera® distributions, it is likely that you need to complete these configuration steps.

The configuration instructions are the same for client and worker MATLAB installations.

Starting in R2018b, configure your MATLAB installation by enabling the appropriate security policy in the Java installation.

- 1 In the MATLAB Editor, open the file `${MATLAB_ROOT}/sys/java/jre/${ARCH}/jre/lib/security/java.security`.
- 2 Change the line

```
#crypto.policy=unlimited

to

crypto.policy=unlimited
```

For previous releases, you must download additional security files from Oracle.

- 1 Download the Oracle Java Cryptography Extension zip file from the Oracle Java SE page.
- 2 Unzip the downloaded zip file into a temporary folder.
- 3 Replace the files `local_policy.jar` and `US_export_policy.jar` in the folder `${MATLABROOT}/sys/java/jre/${ARCH}/jre/lib/security` with the downloaded versions.

Hadoop Version Support

- MATLAB mapreduce is supported on Hadoop 2.x clusters. Note that support for Hadoop 1.x clusters has been removed.
- MATLAB tall arrays are supported on Spark[®] enabled Hadoop 2.x clusters. You can use tall arrays on Spark enabled Hadoop clusters supporting all architectures for the client, while supporting Linux and Mac architectures for the cluster. This includes cross-platform support.

Functionality	Result	Use Instead	Compatibility Considerations
Support for running MATLAB mapreduce on Hadoop 1.x clusters has been removed.	Errors	Use clusters that have Hadoop 2.x installed to run MATLAB mapreduce.	Migrate MATLAB mapreduce code that runs on Hadoop 1.x to Hadoop 2.x.

See Also

`parallel.cluster.Hadoop`

Related Examples

- “Integrate MATLAB with Third-Party Schedulers” on page 3-2
- “Use Tall Arrays on a Spark Enabled Hadoop Cluster” (Parallel Computing Toolbox)
- “Run mapreduce on a Hadoop Cluster” (Parallel Computing Toolbox)
- “Read and Analyze Hadoop Sequence File” (MATLAB)

Admin Center

- “Start Admin Center” on page 4-2
- “Set Up Resources” on page 4-3
- “Test Connectivity” on page 4-11
- “Export and Import Sessions” on page 4-14
- “Prepare for Cluster Profiles” on page 4-15

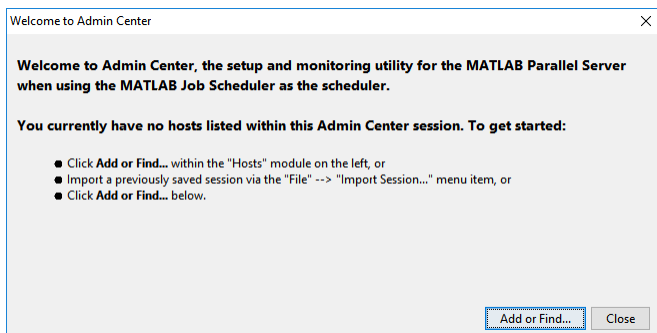
Start Admin Center

Admin Center is a graphical user interface with which you can control and monitor the MATLAB Parallel Server processes of a MATLAB Job Scheduler cluster. Admin center does not support any common job scheduler (CJS) clusters or third-party schedulers.

You start Admin Center outside a MATLAB session by executing the following:

- `matlabroot/toolbox/distcomp/bin/admincenter` (on UNIX operating systems)
- `matlabroot\toolbox\distcomp\bin\admincenter.bat` (on Microsoft Windows operating systems)

The first time you start Admin Center, you see the following welcome dialog box.



A new session of Admin Center has no cluster hosts listed, so the usual first step is to identify the hosts you want to include in your listing. To do this, click **Add or Find**. Further information continues in the next section, “Set Up Resources” on page 4-3.

If you start Admin Center again on the same host, your previous session for that machine is loaded; and unless the update rate is set to *never*, Admin Center performs an update immediately for the listed hosts and processes. To clear this information and start a new session, select the pull-down **File > New Session**.

Set Up Resources

In this section...

“Add Hosts” on page 4-3

“Start mjs Service” on page 4-4

“Start a MATLAB Job Scheduler” on page 4-5

“Start Workers” on page 4-7

“Stop, Destroy, Resume, Restart Processes” on page 4-9

“Move a Worker” on page 4-10

“Update the Display” on page 4-10

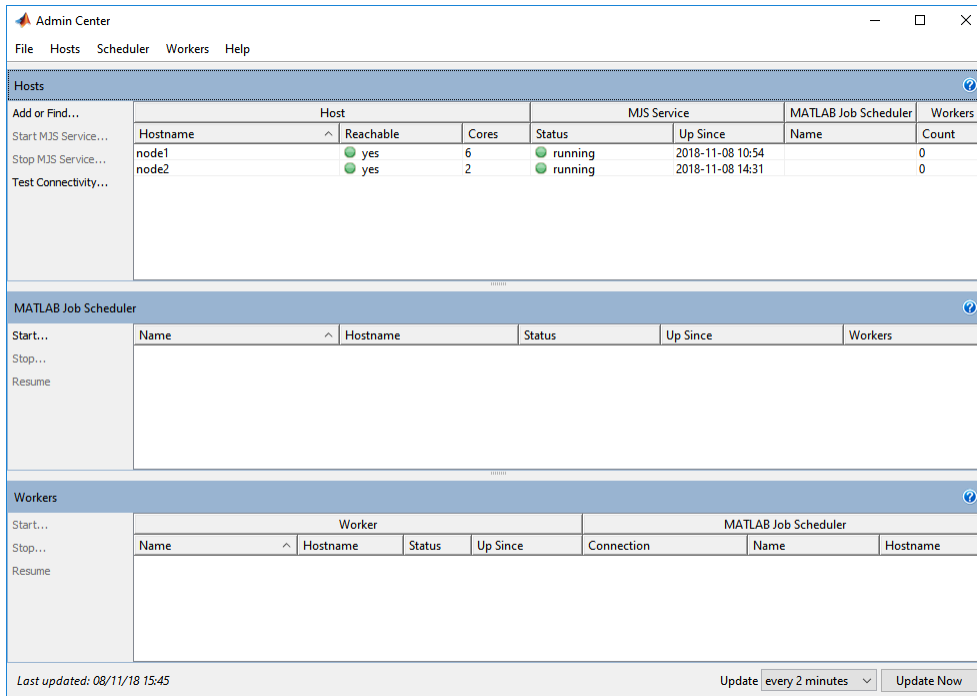
Add Hosts

To specify the hosts you want listed in Admin Center, click **Add or Find** in the Welcome dialog box, or if this is not a new session, click **Add or Find** in the Hosts module.

In the Add or Find Hosts dialog box, identify the hosts you want to add to the listing, by one of the following methods:

- Select **Enter Hostnames** and provide short host names, fully qualified domain names, or individual IP addresses for the hosts.
- Select **Enter IP Range** and provide the range of IP addresses for your hosts.

If one of the hosts you have specified is running a MATLAB Job Scheduler, Admin Center automatically finds and lists all the hosts running workers registered with that MATLAB Job Scheduler. Similarly, if you specify a host that is running a worker, Admin Center finds and lists the host running that worker’s MATLAB Job Scheduler, and then also all hosts running other workers under that MATLAB Job Scheduler.



Start mjs Service

A host must be running the mjs service if an MATLAB Job Scheduler or worker is to run on that host. Normally, you set this up with Admin Center or command-line scripts during the installation of MATLAB Parallel Server on your cluster, as described in the installation instructions available at “Integrate MATLAB Job Scheduler for Network License Manager” on page 3-6.

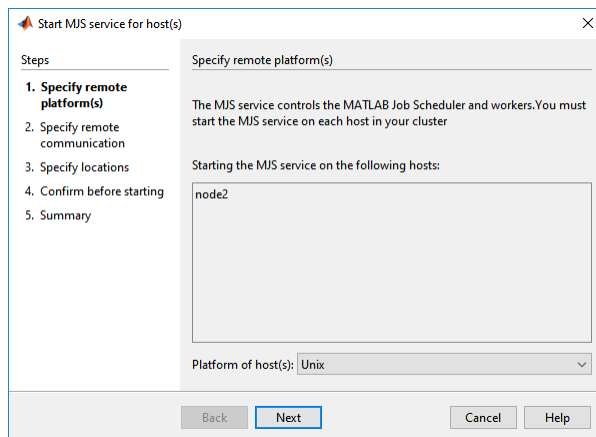
If you want to add or remove hosts to your cluster, Admin Center allows you to start and stop the mjs service on those hosts. To start the mjs service on a group of hosts with the same platform, select all those hosts in the Hosts module, and click **Start MJS Service** in the left column of the panel.

Alternative methods for starting mjs include selecting the pull-down **Hosts > Start MJS Service**, or right-clicking a listed host and selecting, **Start MJS Service**.

A dialog box leads you through the procedure of starting the mjs service on the selected hosts. There are five steps to the procedure in which you provide or confirm information for the service:

- 1 Specify remote platform — Windows or UNIX. You can start mjs on multiple hosts at the same time, but they all must be the same platform. If you have a mixed platform cluster, run the mjs startup separately for each type of platform.
- 2 Specify remote communication — Choose the protocol for communication with the hosts.
- 3 Specify locations — Specify the location of the MATLAB installation and the mjs_def file for the hosts.
- 4 Confirm before starting — Review information before proceeding.
- 5 Summary — Status about the startup attempt.

The dialog box looks like this for the first step:

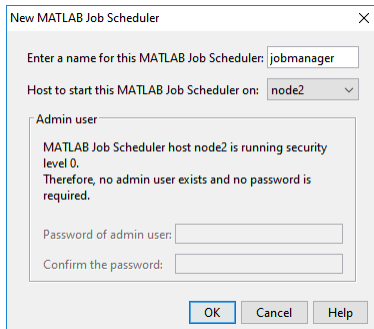


At each step, you can click **Help** to read detailed information about that step.

Start a MATLAB Job Scheduler

To start a MATLAB Job Scheduler, click **Start** in the MATLAB Job Scheduler module.

In the New MATLAB Job Scheduler dialog box, provide a name for the MATLAB Job Scheduler, and select a host to run it on.



The image shows a dialog box titled "New MATLAB Job Scheduler" with a close button (X) in the top right corner. It contains the following fields and text:

- Text input: "Enter a name for this MATLAB Job Scheduler:" with the value "jobmanager".
- Dropdown menu: "Host to start this MATLAB Job Scheduler on:" with the value "node2".
- Section header: "Admin user".
- Text: "MATLAB Job Scheduler host node2 is running security level 0. Therefore, no admin user exists and no password is required."
- Text input: "Password of admin user:" (empty).
- Text input: "Confirm the password:" (empty).
- Buttons: "OK", "Cancel", and "Help" at the bottom.

Alternative methods for starting a MATLAB Job Scheduler include selecting the pull-down **Scheduler > Start**, or right-clicking a listed host and selecting, **Start Scheduler**.

With a MATLAB Job Scheduler running on your cluster, Admin Center might look like the following figure, with the MATLAB Job Scheduler listed in the MATLAB Job Scheduler module, as well as being listed by name in the Hosts module in the line for the host on which it is running.

The screenshot shows the Admin Center interface with three main sections: Hosts, MATLAB Job Scheduler, and Workers.

Hosts Section:

Add or Find...	Host			MJS Service		MATLAB Job Sche...	Workers
	Hostname	Reachable	Cores	Status	Up Since	Name	Count
Start MJS Service...	node1	yes	6	running	2018-11-08 10:54	jobmanager	0
Stop MJS Service...	node2	yes	2	running	2018-11-08 10:59		0
Test Connectivity...							

MATLAB Job Scheduler Section:

Start...	Name	Hostname	Status	Up Since	Workers
	Stop...	jobmanager	node1	running	2018-11-08 11:41
Resume					

Workers Section:

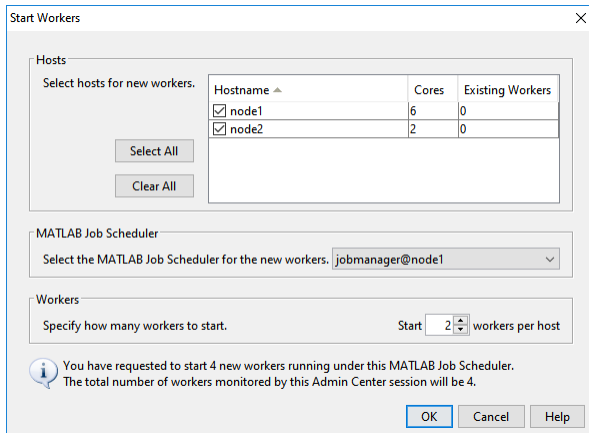
Start...	Worker				MATLAB Job Scheduler		
	Name	Hostname	Status	Up Since	Connection	Name	Hostname
Stop...							
Resume							

At the bottom of the interface, it shows "Last updated: 08/11/18 11:41" and an "Update" button set to "every 2 minutes".

Start Workers

To start MATLAB workers, click **Start** in the Workers module.

In the Start Workers dialog box, specify the numbers of workers to start on each host, and select the hosts to run them. From the list, select the MATLAB Job Scheduler for these workers. Click **OK** to start the workers. Admin center automatically provides names for the workers, based on the hosts running them.



Alternative methods for starting workers include selecting the pull-down **Workers > Start**, or right-clicking a listed host or MATLAB Job Scheduler and selecting **Start Workers**.

With workers running on your cluster, Admin Center might look like the following figure, which shows the workers listed in the Workers module. Also, the number of workers running under the MATLAB Job Scheduler is listed in the MATLAB Job Scheduler module, and the number of workers for each MATLAB Job Scheduler is listed in the Hosts module.

The screenshot shows the Admin Center interface with three main sections: Hosts, MATLAB Job Scheduler, and Workers. Each section has a table of data and a sidebar with control options.

Hosts

Add or Find...		Host		MJS Service		MATLAB Job Scheduler	Workers
Hostname	Reachable	Cores	Status	Up Since	Name	Count	
node1	yes	6	running	2018-11-08 10:54	jobmanager	2	
node2	yes	2	running	2018-11-08 10:59		2	

MATLAB Job Scheduler

Start...	Name	Hostname	Status	Up Since	Workers
Stop...	jobmanager	node1	running	2018-11-08 11:41	4
Resume					

Workers

Start...		Worker				MATLAB Job Scheduler	
Stop...	Name	Hostname	Status	Up Since	Connection	Name	Hostname
Resume	worker1	node1	idle	2018-11-08 13:50	connected	jobmanager	node1
	worker2	node1	idle	2018-11-08 13:50	connected	jobmanager	node1
	worker3	node2	idle	2018-11-08 13:51	connected	jobmanager	node1
	worker4	node2	idle	2018-11-08 13:52	connected	jobmanager	node1

Last updated: 08/11/18 13:52 Update: every 2 minutes Update Now

To get more information on any host, MATLAB Job Scheduler, or worker listed in Admin Center, right-click its name in the display and select **Properties**. Alternatively, you can find the **Properties** option under the **Hosts**, **Scheduler**, and **Workers** drop-down menus.

Stop, Destroy, Resume, Restart Processes

You can **Stop** or **Destroy** the mjs service, MATLAB Job Schedulers, and workers. The primary difference is that stopping a process shuts it down but retains its data; destroying a process shuts it down and clears its data. Use **Start MJS Service** to have mjs continue with existing data. Use **Resume** to have an MATLAB Job Scheduler or worker continue with its existing data. When you use **Restart**, a dialog box requires you to confirm your intention of starting a new process while keeping or discarding data.

Move a Worker

To move a worker from one host to another, you must completely shut it down, than start a new worker on the desired host:

- 1 Right-click the worker in the Workers module list.
- 2 Select **Destroy**. This shuts down the worker process and removes all its data.
- 3 If the old worker host is not running any other MATLAB Parallel Server processes (mjs service, MATLAB Job Scheduler, or workers), you might want to remove it from the Admin Center listing.
- 4 If necessary, add the new host to the Admin Center host listing.
- 5 In the Workers module, click **Start**. Select the desired host in the Start Workers dialog box, along with the appropriate number and MATLAB Job Scheduler name.

Use a similar process to move a MATLAB Job Scheduler from one host to another. Note, however, that all workers registered with the MATLAB Job Scheduler must be destroyed and started again, registering them with the new instance of the MATLAB Job Scheduler.

Update the Display

Admin Center updates its data automatically at regular intervals. To set the update rate, select an option from the **Update** list. Click **Update Now** to immediately update the display data.

Test Connectivity

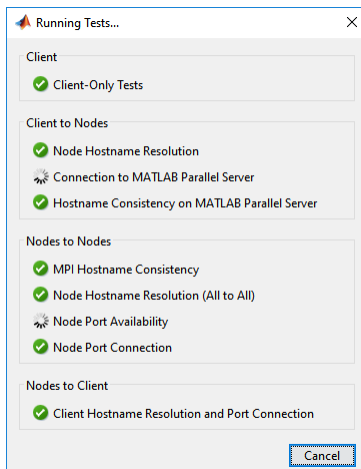
Admin Center lets you test communications between your MATLAB Job Scheduler node, worker nodes, and the node where Admin Center is running.

The tests are divided into four categories:

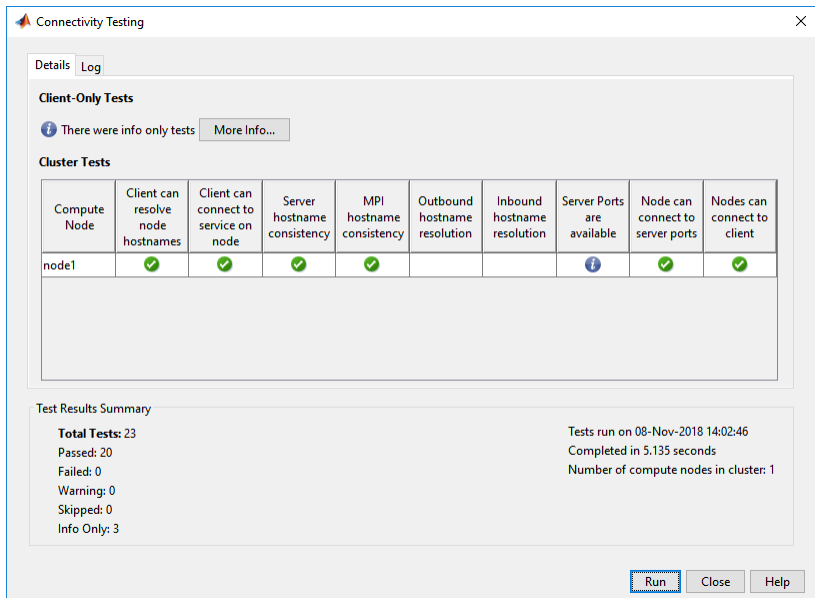
- **Client** — Verifies that the node running Admin Center is properly configured so that further cluster testing can proceed.
- **Client to Nodes** — Verifies that the node running Admin Center can identify and communicate with the other nodes in the cluster.
- **Nodes to Nodes** — Verifies that the other nodes in the cluster can identify each other, and that each node allows its mjs service to communicate with the mjs service on the other cluster nodes.
- **Nodes to Client** — Verifies that other cluster nodes can identify and communicate with the node running Admin Center.

First click **Test Connectivity** to open the Connectivity Testing dialog box. By default, the dialog box displays the results of the last test. To run new tests and update the display, click **Run**.

During test execution, Admin Center displays this progress dialog box.



When the tests are complete, the Running Tests dialog box automatically closes, and Admin Center displays the test results in the Connectivity Testing dialog box.



The possible test result symbols are described in the following table.

Test Result	Description
	Test passed.
	Test passed, extra information is available.
	Test passed, but generated a warning.
	Test failed.
	Test was skipped, possibly because prerequisite tests did not pass.

Test that include failures or other results might look like the following figure.

Connectivity Testing

Details Log

Client-Only Tests

There were info only tests [More Info...](#)

Cluster Tests

Comp... Node	Client can resolve node hostna...	Client can connect to service on node	Server hostna... consist...	MPI hostname consistency	Outbo... hostna... resolut...	Inbound hostna... resoluti...	Server Ports are available	Node can connect to server ports	Nodes can connect to client
node1	✓	✓	✓	✓	✓	✓	i	✓	✓
node2	✓	✓	✓	✗	✓	✓	i	⊘	✓

Test Results Summary

Total Tests: 49
 Passed: 40
 Failed: 1
 Warning: 0
 Skipped: 3
 Info Only: 5

Tests run on 08-Nov-2018 13:54:29
 Completed in 4.633 seconds
 Number of compute nodes in cluster: 2

Run Close Help

Double-click any of the symbols in the test results to drill down for more detail. Use the **Log** tab to see the raw data from the tests.

The results of the tests that run on only the client are displayed in the lower-left corner of the dialog box. To drill into client-only test results, click **More Info**.

Export and Import Sessions

By default, Admin Center saves the cluster definition, process status, and test results, so the next time the same user runs Admin Center on the same machine, that saved information is available and displayed by default. You can export session data so that a different user or a different host can access it, by selecting the pull-down **File > Export Session**. Browse to the location where you want to store the session data and provide a name for the file. Admin Center applies the extension `.mjs` to the file name.

You can import that saved session data into a subsequent session of Admin Center by selecting the pull-down **File > Import Session**. The imported data includes cluster definition and test results.

Note When importing a session file, Admin Center automatically sets its update rate to never (i.e., disabled), so that you can statically examine a cluster setup from the time the session was saved for evaluation or diagnostic purposes.

Prepare for Cluster Profiles

Admin Center does not create cluster profiles, but the information displayed in Admin Center is of vital importance when you create your cluster profiles — information such as MATLAB Job Scheduler name, MATLAB Job Scheduler host, and number of workers. For more information about creating and using profiles, see “Discover Clusters and Use Cluster Profiles” (Parallel Computing Toolbox).

Control Scripts — Alphabetical List

admincenter

Start Admin Center GUI

Syntax

admincenter

Description

admincenter opens the MATLAB Parallel Server Admin Center. When setting up or using a MATLAB job scheduler cluster, Admin Center allows you to establish and verify your cluster, and to diagnose possible problems.

For details about using Admin Center, see:

- “Start Admin Center” on page 4-2
- “Set Up Resources” on page 4-3
- “Test Connectivity” on page 4-11

See Also

mjs | nodestatus | remotemjs

createSharedSecret

Create shared secret for secure communication

Syntax

```
createSharedSecret  
createSharedSecret -file <filename>
```

Description

createSharedSecret creates a shared secret file used for secure communication between job managers and workers. The file is named `secret` in the current folder.

```
createSharedSecret -file <filename>
```

 create a shared secret file as the given filename.

Before passing sensitive data from one service to another (e.g., between job manager and workers), these services need to establish a trust relationship using a shared secret. This script creates a file that serves as a shared secret between the services. Each service is trusted that has access to that secret file.

Create the secret file only once per cluster on one machine, then copy it into the location specified by `SHARED_SECRET_FILE` in the `mjs_def` file on each machine before starting any job managers or workers. In a shared file system, all nodes can point to the same file. Shared secrets can be reused in subsequent sessions.

Examples

Create a shared secret file in a central location for all the nodes of the cluster:

```
cd matlabInstallDir/toolbox/distcomp/bin  
createSharedSecret -file /share/secret
```

Then make sure that the nodes' shared or copied `mjs_def` files set the parameter `SHARED_SECRET_FILE` to `/share/secret` before starting the `mjs` service on each.

See Also

mjs

mjs

Install, start, stop, or uninstall mjs service

Syntax

```
mjs install
mjs uninstall
mjs start
mjs stop
mjs console
mjs restart
mjs ... -mjsdef <mjs_defaults_file>
mjs ... -clean
mjs status
mjs -version
mjs -useonlinelicensing
```

Description

The mjs service ensures that all other processes are running and that it is possible to communicate with them. Once the mjs service is running, you can use the `nodestatus` command to obtain information about the mjs service and all the processes it maintains.

The mjs executable resides in the folder `matlabroot\toolbox\distcomp\bin` (Windows operating system) or `matlabroot/toolbox/distcomp/bin` (UNIX operating system). Enter the following commands at a DOS or UNIX command-line prompt, respectively.

`mjs install` installs the mjs service in the Microsoft Windows Service Control Manager. This causes the service to automatically start when the Windows operating system boots up. The service must be installed before it is started.

`mjs uninstall` uninstalls the mjs service from the Windows Service Control Manager. Note that if you wish to install mjs service as a different user, you must first uninstall the service and then reinstall as the new user.

`mjs start` starts the `mjs` service. This creates the required logging and checkpointing directories, and then starts the service as specified in the `mjs` defaults file.

`mjs stop` stops running the `mjs` service. This automatically stops all job managers and workers on the computer, but leaves their checkpoint information intact so that they will start again when the `mjs` service is started again.

`mjs console` starts the `mjs` service as a process in the current terminal or command window rather than as a service running in the background.

`mjs restart` performs the equivalent of `mjs stop` followed by `mjs start`. This command is available only on UNIX and Macintosh operating systems.

`mjs ... -mjsdef <mjs_defaults_file>` uses the specified alternative `mjs` defaults file instead of the one found in `matlabroot/toolbox/distcomp/bin`. This file is provided for Linux (`mjs_def.sh`) and Windows (`mjs_def.bat`). For information on backwards compatibility, see “Run Multiple MATLAB Parallel Server Versions” on page 3-19.

`mjs ... -clean` performs a complete cleanup of all service checkpoint and log files before installing or starting the service, or after stopping or uninstalling it. This deletes all information about any job managers or workers this service has ever maintained.

`mjs status` reports the status of the `mjs` service, indicating whether it is running and with what PID. Use `nodestatus` to obtain more detailed information about the `mjs` service. The `mjs status` command is available only on UNIX and Macintosh operating systems.

`mjs -version` prints version information of the `mjs` process to standard output, then exits.

`mjs -useonlinelicensing` ensures that workers use online licensing. Unless you specify `-useonlinelicensing`, `mjs` uses the network license manager.

See Also

`nodestatus` | `startjobmanager` | `startworker` | `stopjobmanager` | `stopworker`

nodestatus

Status of mjs processes running on node

Syntax

```
nodestatus
nodestatus -flags
```

Description

`nodestatus` displays the status of the mjs service and the processes which it maintains. The mjs service must already be running on the specified computer.

The `nodestatus` executable resides in the folder `matlabroot\toolbox\distcomp\bin` (Windows operating system) or `matlabroot/toolbox/distcomp/bin` (UNIX operating system). Enter the following command syntax at a DOS or UNIX command-line prompt, respectively.

`nodestatus -flags` accepts the following input flags. Multiple flags can be used together on the same command.

Flag	Operation
<code>-remotehost <hostname></code>	Displays the status of the mjs service and the processes it maintains on the specified host. The default value is the local host.
<code>-infolevel <level></code>	Specifies how much status information to report, using a level of 1-3. 1 means only the basic information, 3 means all information available. The default value is 1.

Flag	Operation
<code>-baseport <port_number></code>	Specifies the base port that the mjs service on the remote host is using. You need to specify this only if the value of <code>BASE_PORT</code> in the local <code>mjs_def</code> file does not match the base port being used by the mjs service on the remote host.
<code>-v</code>	Verbose mode displays the progress of the command execution.
<code>-json</code>	View the output in JavaScript Object Notation (JSON) format. Output in <code>json</code> format is easy to parse.

Examples

Display basic information about the mjs processes on the local host.

```
nodestatus
```

Display detailed information about the status of the mjs processes on host `node27`.

```
nodestatus -remotehost node27 -infolevel 2
```

See Also

`mjs` | `startjobmanager` | `startworker` | `stopjobmanager` | `stopworker`

remotecopy

Copy file or folder to or from one or more remote hosts using transport protocol

Syntax

```
remotecopy <flags> <protocol options>
```

Description

remotecopy <flags> <protocol options> copies a file or folder to or from one or more remote hosts by using a transport protocol (such as ssh). Copying from multiple hosts creates a separate file per host, appending the hostname to the specified filename.

The following table describes the supported flags and options. You can combined multiple flags in the same command, preceding each flag by a dash (-).

Flags and Options	Operation
-local <file-or-foldername>	Specify the name of the file or folder on the local host.
-remote <file-or-foldername>	Specify the name of the file or folder on the remote host.
-from	Specify to copy from the remote hosts to the local host. You must use either the -from flag, or the -to flag.
-to	Specify to copy to the remote hosts from the local host. You must use either the -from flag, or the -to flag.
-remotehost host1[,host2[,...]]	Specify the names of the hosts where you want to copy to or from. Separate the host names by commas without any white spaces. This is a mandatory argument.

Flags and Options	Operation
<code>-remoteplatform { unix windows }</code>	Specify the platform of the remote hosts. This option is required only if different from the local platform.
<code>-quiet</code>	Prevent <code>remotecopy</code> from prompting for missing information. The command fails if all required information is not specified.
<code>-help</code>	Print the help information for this command.
<code>-protocol <type></code>	<p>Force the usage of a particular protocol type. Specifying a protocol type with all its required parameters also avoids interactive prompting and allows for use in scripts.</p> <p>The supported protocol types are <code>scp</code> and <code>sftp</code>.</p> <p>To get more information about one particular protocol type, enter</p> <pre>remotecopy -protocol <type> -help</pre> <p>For example:</p> <pre>remotecopy -protocol sftp -help</pre>
<code><protocol options></code>	Specify particular options for the protocol type being used.

Note The file permissions on the copy might not be the same as the permissions on the original file.

Examples

Copy the local file `mjs_def.sh` to two other machines. (Enter this command on a single line.)

```
remotecopy -local mjs_def.sh -to
  -remote /matlab/toolbox/distcomp/bin -remotehost hostA,hostB
```


Retrieve folders of the same name from two hosts to the local machine. (Enter command on a single line.)

```
remotecopy -local C:\temp\log -from -remote C:\temp\mjs\log  
-remotehost winHost1,winHost2
```

Compatibility Considerations

The remotecopy function will no longer support the rcp protocol in a future release

Warns starting in R2018b

The rcp protocol is insecure. The remotecopy function will no longer support rcp as an option to the `-protocol` flag in a future release. Instead, specify any other of the supported protocols, such as scp.

See Also

remotemjs

remotemjs

Execute mjs command on one or more remote hosts by transport protocol

Syntax

```
remotemjs <mjs options> <flags> <protocol options>
```

Description

remotemjs <mjs options> <flags> <protocol options> allows you to execute the mjs service on one or more remote hosts.

For a description of the mjs service, see the mjs reference page.

The following table describes the supported flags and options. You can combined multiple flags in the same command, preceding each flag by a dash (-).

Flags and Options	Operation
<mjs options>	Options and arguments of the mjs command, such as <code>start</code> , <code>stop</code> , etc. See the mjs reference page for a full list.
-matlabroot <installfoldername>	The MATLAB installation folder on the remote hosts, required only if the remote installation folder differs from the one on the local machine.
-remotehost host1[,host2[,...]]	The names of the hosts where you want to run the mjs command. Separate the host names by commas without any white spaces. This is a mandatory argument.
-remoteplatform { unix windows }	The platform of the remote hosts. This option is required only if different from the local platform.
-quiet	Prevent mjs from prompting the user for missing information. The command fails if all required information is not specified.

Flags and Options	Operation
-help	Print help information.
-protocol <type>	<p>Force the usage of a particular protocol type. Specifying a protocol type with all its required parameters also avoids interactive prompting and allows for use in scripts.</p> <p>The supported protocol types are <code>ssh</code> and <code>winsc</code>.</p> <p>To get more information about one particular protocol type, enter</p> <pre>remotemjs -protocol <type> -help</pre> <p>For example:</p> <pre>remotemjs -protocol winsc -help</pre> <p>Using the <code>winsc</code> protocol requires that you log in as a user with admin privileges on the remote host.</p>
<protocol options>	Specify particular options for the protocol type being used.

Note If you are using OpenSSHd on a Microsoft Windows operating system, you can encounter a problem when using backslashes in path names for your command options. In most cases, you can work around this problem by using forward slashes instead. For example, to specify the file `C:\temp\mjs_def.bat`, you should identify it as `C:/temp/mjs_def.bat`.

Examples

Start mjs on three remote machines of the same platform as the client:

```
remotemjs start -remotehost hostA,hostB,hostC
```

Start mjs in a clean state on two UNIX operating system machines from a Windows operating system machine, using the ssh protocol. Enter the following command on a single line:

```
remotemjs start -clean -matlabroot /usr/local/matlab  
-remotehost unixHost1,unixHost2 -remoteplatform UNIX  
-protocol ssh
```

Compatibility Considerations

The remotemjs function will no longer support the rsh protocol in a future release

Warns starting in R2018b

The rsh protocol is insecure. The remotemjs function will no longer support rsh as an option to the `-protocol` flag in a future release. Instead, specify any other of the supported protocols, such as ssh.

See Also

mjs | remotecopy

pausejobmanager

Pause job manager process

Syntax

```
pausejobmanager
pausejobmanager -flags
```

Description

pausejobmanager pauses a job manager that is running under the mjs service.

The pausejobmanager executable resides in the folder *matlabroot\toolbox\distcomp\bin* (Windows operating system) or *matlabroot/toolbox/distcomp/bin* (UNIX operating system). Enter the following command syntax at a DOS or UNIX command-line prompt, respectively.

pausejobmanager -flags accepts the following input flags. Multiple flags can be used together on the same command.

Flag	Operation
-name <job_manager_name>	Specifies the name of the job manager to pause. The default is the value of DEFAULT_JOB_MANAGER_NAME parameter the mjs_def file.
-remotehost <hostname>	Specifies the name of the host where you want to pause the job manager. The default value is the local host.
-baseport <port_number>	Specifies the base port that the mjs service on the remote host is using. You need to specify this only if the value of BASE_PORT in the local mjs_def file does not match the base port being used by the mjs service on the remote host.

Flag	Operation
-v	Verbose mode displays the progress of the command execution.

Examples

Pause the job manager MyJobManager on the local host.

```
pausejobmanager -name MyJobManager
```

Pause the job manager MyJobManager on the host JMHost.

```
pausejobmanager -name MyJobManager -remotehost JMHost
```

See Also

mjs | nodestatus | resumejobmanager | startjobmanager | stopjobmanager

resumejobmanager

Resume job manager process

Syntax

```
resumejobmanager
resumejobmanager -flags
```

Description

resumejobmanager resumes a job manager that is running under the mjs service.

The resumejobmanager executable resides in the folder *matlabroot\toolbox\distcomp\bin* (Windows operating system) or *matlabroot/toolbox/distcomp/bin* (UNIX operating system). Enter the following command syntax at a DOS or UNIX command-line prompt, respectively.

resumejobmanager -flags accepts the following input flags. Multiple flags can be used together on the same command.

Flag	Operation
-name <job_manager_name>	Specifies the name of the job manager to resume. The default is the value of DEFAULT_JOB_MANAGER_NAME parameter the mjs_def file.
-remotehost <hostname>	Specifies the name of the host where you want to resume the job manager. The default value is the local host.
-baseport <port_number>	Specifies the base port that the mjs service on the remote host is using. You need to specify this only if the value of BASE_PORT in the local mjs_def file does not match the base port being used by the mjs service on the remote host.

Flag	Operation
-v	Verbose mode displays the progress of the command execution.

Examples

Resume the job manager MyJobManager on the local host.

```
resumejobmanager -name MyJobManager
```

Resume the job manager MyJobManager on the host JMHost.

```
resumejobmanager -name MyJobManager -remotehost JMHost
```

See Also

mjs | nodestatus | pausejobmanager | startjobmanager | stopjobmanager

startjobmanager

Start job manager process

Syntax

```
startjobmanager
startjobmanager -flags
```

Description

`startjobmanager` starts a job manager process and the associated job manager lookup process under the `mjs` service, which maintains them after that. The job manager handles the storage of jobs and the distribution of tasks contained in jobs to MATLAB workers that are registered with it. The `mjs` service must already be running on the specified computer.

The `startjobmanager` executable resides in the folder `matlabroot\toolbox\distcomp\bin` (Windows operating system) or `matlabroot/toolbox/distcomp/bin` (UNIX operating system). Enter the following command syntax at a DOS or UNIX command-line prompt, respectively.

`startjobmanager -flags` accepts the following input flags. Multiple flags can be used together on the same command.

Flag	Operation
<code>-name <job_manager_name></code>	Specifies the name of the job manager. This identifies the job manager to MATLAB worker sessions and MATLAB clients. The default is the value of the <code>DEFAULT_JOB_MANAGER_NAME</code> parameter in the <code>mjs_def</code> file.
<code>-remotehost <hostname></code>	Specifies the name of the host where you want to start the job manager and the job manager lookup process. If omitted, they start on the local host.

Flag	Operation
-clean	Deletes all checkpoint information stored on disk from previous instances of this job manager before starting. This cleans the job manager so that it initializes with no existing jobs or tasks.
-baseport <port_number>	Specifies the base port that the mjs service on the remote host is using. You need to specify this only if the value of BASE_PORT in the local mjs_def file does not match the base port being used by the mjs service on the remote host.
-useMSMPI	Use Microsoft MPI (MS-MPI) for clusters on Windows platforms.
-v	Verbose mode displays the progress of the command execution.

Examples

Start the job manager MyJobManager on the local host.

```
startjobmanager -name MyJobManager
```

Start the job manager MyJobManager on the host JMHost.

```
startjobmanager -name MyJobManager -remotehost JMHost
```

See Also

mjs | nodestatus | pausejobmanager | resumejobmanager | startworker | stopjobmanager | stopworker

startworker

Start MATLAB worker session

Syntax

```
startworker
startworker -flags
```

Description

`startworker` starts a MATLAB worker process under the `mjs` service, which maintains it after that. The worker registers with the specified job manager, from which it will get tasks for evaluation. The `mjs` service must already be running on the specified computer.

The `startworker` executable resides in the folder `matlabroot\toolbox\distcomp\bin` (Windows operating system) or `matlabroot/toolbox/distcomp/bin` (UNIX operating system). Enter the following command syntax at a DOS or UNIX command-line prompt, respectively.

`startworker -flags` accepts the following input flags. Multiple flags can be used together on the same command, except where noted.

Flag	Operation
-name <worker_name>	Specifies the name of the MATLAB worker. The default is the value of the <code>DEFAULT_WORKER_NAME</code> parameter in the <code>mjs_def</code> file.
-remotehost <hostname>	Specifies the name of the computer where you want to start the MATLAB worker. If omitted, the worker is started on the local computer.

Flag	Operation
-jobmanager <job_manager_name>	Specifies the name of the job manager this MATLAB worker will receive tasks from. The default is the value of the <code>DEFAULT_JOB_MANAGER_NAME</code> parameter in the <code>mjs_def</code> file.
-jobmanagerhost <job_manager_hostname>	Specifies the host on which the job manager is running. The worker contacts the job manager lookup process on that host to register with the job manager. This overrides the setting of <code>JOB_MANAGER_HOST</code> in the <code>mjs_def</code> file on the worker computer. You must specify the job manager host by one of these means.
-clean	Deletes all checkpoint information associated with this worker name before starting.
-baseport <port_number>	Specifies the base port that the <code>mjs</code> service on the remote host is using. You only need to specify this if the value of <code>BASE_PORT</code> in the local <code>mjs_def</code> file does not match the base port being used by the <code>mjs</code> service on the remote host.
-v	Verbose mode displays the progress of the command execution.

Examples

Start a worker on the local host, using the default worker name, registering with the job manager `MyJobManager` on the host `JMHost`.

```
startworker -jobmanager MyJobManager -jobmanagerhost JMHost
```

Start a worker on the host `WorkerHost`, using the default worker name, and registering with the job manager `MyJobManager` on the host `JMHost`. (The following command should be entered on a single line.)

```
startworker -jobmanager MyJobManager -jobmanagerhost JMHost
            -remotehost WorkerHost
```

Start two workers, named `worker1` and `worker2`, on the host `WorkerHost`, registering with the job manager `MyJobManager` that is running on the host `JMHost`. Note that to start two workers on the same computer, you must give them different names. (Each of the two commands below should be entered on a single line.)

```
startworker -name worker1 -remotehost WorkerHost  
            -jobmanager MyJobManager -jobmanagerhost JMHost  
startworker -name worker2 -remotehost WorkerHost  
            -jobmanager MyJobManager -jobmanagerhost JMHost
```

See Also

`mjs` | `nodestatus` | `startjobmanager` | `stopjobmanager` | `stopworker`

stopjobmanager

Stop job manager process

Syntax

```
stopjobmanager  
stopjobmanager -flags
```

Description

stopjobmanager stops a job manager that is running under the mjs service.

The stopjobmanager executable resides in the folder *matlabroot\toolbox\distcomp\bin* (Windows operating system) or *matlabroot/toolbox/distcomp/bin* (UNIX operating system). Enter the following command syntax at a DOS or UNIX command-line prompt, respectively.

stopjobmanager -flags accepts the following input flags. Multiple flags can be used together on the same command.

Flag	Operation
-name <job_manager_name>	Specifies the name of the job manager to stop. The default is the value of DEFAULT_JOB_MANAGER_NAME parameter the mjs_def file.
-remotehost <hostname>	Specifies the name of the host where you want to stop the job manager and the associated job manager lookup process. The default value is the local host.
-clean	Deletes all checkpoint information stored on disk for the current instance of this job manager after stopping it. This cleans the job manager of all its job and task data.

Flag	Operation
-baseport <port_number>	Specifies the base port that the mjs service on the remote host is using. You need to specify this only if the value of BASE_PORT in the local mjs_def file does not match the base port being used by the mjs service on the remote host.
-v	Verbose mode displays the progress of the command execution.

Examples

Stop the job manager MyJobManager on the local host.

```
stopjobmanager -name MyJobManager
```

Stop the job manager MyJobManager on the host JMHost.

```
stopjobmanager -name MyJobManager -remotehost JMHost
```

See Also

mjs | nodestatus | startjobmanager | startworker | stopworker

stopworker

Stop MATLAB worker session

Syntax

```
stopworker  
stopworker -flags
```

Description

stopworker stops a MATLAB worker process that is running under the mjs service.

The stopworker executable resides in the folder *matlabroot\toolbox\distcomp\bin* (Windows operating system) or *matlabroot/toolbox/distcomp/bin* (UNIX operating system). Enter the following command syntax at a DOS or UNIX command-line prompt, respectively.

stopworker *-flags* accepts the following input flags. Multiple flags can be used together on the same command.

Flag	Operation
-name <worker_name>	Specifies the name of the MATLAB worker to stop. The default is the value of the DEFAULT_WORKER_NAME parameter in the mjs_def file.
-remotehost <hostname>	Specifies the name of the host where you want to stop the MATLAB worker. The default value is the local host.
-clean	Deletes all checkpoint information associated with this worker name after stopping it.

Flag	Operation
-baseport <port_number>	Specifies the base port that the mjs service on the remote host is using. You need to specify this only if the value of BASE_PORT in the local mjs_def file does not match the base port being used by the mjs service on the remote host.
-v	Verbose mode displays the progress of the command execution.

Examples

Stop the worker with the default name on the local host.

```
stopworker
```

Stop the worker with the default name, running on the computer WorkerHost.

```
stopworker -remotehost WorkerHost
```

Stop the workers named worker1 and worker2, running on the computer WorkerHost.

```
stopworker -name worker1 -remotehost WorkerHost  
stopworker -name worker2 -remotehost WorkerHost
```

See Also

mjs | nodestatus | startjobmanager | startworker | stopjobmanager

Glossary

CHECKPOINTBASE	The name of the parameter in the <code>mjs_def</code> file that defines the location of the checkpoint directories for the MATLAB Job Scheduler and workers.
checkpoint directory	See CHECKPOINTBASE.
client	The MATLAB session that defines and submits the job. This is the MATLAB session in which the programmer usually develops and prototypes applications. Also known as the MATLAB client.
client computer	The computer running the MATLAB client; often your desktop.
cluster	A collection of computers that are connected via a network and intended for a common purpose.
coarse-grained application	An application for which run time is significantly greater than the communication time needed to start and stop the program. Coarse-grained distributed applications are also called embarrassingly parallel applications.
codistributed array	An array partitioned into segments, with each segment residing in the workspace of a different worker. When created, viewed, accessed, or manipulated from one of the worker sessions that contains part of the array, it is referred to as a codistributed array. Compare to distributed array.
communicating job	Job composed of tasks that communicate with each other during evaluation. All tasks must run simultaneously. A special case of communicating job is a parallel pool, used for executing <code>parfor</code> -loops and <code>spmd</code> blocks.
Composite	An object in a MATLAB client session that provides access to data values stored on the workers in a parallel pool, such as the values of variables that are assigned inside an <code>spmd</code> statement.

computer	A system with one or more processors.
distributed application	The same application that runs independently on several nodes, possibly with different input parameters. There is no communication, shared data, or synchronization points between the nodes, so they are generally considered to be coarse-grained.
distributed array	An array partitioned into segments, with each segment residing in the workspace of a different worker. When created, viewed, accessed, or manipulated from the client session, it is referred to as a distributed array. Compare to codistributed array.
DNS	Domain Name System. A system that translates Internet domain names into IP addresses.
dynamic licensing	The ability of a MATLAB worker to employ all the functionality you are licensed for in the MATLAB client, while checking out only an engine license. When a job is created in the MATLAB client with Parallel Computing Toolbox software, the products for which the client is licensed will be available for all workers that evaluate tasks for that job. This allows you to run any code on the cluster that you are licensed for on your MATLAB client, without requiring extra licenses for the worker beyond MATLAB Parallel Server software. For a list of products that are not eligible for use with Parallel Computing Toolbox software, see https://www.mathworks.com/products/ineligible_programs/ .
fine-grained application	An application for which run time is significantly less than the communication time needed to start and stop the program. Compare to coarse-grained applications.
head node	Usually, the node of the cluster designated for running the job scheduler and network license manager. It is often useful to run all the nonworker related processes on a single machine.
heterogeneous cluster	A cluster that is not homogeneous.

homogeneous cluster	A cluster of identical machines, in terms of both hardware and software.
independent job	A job composed of independent tasks, which do not communicate with each other during evaluation. Tasks do not need to run simultaneously.
job	The complete large-scale operation to perform in MATLAB, composed of a set of tasks.
job scheduler checkpoint information	Snapshot of information necessary for the MATLAB Job Scheduler to recover from a system crash or reboot.
job scheduler database	The database that the MATLAB Job Scheduler uses to store the information about its jobs and tasks.
LOGDIR	The name of the parameter in the <code>mjs_def</code> file that defines the directory where logs are stored.
MATLAB client	See client.
MATLAB Job Scheduler	The MathWorks process that queues jobs and assigns tasks to workers. Formerly known as a job manager.
MATLAB worker	See worker.
mjs	The service that has to run on all machines before they can run a MATLAB Job Scheduler or worker. This is the engine foundation process, making sure that the job scheduler and worker processes that it controls are always running. Note that the program and service name is all lowercase letters.
mjs_def file	The file that defines all the defaults for the mjs processes by allowing you to set preferences or definitions in the form of parameter values.
MPI	Message Passing Interface, the means by which workers communicate with each other while running tasks in the same job.

node	A computer that is part of a cluster.
parallel application	The same application that runs on several workers simultaneously, with communication, shared data, or synchronization points between the workers.
parallel pool	A collection of workers that are reserved by the client and running a special communicating job for execution of parfor-loops, spmd statements, and distributed arrays.
private array	An array which resides in the workspaces of one or more, but perhaps not all workers. There might or might not be a relationship between the values of these arrays among the workers.
random port	A random unprivileged TCP port, i.e., a random TCP port above 1024.
register a worker	The action that happens when both worker and MATLAB job scheduler are started and the worker contacts the job scheduler.
replicated array	An array which resides in the workspaces of all workers, and whose size and content are identical on all workers.
scheduler	The process, either local, third-party, or the MATLAB job scheduler, that queues jobs and assigns tasks to workers.
spmd (single program multiple data)	A block of code that executes simultaneously on multiple workers in a parallel pool. Each worker can operate on a different data set or different portion of distributed data, and can communicate with other participating workers while performing the parallel computations.
task	One segment of a job to be evaluated by a worker.
variant array	An array which resides in the workspaces of all workers, but whose content differs on these workers.
worker	The MATLAB session that performs the task computations. Also known as the MATLAB worker or worker process.

**worker checkpoint
information**

Files required by the worker during the execution of tasks.

