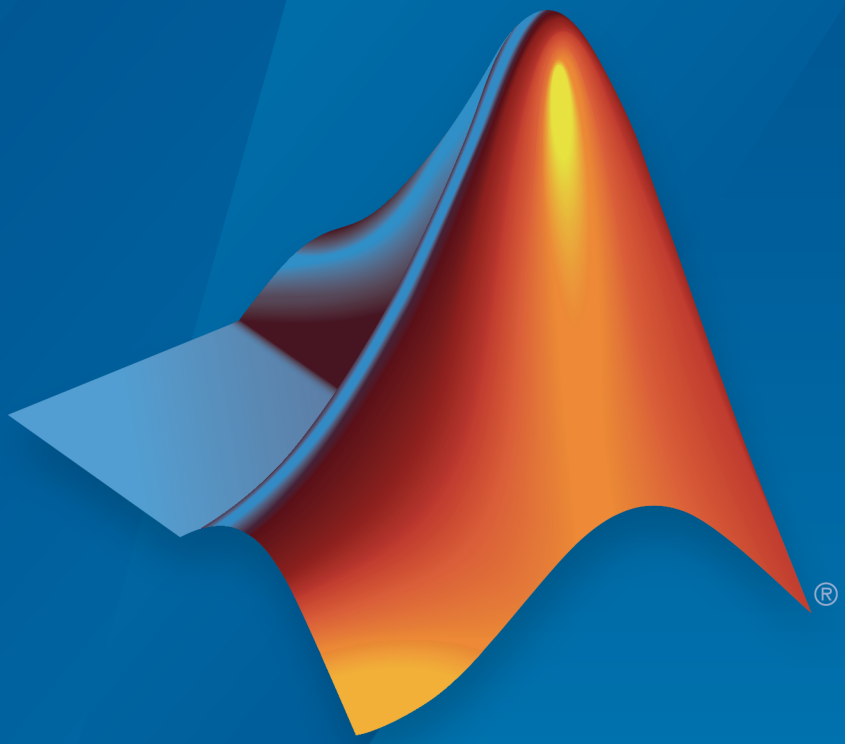
MATLAB® Distributed Computing Server™

System Administrator's Guide



MATLAB®

R2017a

MathWorks®

How to Contact MathWorks

Latest news: www.mathworks.com

Sales and services: www.mathworks.com/sales_and_services

User community: www.mathworks.com/matlabcentral

Technical support: www.mathworks.com/support/contact_us

Phone: 508-647-7000

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

MATLAB® Distributed Computing Server™ System Administrator's Guide

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)

Contents

1 Introduction

2 Network Administration

3 Product Installation

4 Admin Center

1

Introduction

MATLAB Distributed Computing Server Product Description

Perform MATLAB and Simulink computations on clusters, clouds, and grids

MATLAB Distributed Computing Server lets you run computationally intensive MATLAB programs and Simulink® models on computer clusters, clouds, and grids, enabling you to speed up computations and solve large problems. You develop your program or model on a multicore desktop computer using Parallel Computing Toolbox and then scale up to many computers by running it on MATLAB Distributed Computing Server. The server supports batch jobs, parallel computations, and distributed large data. The server includes a built-in cluster job scheduler and provides support for commonly used third-party schedulers.

MATLAB Distributed Computing Server enables you to run your programs and models on a cluster without having to acquire additional MathWorks® product licenses for each computer in the cluster.

Key Features

- Access to all eligible licensed toolboxes or blocksets with a single server license on the distributed computing resource
- Execution of GPU-enabled functions on distributed computing resources
- Execution of parallel computations from applications and software components generated using MATLAB Compiler™ on distributed computing resources
- Support for all hardware platforms and operating systems supported by MATLAB and Simulink
- Application scheduling using a built-in job scheduler or third-party schedulers such as Platform LSF®, Microsoft® Windows® HPC Server 2008, Altair PBS Pro®, and TORQUE

Product Overview

In this section...

Parallel Computing Concepts

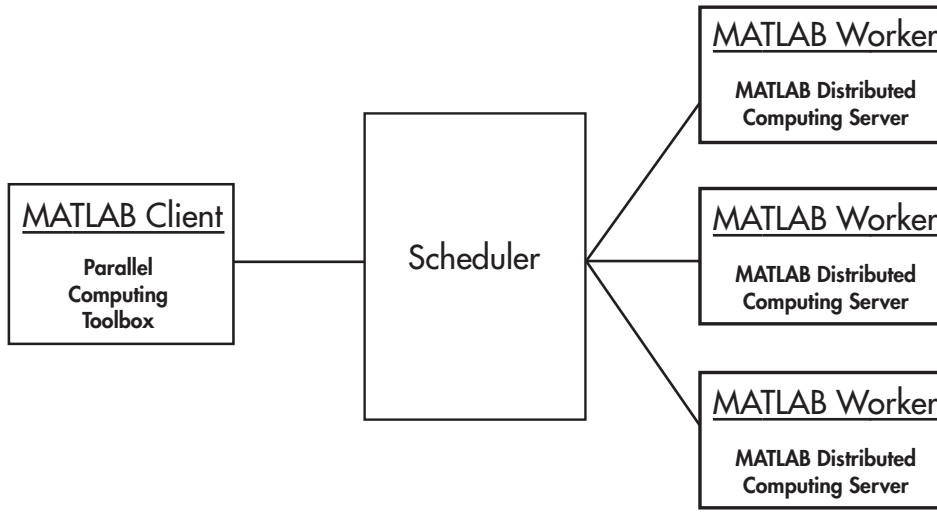
A *job* is some large operation that you need to perform in your MATLAB session. A job is broken down into segments called *tasks*. You decide how best to divide your job into tasks. You could divide your job into identical tasks, but tasks do not have to be identical.

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 Distributed Computing 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 Distributed Computing Server software allows you to run as many MATLAB workers on a remote cluster of computers as your licensing allows.

The MATLAB job scheduler (MJS) is the part of the server software that coordinates the execution of jobs and the evaluation of their tasks. The MJS distributes the tasks for evaluation to the server's individual MATLAB sessions called *workers*. Use of the MJS 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 Distributed Computing 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...

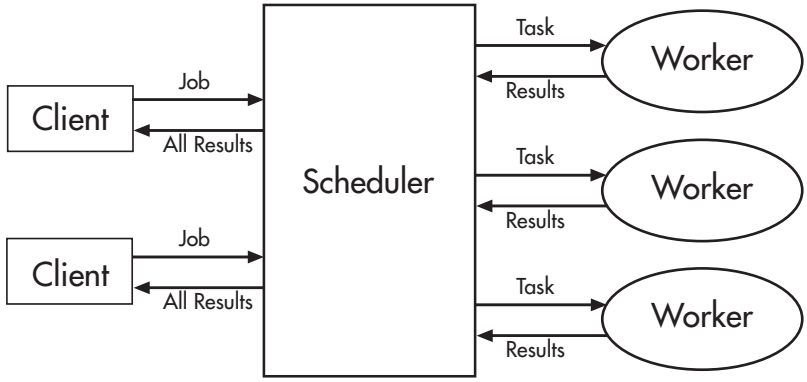
Schedulers, Workers, and Clients

The optional MJS can run on any machine on the network. The MJS 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 MJS, executes the task, returns the result to the MJS, and then receives another task. When all tasks for a running job have been assigned to workers, the MJS starts running the next job with the next available worker.

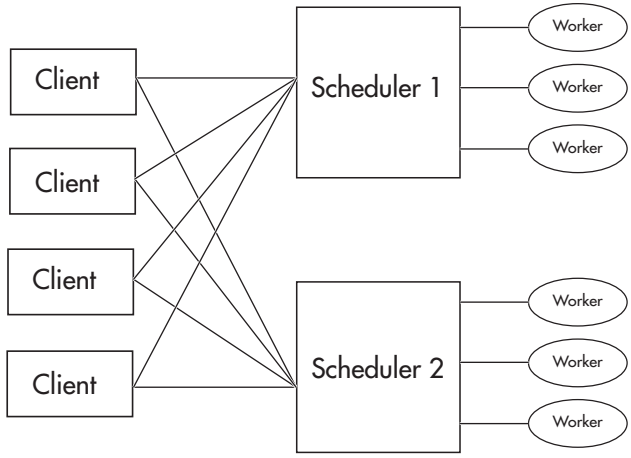
A MATLAB Distributed Computing Server network configuration usually includes many workers that can all execute tasks simultaneously, speeding up execution of large MATLAB jobs. It is generally not important which worker executes a specific task. Each worker evaluates tasks one at a time, returning the results to the MJS. The MJS 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 MJS. You can also have more than one worker session or more than one MJS session on a machine.



Interactions of Parallel Computing Sessions

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



Configuration with Multiple Clients and MJS Processes

Third-Party Schedulers

As an alternative to using the MJS, 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 MJS

You should consider the following when deciding to use a scheduler or the MJS 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 an MJS, 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 MJS 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 MJS 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 an MJS, worker processes usually remain running at all times, dedicated to their MJS. 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 MJS.

- 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 Distributed Computing Server software are supported on Windows, UNIX® (including Linux®), and Macintosh operating systems. Mixed platforms are supported, so that the clients, MJS, 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 Distributed Computing Server software at

```
http://www.mathworks.com/products/distriben/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.

mdce Service

If you are using the MJS, every machine that hosts a worker or MJS session must also run the mdce service.

The mdce service recovers worker and MJS sessions when their host machines crash. If a worker or MJS machine crashes, when mdce starts up again (usually configured to start at machine boot time), it automatically restarts the MJS 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 MJS available (but usually only one scheduler). The function you use to find an MJS or scheduler creates an object in your current MATLAB session to represent the MJS 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 MJS (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 MJS or scheduler. The MJS 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 Distributed Computing Server software.

Prepare for Parallel Computing

In this section...

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

Plan Your Network Layout

Generally, it is easy to decide which machines will run worker processes and which will run client processes. Worker sessions usually run on the cluster of machines dedicated to that purpose. The MATLAB client session usually runs where MATLAB programs are run, often on a user's desktop.

The job manager process should run on a stable machine, with adequate resources to manage the number of tasks and amount of data expected in your parallel computing applications.

The following table shows what products and processes are needed for each of these roles in the parallel computing configuration.

Session	Product	Processes
Client	Parallel Computing Toolbox	MATLAB with toolbox
Worker	MATLAB Distributed Computing Server	worker; mdce service (if you are using a job manager)
Job manager	MATLAB Distributed Computing Server	mdce service; job manager

The server software includes the mdce service or daemon. The mdce service is separate from the worker and job manager processes, and it must be running on all machines that run job manager sessions or workers that are registered with a job manager. (The mdce service is not used with third-party schedulers.)

You can install both toolbox and server software on the same machine, so that one machine can run both client and server sessions.

Network Requirements

To view the network requirements for MATLAB Distributed Computing Server software, visit the product requirements page on the MathWorks Web site at

`http://www.mathworks.com/products/distriben/requirements.html`

Fully Qualified Domain Names

MATLAB Distributed Computing Server software and Parallel Computing Toolbox software support both short host names and fully qualified domain names. The default usage is short host names. If your network requires fully qualified host names, you can use the `mdce_def` file to identify the worker nodes by their full names. See "Customize Startup Parameters" on page 2-12. To set the host name used for a MATLAB client session, see the `pctconfig` reference page.

Security Considerations

The parallel computing products do not provide any security measures. Therefore, be aware of the following security considerations:

- MATLAB workers run as whatever user the administrator starts the node's mdce service under. By default, the mdce 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 mdce service does not enforce any access control or authentication. Anyone with local or remote access to the mdce 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. 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 2-12.
- 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.

Use Different MPI Builds on UNIX Systems

In this section...

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 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 `sge/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...

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 mdce Daemon

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

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

---

1. Use the following command to stop the mdce daemon:

   ```
   /etc/init.d/mdce stop
   ```

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

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

Stop the Daemon Manually

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

```
cd matlabroot/toolbox/distcomp/bin
mdce 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 mdce Service

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

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

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

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

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

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

Customize Startup Parameters

In this section...

The MATLAB Distributed Computing 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 `mdce_def` file.

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

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

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

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

The `mdce_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 `mdce_def` file.

Parameter	Description
MDCEUSER	Set this parameter to run the mdce 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.
MDCEPASS	On a Windows operating system, set this parameter to specify the password for the user identified in the MDCEUSER parameter; otherwise, the system prompts you for the password when the service is installed.

On UNIX operating systems, MDCEUSER 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 MDCEUSER. For further information, refer to your system documentation on the sudo and sudoers utilities (for example, man sudo and man sudoers).

The MDCEUSER 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 MDCEUSER. 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 mdce service, job managers, and workers are defined in the file:

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

Before installing and starting the mdce 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

```
mdce start -mdcedef my_mdce_def.sh
```

On Windows operating systems, enter the command

```
mdce install -mdcedef my_mdce_def.bat
mdce start -mdcedef my_mdce_def.bat
```

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

For more information, see “Define Script Defaults” on page 2-12.

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

The MATLAB Distributed Computing Server services generate various record files in the normal course of their operations. The mdce 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	The default location of the log files is `<TEMP>\MDCE\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\MDCE\Log`. You can set alternative locations for the log files by modifying the `LOGBASE` setting in the `mdce_def.bat` file before starting the mdce service.
Linux and Macintosh	The default location of the log files is `/var/log/mdce/`. You can set alternative locations for the log files by modifying the `LOGBASE` setting in the `mdce_def.sh` file before starting the mdce service.

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	The default location of the checkpoint folders is `<TEMP>\MDCE\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\MDCE\Checkpoint`. You can set alternative locations for the checkpoint folders by modifying the `CHECKPOINTBASE` setting in the `mdce_def.bat` file before starting the mdce service.
Linux and Macintosh	The checkpoint folders are placed by default in `/var/lib/mdce/`. You can set alternative locations for the checkpoint folder by modifying the `CHECKPOINTBASE` setting in the `mdce_def.sh` file before starting the mdce service.

Set MJS Cluster Security

In this section...

Set the Security Level

You set the job manager or MJS security level with the `SECURITY_LEVEL` parameter in the `mdce_def` file before starting the mdce service on your cluster nodes. The `mdce_def` file indicates what values are allowed, and briefly describes each security level.

The following table describes the available security levels for accessing an MJS and its jobs.

Security Level	Description	User Requirements
0	No security. • Any user can access any job. • Tasks run as the user who started the mdce process on the worker machines (typically root or Local System). • This is the default, and is the behavior in all releases prior to R2010b.	• Jobs are associated with the default user name of the programmer, but no protection is provided.
1	Jobs are identified with the submitting user. • 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 mdce process on the worker machines (typically root or Local System).	• A dialog requires you to establish a user name when you first access the job manager. • Your job manager (MJS) user name does not have to match your system/ network user name. • No passwords are used.
2	Job manager (MJS) password protection on jobs.	• When you start the job manager (MJS), it prompts you to provide a

Security Level	Description	User Requirements
	• 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 mdce process on the worker machines (typically root or Local System).	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 job manager (MJS) from the MATLAB client. • Your job manager (MJS) user name and password do not have to match your system/network user name and password.

Security Level	Description	User Requirements
3	In addition to the security of level 2, tasks run as the submitting user on worker machines. • 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.	• On UNIX systems, the mdce process on the cluster nodes must be started by the root user. • The job manager (MJS) must use secure communication with the workers (set in the `mdce_def` file). • When you start the job manager (MJS), 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 job manager (MJS) from the MATLAB client. • Your job manager (MJS) 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, MJS, and Network Passwords

For any security above level 0, when you start the MJS (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, the job manager (MJS) identifies every job with the user who submits the job. Therefore, whenever you access the MJS or a job, the MJS must be aware of who you are.

At security level 0, the MJS 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 MJS 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 MJS 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 MJS. 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 an MJS, see `changePassword` and `logout`.

Set Secure Communication

To establish secure encrypted communication between job manager (MJS) and workers, set the `USE_SECURE_COMMUNICATION` parameter in the `mdce_def` file.

You must also provide a value for the `SHARED_SECRET_FILE` parameter in the `mdce_def` file, identifying where the file can be found from the job manager (MJS) 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 `mdce_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 `mdce_def` file indicates where its particular secret file is located.

Note Secure communication is required when using job manager (MJS) security level 3.

Troubleshoot Common Problems

In this section...

This section offers advice on solving problems you might encounter with MATLAB Distributed Computing 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 your 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 Distributed Computing 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 mdce 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/mdce_user/node_node_worker_05-11-01_16-52-03_953.log
   for more detailed information.  The mdce log file
   /tmp/mdce_user/mdce-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:
http://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 Distributed Computing 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 mdce process under a different user with rights to log on as a service. See "Set the User" on page 2-12.

Required Ports

With Job Manager

BASE_PORT

The `mdce_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/mdce_def.sh` (on UNIX operating systems)
- *`matlabroot`*`\toolbox\distcomp\bin\mdce_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 (MJS, or job manager), 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 Distributed Computing 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 4-10). 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 2-28
- "Multicast" on page 2-29

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 WWWW 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 `WWWW` 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 MJS 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 MJS. For example, `mjs-1.company.com`.

A valid DNS SRV record for the `company.com` network running an MJS 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 (MJS) 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
- `winpcap` 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;
```

3

Product Installation

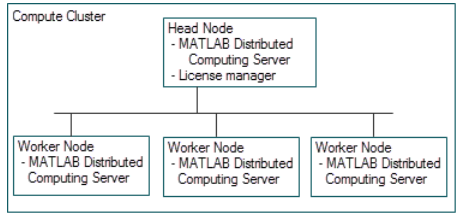
Install Products and Choose Cluster Configuration

In this section...

Cluster Description

To set up a cluster, you first install MATLAB Distributed Computing Server on a node called the *head* node. You can also install the license manager on the head node. After performing this installation, you can then optionally install MATLAB Distributed Computing Server on the individual cluster nodes, called *worker* nodes. You do not need to install the license manager on worker nodes.

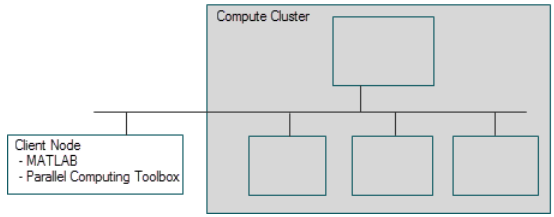
This figure shows the installations that you perform on your cluster nodes. This is only one possible configuration. (You can install the cluster license manager and MATLAB Distributed Computing Server on separate nodes, but this document does not cover this type of installation.)



Product Installations on Cluster Nodes

You install Parallel Computing Toolbox software on the computer that you use to write MATLAB applications. This is called the *client* node.

This figure shows the installations that you must perform on client nodes.



Product Installations on Client Nodes

Your installation on the client node might include other MathWorks products.

Install Products

On the Cluster Nodes

Install the MathWorks products on your cluster as a network installation. You can install in a central location, or individually on each cluster node.

If you need help with this step, you can find instructions for this release at "Installation, Licensing, and Activation" in the Documentation Center. These instructions include steps for installing, licensing, and activating your installation.

Note MathWorks highly recommends installing all MathWorks products on the cluster. MATLAB Distributed Computing Server cannot run jobs whose code requires products that are not installed.

Backwards Compatibility Note You can upgrade your MATLAB Job Scheduler clusters and continue to use the R2016a release of Parallel Computing Toolbox on your MATLAB Desktop client to connect to it. This only applies to the R2016a release onward. You must install the same release of MATLAB Distributed Computing Server for each release of MATLAB you want to support. You can configure MATLAB Job Scheduler with the location of these installations in the `mdce_def` file.

To set up a MATLAB Job Scheduler cluster for R2016b or following releases:

- Install MATLAB Distributed Computing Server for each release that the cluster supports. For example, to use R2016a and R2016b with your cluster, install both the R2016a and R2016b releases of MATLAB Distributed Computing Server.

- Specify the R2016a installation of MATLAB Distributed Computing Server in the `MDCS_ADDITIONAL_MATLABROOTS` variable in the `mdce_def` file. This file is provided in *`matlabroot`*`/toolbox/distcomp/bin` for Linux (`mdce_def.sh`) and Windows (`mdce_def.bat`). For more information, see `mdce`.

On the Client Nodes

On the client computers from which you will write applications to submit jobs to the cluster, install the MathWorks products for which you are licensed, including Parallel Computing Toolbox.

If you need help with this step, you can find instructions for the current release at "Installation, Licensing, and Activation" in the Documentation Center. These instructions include steps for installing, licensing, and activating your installation.

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 one of the following chapters in this document to complete your configuration and to test the installation:

- "Configure for an MJS" on page 3-5
- "Configure for HPC Server" on page 3-28
- "Configure for PBS Pro, Platform LSF, TORQUE" on page 3-34
- "Configure for a Generic Scheduler" on page 3-40
- "Configure a Hadoop Cluster" on page 3-55

Note You must use the generic scheduler interface for any of the following:

- Schedulers not supported by the direct integration (PBS Pro, Torque, LSF, HPC Server)
- A nonshared file system when you want to use`ssh` as a submission tool through a submission host

---

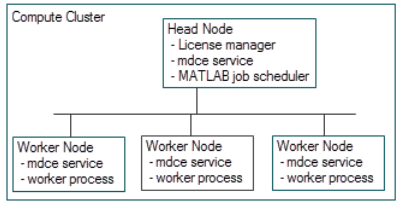
Configure for an MJS

In this section...

Configure Cluster to Use a MATLAB Job Scheduler (MJS)

The mdce service must be running on all machines being used for MATLAB job schedulers (MJS) or workers. This service manages the MJS and worker processes. One of the major tasks of the mdce service is to recover the MJS and worker sessions after a system crash, so that jobs and tasks are not lost as a result of such accidents.

The following figure shows the processes that run on your cluster nodes.



Note The MATLAB job scheduler (MJS) was formerly known as the MathWorks job manager. The process is the same, is started in the same way, and performs the same functions.

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

Step 1: Set Up Windows Cluster Hosts

If this is the first installation of MATLAB Distributed Computing 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 Distributed Computing Server on your Windows cluster, you can skip this step and proceed to Step 2.

Configure Windows Firewalls

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 User Access to Installation

The user that mdce runs as requires access to the cluster MATLAB installation location. By default, mdce runs as the user `LocalSystem`. If your network allows `LocalSystem` to access the install location, you can proceed to the next 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 mdce as a different user.

You can set a different user with these steps:

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

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

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

   ```
   set MDCEUSER=mydomain\myusername
   ```

3. Provide the user password by setting the `MDCEPASS` parameter:

   ```
   set MDCEPASS=password
   ```

4 Save the file. Proceed to the next step.

Step 2: Stop mdce Services of Old Installation

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

- "Stop mdce on Windows" on page 3-7
- "Stop mdce on UNIX" on page 3-8

Stop mdce on Windows

If this is your first installation of the parallel computing products, proceed to "Step 3: Start the mdce Service, MJS, and Workers" on page 3-8.

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 mdce service and remove its associated files by typing the command:

   ```
   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 mdce on UNIX

1. Log in as root. (If you cannot log in as root, you must alter the following parameters in the *matlabroot*/toolbox/distcomp/bin/mdce_def.sh file to point to a folder for which you have write privileges: CHECKPOINTBASE, LOGBASE, PIDBASE, and LOCKBASE if applicable.)
2. **On each cluster node**, stop the mdce service and remove its associated files by typing the commands:

```
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.

Step 3: Start the mdce Service, MJS, and Workers

You can start the MJS (job manager) by using a GUI or the command line. Choose one:

- "Using Admin Center GUI" on page 3-8
- "Using the Command-Line Interface (Windows)" on page 3-16
- "Using the Command-Line Interface (UNIX)" on page 3-19

Using Admin Center GUI

Note To use 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, follow the instructions in "Using the Command-Line Interface (Windows)" on page 3-16 or "Using the Command-Line Interface (UNIX)" on page 3-19.

1. **Identify Hosts and Start the mdce Service**

   a. To open Admin Center, navigate to the folder:

   matlabroot\toolbox\distcomp\bin (on Windows)

   matlabroot/toolbox/distcomp/bin (on UNIX)

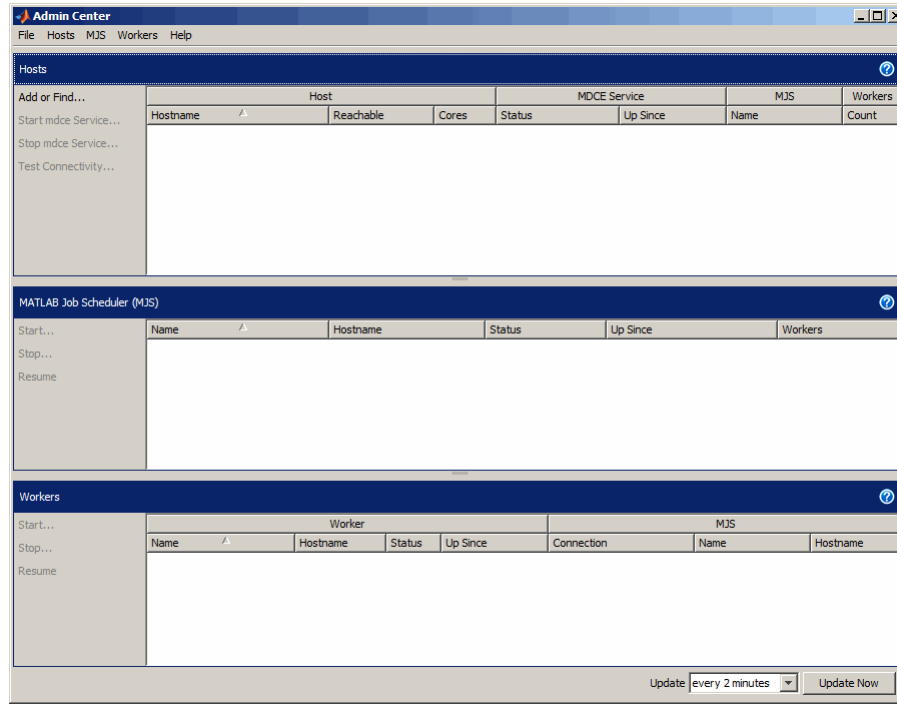
   Then execute the file:

`admincenter.bat` (on Windows)

`admincenter` (on UNIX)

Note To start the mdce service on remote machines from Admin Center, requires that you run Admin Center as a user who has administrator privileges on all the machines.

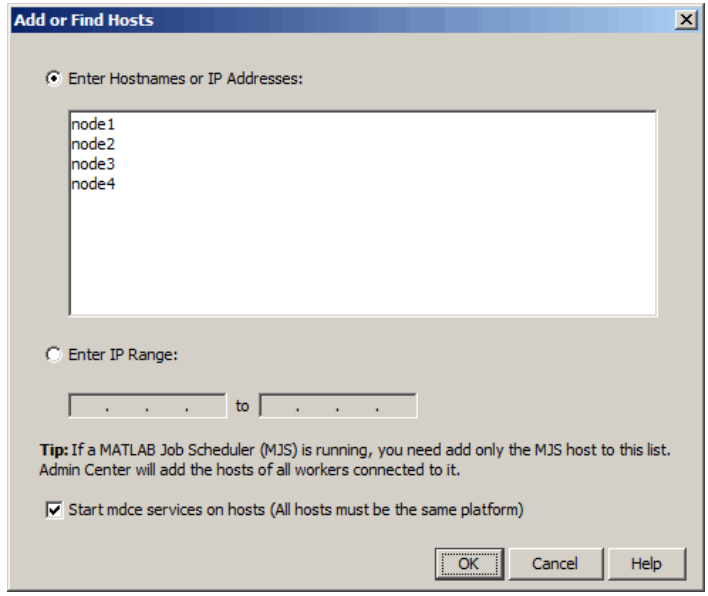
If there are no past sessions of Admin Center saved for you, the GUI opens with a blank listing, superimposed by a welcome dialog box, which provides information on how to get started.



b Click **Add or Find**.

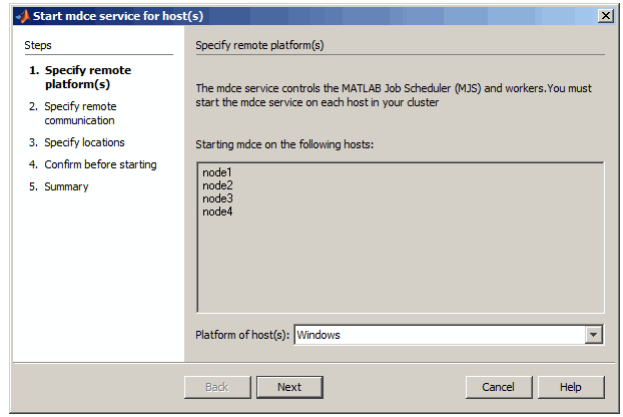
The Add or Find Hosts dialog box opens.

c Select **Enter Hostnames**, then list your hosts in the text box. You can use short host names, fully qualified domain names, or individual IP addresses. The following figure shows an example using host names `node1`, `node2`, `node3`, and `node4`. In your case, use your own host names.

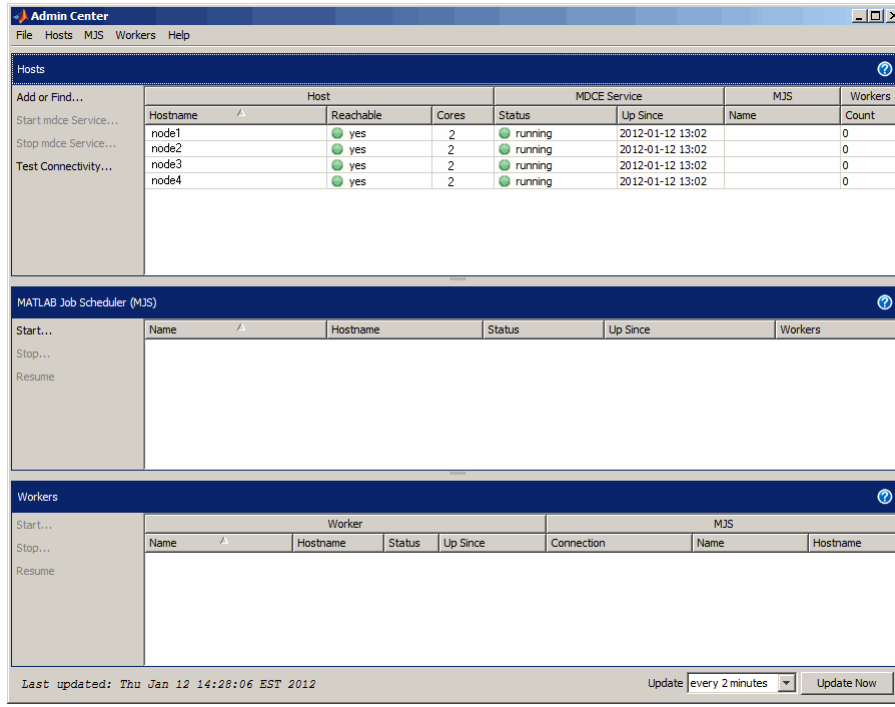


Keep the check to start mdce service.

d Click **OK** to open the Start mdce service dialog box. Proceed through the steps clicking **Next** and checking the settings at each step. For most settings, the default is appropriate.



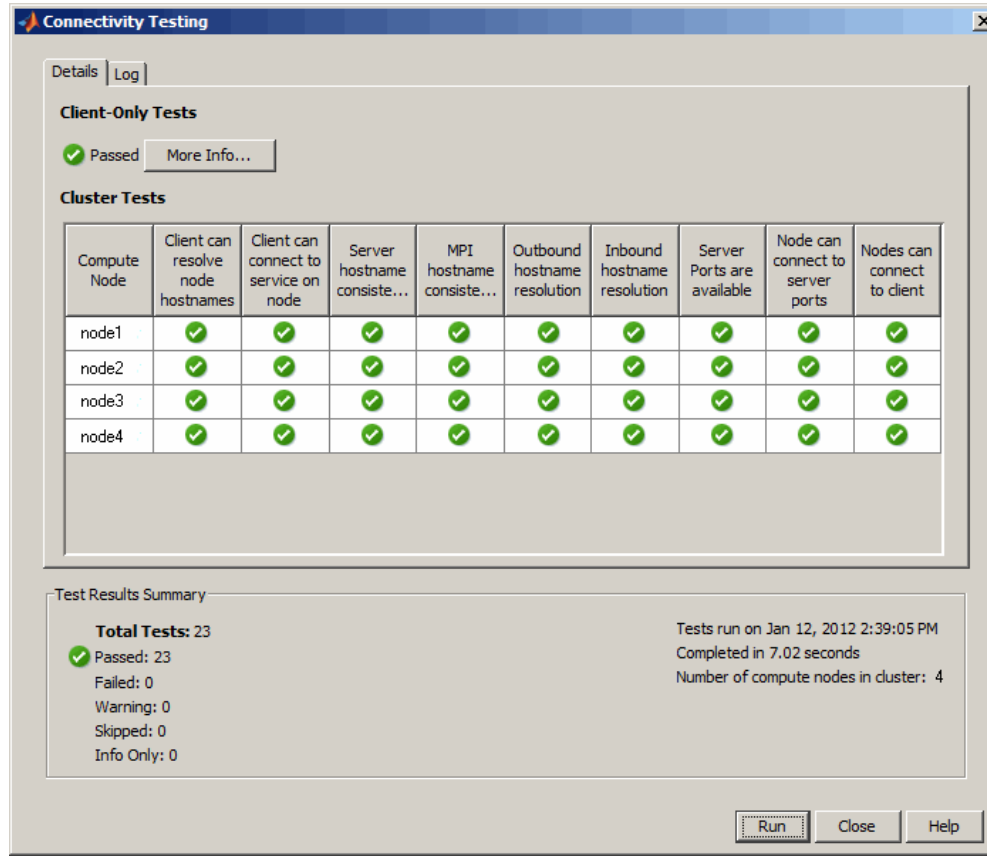
It might take a moment for Admin Center to communicate with all the nodes, start the services, and acquire the status of all of them. When Admin Center completes the update, the listing should look something like the following figure.



e At this point, you should test the connectivity between the nodes. This assures that your cluster can perform the necessary communications for running other MATLAB Distributed Computing Server processes.

In the Hosts module, click **Test Connectivity**.

f When the Connectivity Testing dialog box opens, it shows the results of the last test, if there are any. Click **Run** to run the tests and generate new data.



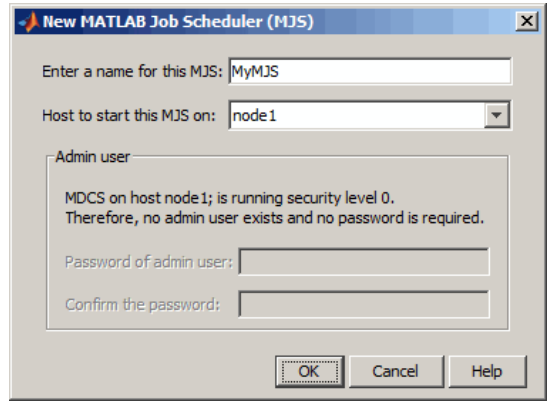
If any of the connectivity tests fail, double-click the icon that indicates a failure to get information about that specific test; or use the **Log** tab to get all test results. With this information, you can refer to "Troubleshoot Common Problems" on page 2-22. If you need further help, contact the MathWorks install support team.

g If your tests pass, click **Close** to return to the Admin Center GUI.

2 Start the MJS

a To start an MJS (job manager), click **Start** in the MJS module. (This is one of several ways to open the New MJS dialog box.) In the New MJS dialog box,

specify a name and host for your MJS. This example shows an MJS called `MyMJS` to run on host `node1`.



b Click **OK** to start the MJS and return to the Admin Center GUI.

3 Start the Workers

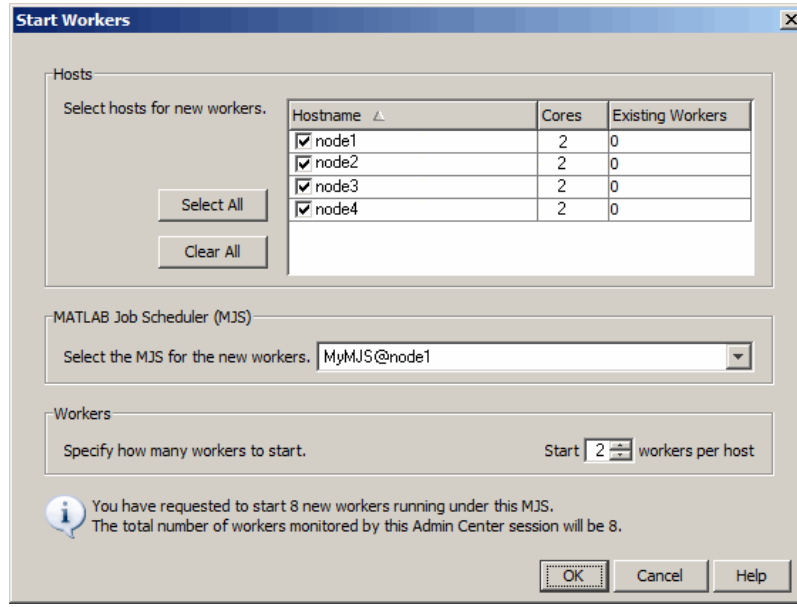
a To start workers, click **Start** in the Workers module. (This is one of several ways to open the Start Workers dialog box.)

b In the Start Workers dialog box, specify the number of workers to start on each host. The number is up to you, but you cannot exceed the total number of licenses you have. A good starting value might be to start one worker per computational core on your hosts.

c Select the hosts to start the workers on. Click **Select All** if you want to start workers on all listed hosts.

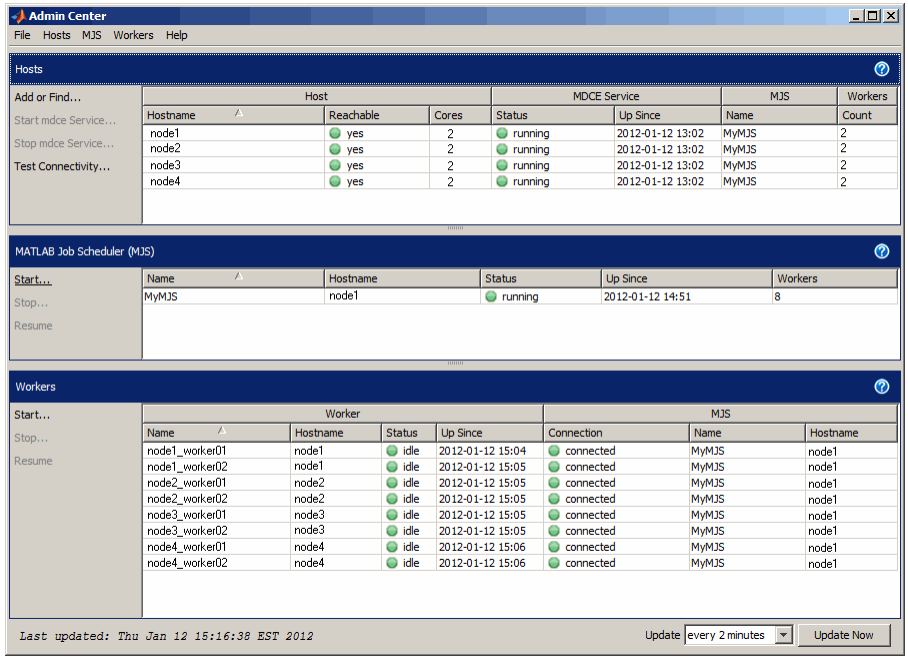
d Select the MJS for your workers. If you have only one MJS running in this Admin Center session, that is the default.

The following example shows a setup for starting eight workers on four hosts (two workers each). Your names and numbers will vary.



e Click OK to start the workers and return to the Admin Center dialog box. It might take a moment for Admin Center to initialize all the workers and acquire their status.

When all the workers are started, Admin Center looks something like the following figure. If your workers are all idle and connected, your cluster is ready for use.



If you encounter any problems or failures, contact the MathWorks install support team.

For more information about Admin Center functionality, such as stopping processes or saving sessions, see “Cluster Processes and Profiles”.

Using the Command-Line Interface (Windows)

1 Start the mdce Service

You must install the mdce 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 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).

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 mdce service by typing the command:

```
mdce install
```

d Start the mdce service by typing the command:

```
mdce 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 mdce service on several nodes remotely from one machine by typing:

```
cd matlabroot\toolbox\distcomp\bin
remotemdce install -remotehost hostA,hostB,hostC . . .
remotemdce 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 `remotemdce` by typing:

```
remotemdce -help
```

Once installed, the mdce service starts running each time the machine reboots. The mdce service continues to run until explicitly stopped or uninstalled, regardless of whether an MJS or worker session is running.

2 Start the MJS

To start the MATLAB job scheduler (MJS), enter the following commands in a DOS command window. You do not have to be at the machine on which the MJS runs, as long as you have access to the MATLAB Distributed Computing Server installation.

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

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

b Start the MJS, using any unique text you want for the name <MyMJS>:

```
startjobmanager -name <MyMJS> -remotehost <MJS host name> -v
```

c Verify that the MJS is running on the intended host.

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

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

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

3 Start the Workers

Note Before you can start a worker on a machine, the mdce service must already be running on that machine, and the license manager for MATLAB Distributed Computing Server 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 Distributed Computing 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 MJS you want this worker registered with. Enter this text on a single line:

```
startworker -jobmanagerhost <MJS 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 mdce, MJS, and worker processes, such as how to shut them down or customize them, see "MJS Cluster Customization".

Using the Command-Line Interface (UNIX)

1 Start the mdce Service

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

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

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

```
cd matlabroot/toolbox/distcomp/bin
./remotemdce 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 `remotemdce` by typing

```
./remotemdce -help
```

2 Start the MJS

To start the MATLAB job scheduler (MJS), enter the following commands. You do not have to be at the machine on which the MJS runs, as long as you have access to the MATLAB Distributed Computing Server installation.

a Navigate to the folder with the startup scripts:

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

b Start the MJS, using any unique text you want for the name `<MyMJS>`. Enter this text on a single line.

```
./startjobmanager -name <MyMJS> -remotehost <MJS host name> -v
```

c Verify that the MJS is running on the intended host:

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

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

3 Start the Workers

Note Before you can start a worker on a machine, the mdce service must already be running on that machine, and the license manager for MATLAB Distributed Computing Server 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 Distributed Computing 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 MJS you want this worker registered with. Enter this text on a single line:

```
./startworker -jobmanagerhost <MJS 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 mdce, MJS, and worker processes, such as how to shut them down or customize them, see “MJS Cluster Customization”.

Step 4: Install the mdce 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 mdce service starts running each time the machine reboots. The mdce service continues to run until explicitly stopped, regardless of whether an MJS or worker session is running.

You must have root privileges to do this step.

Choose your platform:

- "Debian, Fedora Platforms" on page 3-21
- "SUSE Platform" on page 3-21
- "Red Hat Platform (non-Fedora)" on page 3-22
- "Macintosh Platform" on page 3-22

Debian, Fedora Platforms

On each cluster node, register the mdce 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/mdce /etc/mdce
   ```

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

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

3. Set the boot script file permissions:

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

4. Look in `/etc/inittab` for the default run level. Create a link in the `rc` folder associated with that run level. For example, if the run level is `5`, execute these commands:

   ```
   cd /etc/rc5.d;
   ln -s ../init.d/mdce S99MDCE
   ```

SUSE Platform

On each cluster node, register the mdce 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/mdce /etc/mdce
```

2 Create the following link to the boot script file:

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

3 Set the boot script file permissions:

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

4 Look in `/etc/inittab` for the default run level. Create a link in the `rc` folder associated with that run level. For example, if the run level is `5`, execute these commands:

```
cd /etc/init.d/rc5.d;
ln -s ../mdce S99MDCE
```

Red Hat Platform (non-Fedora)

On each cluster node, register the mdce 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/mdce /etc/mdce
```

2 Create the following link to the boot script file:

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

3 Set boot script file permissions:

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

4 Look in `/etc/inittab` for the default run level. Create a link in the `rc` folder associated with that run level. For example, if the run level is `5`, execute these commands:

```
cd /etc/rc.d/rc5.d;
ln -s ../../init.d/mdce S99MDCE
```

Macintosh Platform

On each cluster node, register the mdce 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 mdce service:

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

```
sudo ./mdce stop
```

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

```
sudo ln -s matlabroot/toolbox/distcomp/bin/mdce /usr/sbin/mdce
```

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

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

4 Start `mdce` and observe that it starts inside launchd:

```
sudo ./mdce start
```

The command output should read:

```
Starting the MATLAB Distributed Computing Server using launchctl.
```

Configure Windows Firewalls on Client

If you are using Windows firewalls on your client node,

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.

Configure Firewalls on Server

The `mdce` 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 `mdce` service reserves a total of `6+nJ+3*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 `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 – 27405 for the mdce service.
- 28350 – 28382 for MPI communication.

Some operating systems are reluctant to immediately free TCP ports from the TIME_WAIT state for use by the same or other processes. Therefore you must allow unfirewalled communication on 2*nW ports for MPI communications.

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 MJS cluster profile. You must do this in the form Hostname:BASE_PORT, for example myMJSHost:44001.

Validate Installation with MJS

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

Step 1: Verify the Cluster Connection

To verify the network connection from the client computer to the MJS computer, follow these instructions.

Note In these instructions, *matlabroot* refers to the folder where MATLAB is installed on the client computer. Do not confuse this with the MATLAB Distributed Computing Server cluster computers.

---

1. On the client computer where Parallel Computing Toolbox is installed, open a DOS command window (for Windows software) or a shell (for UNIX software) and go to the control script folder.

   ```
   cd matlabroot\toolbox\distcomp\bin (for Windows)
   cd matlabroot/toolbox/distcomp/bin (for UNIX)
   ```

2. Run nodestatus to verify your cluster communications. Substitute <MJS Host> with the host name of your MJS computer.

   ```
   nodestatus -remotehost <MJS Host>
   ```

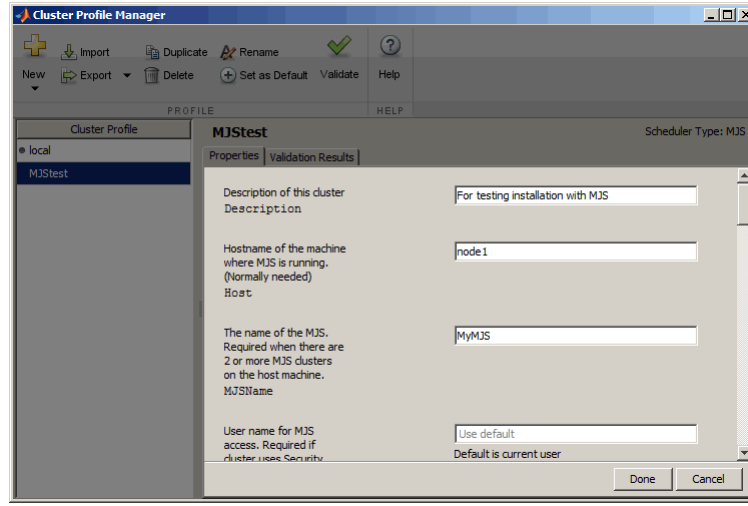
   If successful, you should see the status of your MJS (job manager) and its workers. Otherwise, refer to “Troubleshoot Common Problems” on page 2-22.

Step 2: 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 > MATLAB Job Scheduler (MJS)**.
3. With the new profile selected in the list, click **Rename** and edit the profile name to be `MJStest`. Press **Enter**.
4. In the Properties tab, provide settings for the following fields:
   a. Set the **Description** field to `For testing installation with MJS`.
   b. Set the **Host** field to the name of the host on which your MJS is running. Depending on your network, this might be only a host name, or it might have to be a fully qualified domain name.
   c. Set the **MJSName** field to the name of your MJS, which you started earlier.

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



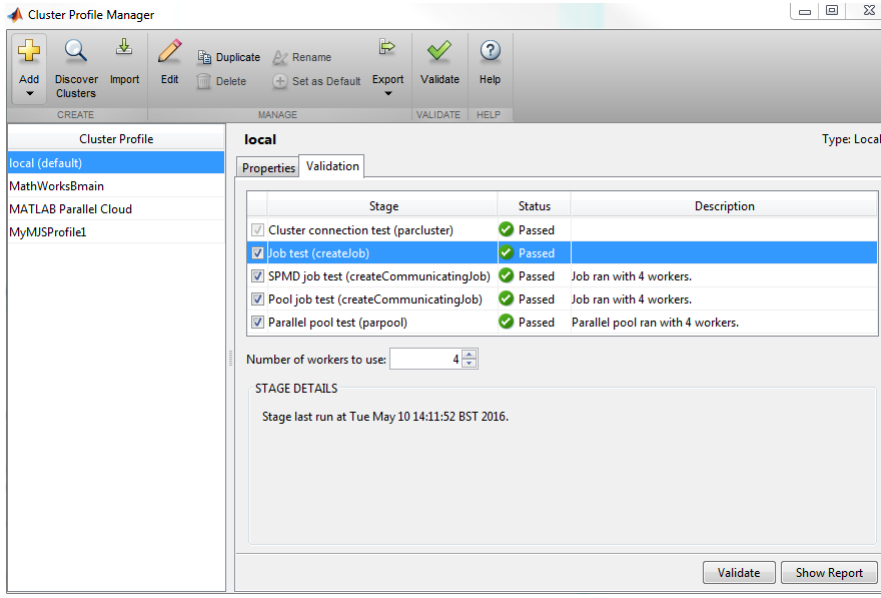
5. Click **Done** to save your cluster profile.

Step 3: Validate the Cluster Profile

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. In the case of MJS, you cannot specify the value of `NumWorkers` on the **Properties** tab and the whole cluster would be used. 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 for HPC Server

In this section...

Configure Cluster for Microsoft Windows HPC Server

Follow these instruction to configure your MATLAB Distributed Computing Server installation to work with Windows HPC Server or Compute Cluster Server (CCS). In the following instructions, *`matlabroot`* refers to the MATLAB installation location.

Note If you are using an HPC Server 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 Server

This configuring applies to all versions of HPC Server.

Note If you are using an HPC Server 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 Server 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 http://www.microsoft.com/hpc/en/us/default.aspx.

Validate Installation Using Microsoft Windows HPC Server

This procedure verifies that your parallel computing products are installed and configured correctly for using Microsoft Windows HPC Server 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 Windows HPC Server" on page 3-28, 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.

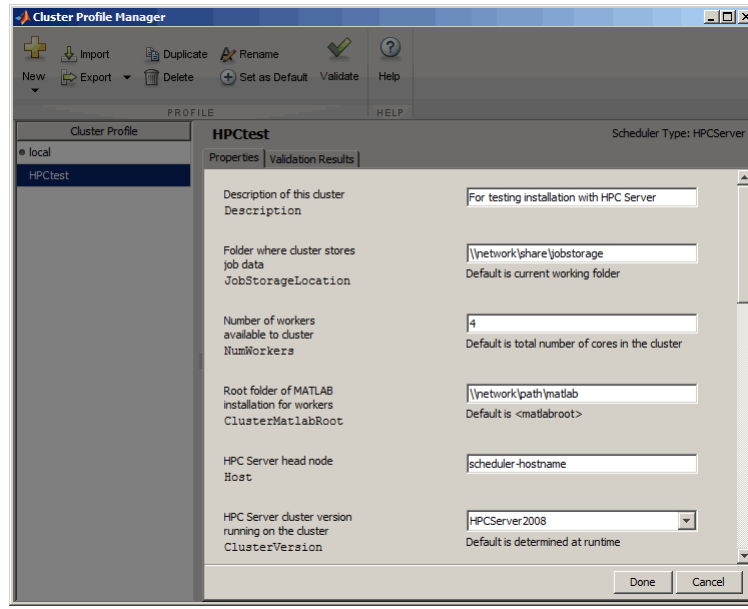
      Note `JobStorageLocation` should not be shared by parallel computing products running different versions; each version on your cluster should have its own `JobStorageLocation`.

   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`. Otherwise leave the setting `Use default` or `false`. If you plan on using SOA job submissions with your cluster, you should test this first without SOA submission, then later return and test it with SOA job submission.

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



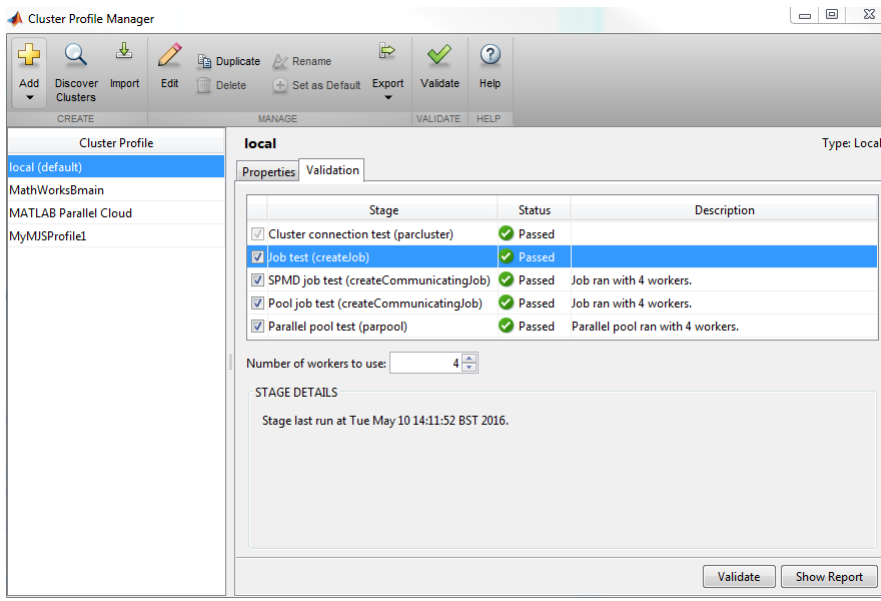
5 Click **Done** to save your cluster profile.

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.



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 PBS Pro, Platform LSF, TORQUE

In this section...

Note You must 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)

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 MPICH2 home page at http://www.mcs.anl.gov/research/projects/mpich2/. 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 pathname to the executable. Microsoft Windows Vista does not support the `smpd` executable on network share installations, so with Vista the installation must be local.

Choose one of the following configurations:

- "Without Delegation" on page 3-35
- "Using Passwordless Delegation" on page 3-35

Without Delegation

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 Distributed Computing Server software (for example, if you are running MATLAB Distributed Computing 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.

Using Passwordless Delegation

1. Log in as a user with administrator privileges.

2 Start smpd by typing in a DOS command window:

```
matlabroot\bin\win64\smpd -register_spn
```

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 Distributed Computing Server software (for example, if you are running MATLAB Distributed Computing 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 for similar accommodation.

5 Repeat 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 an 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

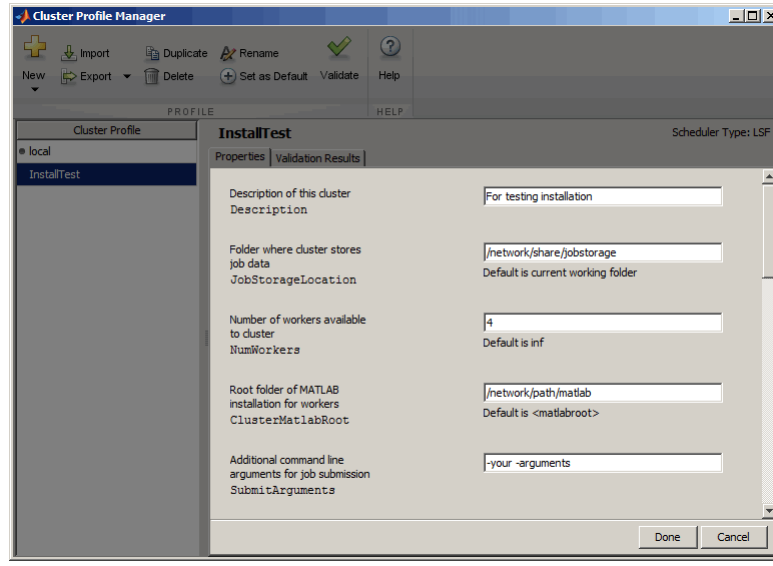
In this step you define a cluster profile to use in subsequent steps.

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 **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**.

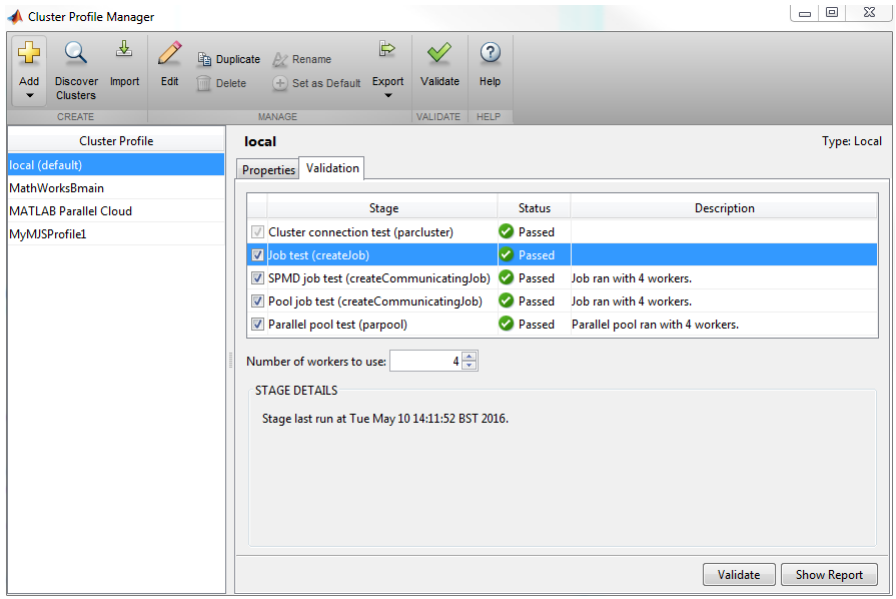
2 Select your cluster profile in the listing.

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 for a Generic Scheduler

In this section...

Note You must use the generic scheduler interface for any of the following:

- Schedulers other than those supported by the direct integration. Schedulers supported by direct integration include PBS Pro, Torque, LSF, and HPC Server.
- A nonshared file system when you want to use `ssh` as a submission tool through a submission host

From the following sections, you can select the ones that apply to your configuration:

In this section...

Interfacing with Generic Schedulers

Support Scripts

To support usage of the generic scheduler interface, templates and scripts can be installed from the following locations:

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

Each installer provides templates and scripts for the supported submission modes for shared file system, nonshared file system, or remote submission. Each submission mode has its own subfolder within the installation folder, which contains a file named `README` that provides specific instructions on how to use the scripts.

For more information on programming jobs for generic schedulers, see:

- "Program Independent Jobs for a Generic Scheduler" (Parallel Computing Toolbox)
- "Program Communicating Jobs for a Generic Scheduler" (Parallel Computing Toolbox)

Submission Mode

The provided scripts support three possible submission modes:

- Shared — When the client machine is able to submit directly to the cluster and there is a shared file system present between the client and the cluster machines.
- Remote Submission — When there is a shared file system present between the client and the cluster machines, but the client machine is not able to submit directly to the cluster (for example, if the scheduler's client utilities are not installed).
- Nonshared — When there is not a shared file system between client and cluster machines.

Before using the support scripts, decide which submission mode describes your particular network setup.

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.

Configure Generic 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. You must also use 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 MPICH2 home page at http://www.mcs.anl.gov/research/projects/mpich2/. 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 pathname to the executable. Microsoft Windows Vista does not support the `smpd` executable on network share installations, so with Vista the installation must be local.

Choose one of the following configurations:

- "Without Delegation" on page 3-42
- "Using Passwordless Delegation" on page 3-43

Without Delegation

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 Distributed Computing Server software (for example, if you are running MATLAB

Distributed Computing 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. Add MATLAB as an allowed program to your firewall. If you are using Windows firewalls on your cluster nodes, you can do this by executing 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.
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.

Using Passwordless Delegation

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

   ```
   matlabroot\bin\win64\smpd -register_spn
   ```

   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 Distributed Computing Server software (for example, if you are running MATLAB Distributed Computing 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 Add MATLAB as an allowed program to your firewall. If you are using Windows firewalls on your cluster nodes, you can do this by executing 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.

5 Repeat these steps on all Windows nodes in your cluster.

Configure Grid Engine Family on Linux Cluster

To run communicating jobs with MATLAB Distributed Computing Server and a Grid Engine family cluster, you need to establish a "matlab" parallel environment. To use this parallel environment, you must use the `matlabpe.template`, customized to match the number of slots available, and to indicate where the `startmatlabpe.sh` and `stopmatlabpe.sh` scripts are installed on your cluster.

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

Create the Parallel Environment

The following steps create the parallel environment (PE), and then make the parallel environment runnable on a particular queue. You should perform these steps on the head node of your cluster.

1 Install templates and scripts from here:

- Grid Engine family

Each submission mode has its own subfolder within the installation folder, which contains a file named `README` that provides specific instructions on how to use the scripts.

2 Modify the contents of `matlabpe.template` to use the desired number of slots 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 they can be copied to the same local 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.

3 Add the "matlab" parallel environment, using a shell command like:

```
qconf -Ap matlabpe.template
```

4 Make the “matlab” parallel environment runnable on all queues:

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

This will bring up a text editor for you to make changes: search for the line `pe_list`, and add `matlab`.

5 Ensure you can submit a trivial job to the PE:

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

6 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 filename for the output file is `~/STDIN.o###`, where `###` is the Grid Engine job number.

Note The example submit functions for Grid Engine family rely on the presence of the “matlab” parallel environment. If you change the name of the parallel environment to something other than “matlab”, you must ensure that you also change the submit functions.

Configure Firewalls on Windows Client

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

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.

Validate Installation Using a Generic Scheduler

Testing the installation of the parallel computing products with a generic scheduler requires familiarity with your network configuration, with your scheduler interface, and with the generic scheduler interface of Parallel Computing Toolbox software.

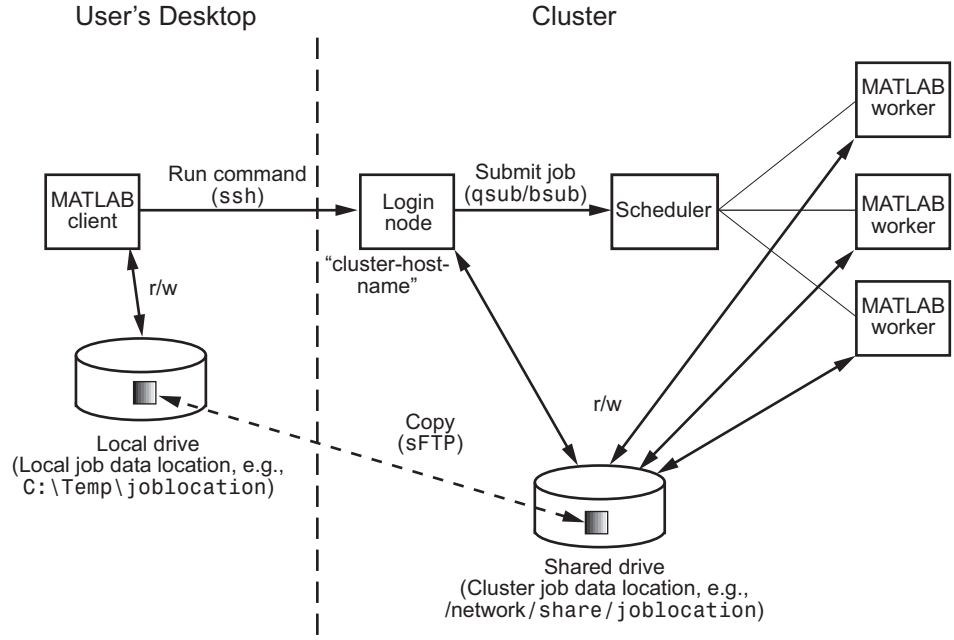
Note The remainder of this chapter illustrates only the case of using LSF in a nonshared file system. For other schedulers or a shared file system, look for the appropriate scripts and modify them as necessary, using the following instructions as a guide. If you have any questions, contact the MathWorks install support team.

Example Setup for LSF

This section provides guidelines for setting up your cluster profile to use the generic scheduler interface with an LSF scheduler in a network without a shared file system between the client the cluster machines. You can install templates and scripts for LSF from here:

- IBM Platform LSF

The scripts necessary to set up your test can be found in the nonshared subfolder within the installation folder. These scripts are written for an LSF scheduler, but might require modification to work in your network. 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

For this setup to work, the following conditions must be met:

- 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.

If these requirements are met, use the following steps to implement the solution:

Step 1: Define a Cluster Profile

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

1. Start a MATLAB session on the client host.
2. Start the Cluster Profile Manager from the MATLAB desktop by selecting **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).

      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** to the number of workers for which you want to test your installation.

d Set the **NumThreads** to the number of threads to use on each worker.

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 **RequiresMathWorksHostedLicensing** to `true` if the cluster uses MathWorks hosted licensing.

g If you have selected `true` in step `f`. , then enter your `LicenseNumber`.

h Set the **OperatingSystem** to the operating system of your cluster worker machines.

i Set **HasSharedFilesystem** to `false`, indicating that the client node and worker nodes cannot share the same data location.

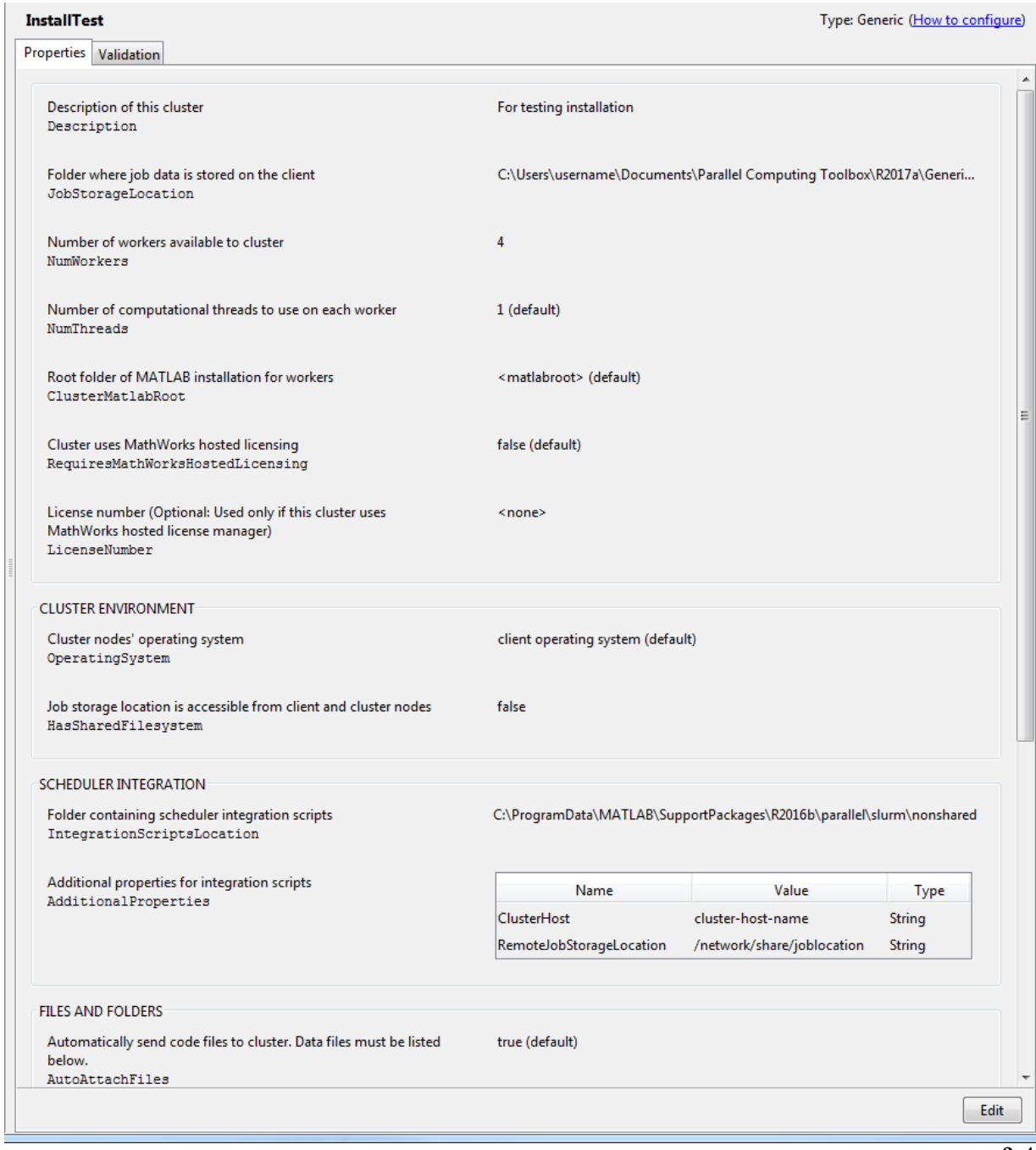
j Set the **IntegrationScriptsLocation** to the location of the nonshared subfolder within the LSF installation folder. As part of using nonshared submission mode, you must set the properties in steps `k`. and `l`.

k In the **AdditionalProperties** table, select **Add** and specify a new property with name `ClusterHost`, value `cluster-host-name`, and type `String`.

l In the **AdditionalProperties** table, select **Add** and 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 should look as follows.

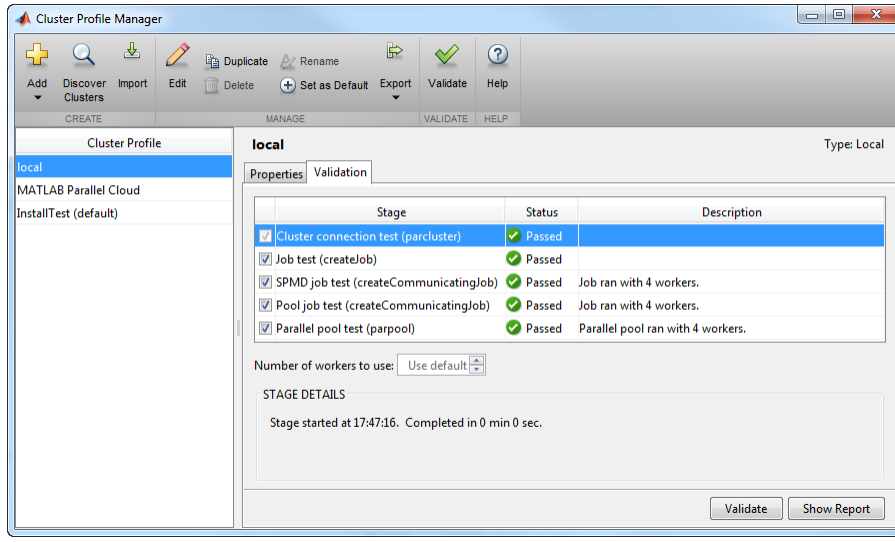

InstallTest
Type: Generic (How to configure)
Properties
Validation
Description of this cluster
Description
For testing installation
Folder where job data is stored on the client
JobStorageLocation
C:\Users\username\Documents\Parallel Computing Toolbox\R2017a\Generi...
Number of workers available to cluster
NumWorkers
4
Number of computational threads to use on each worker
NumThreads
1 (default)
Root folder of MATLAB installation for workers
ClusterMatlabRoot
<matlabroot> (default)
Cluster uses MathWorks hosted licensing
RequiresMathWorksHostedLicensing
false (default)
License number (Optional: Used only if this cluster uses MathWorks hosted license manager)
LicenseNumber
<none>
CLUSTER ENVIRONMENT
Cluster nodes' operating system
OperatingSystem
client operating system (default)
Job storage location is accessible from client and cluster nodes
HasSharedFilesystem
false
SCHEDULER INTEGRATION
Folder containing scheduler integration scripts
IntegrationScriptsLocation
C:\ProgramData\MATLAB\SupportPackages\R2016b\parallel\slurm\nonshared
Additional properties for integration scripts
AdditionalProperties
Name
Value
Type
ClusterHost
cluster-host-name
String
RemoteJobStorageLocation
/network/share/joblocation
String
FILES AND FOLDERS
Automatically send code files to cluster. Data files must be listed below.
AutoAttachFiles
true (default)
Edit

Step 2: Validate Cluster Profile

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.



Note If your validation fails any stage, 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**.

Distribute a Generic Cluster Profile and Integration Scripts

Read this page if you wish to distribute your Generic cluster profile and integration scripts for others to use. You can create a Generic cluster profile and integration scripts, if you do not have them already, by

1. Installing the appropriate support package for your third-party scheduler (see "Support Scripts" on page 3-41).
2. Using 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) in order to submit jobs to the cluster. As the person 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 Location for IntegrationScriptsLocation Folder" on page 3-52.
- If you prefer to give other users a copy of your integration scripts, follow the steps in "Give Users a Copy of the IntegrationScriptsLocation Folder" on page 3-53.

The first option, putting the integration scripts in a shared location, simplifies subsequent steps and allows any changes you make to the integration scripts to take immediate effect for all users.

Shared Location for IntegrationScriptsLocation Folder

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

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

- You must carry out these steps:

  1. Export your profile to a `.settings` file. Navigate to **Home** > **Parallel** > **Manage Cluster Profiles** to open the Cluster Profile Manager. Select your profile in the list and click **Export**. Choose a name for your `.settings` file and click **Save**.
  2. Send a copy of the `.settings` file, which contains your profile, to other users.

- Ask other users to carry out these steps:

  1. Save the `.settings` file to a location of their choice.
  2. Import the profile into MATLAB using the `.settings` file. Navigate to **Home** > **Parallel** > **Manage Cluster Profiles** to open the Cluster Profile Manager and click **Import**. Select the `.settings` file and click **Open**. A copy of your profile appears in their profile list.
  3. Check that the profile works by selecting the profile in the Cluster Profile Manager and clicking **Validate**.

Give Users a Copy of the IntegrationScriptsLocation Folder

If you cannot put your integration scripts in a shared location or prefer to give users their own copy of your integration scripts, follow these steps.

Note: After distributing your integration scripts, other users will not see any changes you make to your copy of the integration scripts.

- You must carry out these steps:

  1. Export your profile to a `.settings` file. Navigate to **Home** > **Parallel** > **Manage Cluster Profiles** to open the Cluster Profile Manager. Select your profile in the list and click **Export**. Choose a name for your `.settings` file and click **Save**.
  2. Send other users a copy of

     - The `.settings` file containing the exported profile.
     - The **IntegrationScriptsLocation** folder and all files therein.

- Ask other users to carry out these steps:

1. Save all files to a location of their choice.
2. Import the profile into MATLAB using the `.settings` file. Navigate to **Home** > **Parallel** > **Manage Cluster Profiles** to open the Cluster Profile Manager and click **Import**. Select the `.settings` file and click **Open**. A copy of your profile appears in their profile list.
3. Update the **IntegrationScriptsLocation** of the imported profile. 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.
4. Check that the profile works by selecting the profile in the Cluster Profile Manager and clicking **Validate**.

Note: If you are distributing a Generic cluster profile with the **HasSharedFileSystem** property set to true (shared or remote submission modes, see "Interfacing with Generic Schedulers" on page 3-40), the cluster machines must have read and write access to the location set in the **JobStorageLocation** property of the profile. Remind those receiving the profile that they must either

1. Set the **JobStorageLocation** in their profile to a shared location, preferably one that is unique to their username and MATLAB version;

2. Only create/submit jobs to the cluster when the current working folder is a shared location.

See Also

"Support Scripts" on page 3-41 | "Interfacing with Generic Schedulers" on page 3-40

Configure a Hadoop Cluster

This topic describes the requirements to allow jobs to run on an existing Hadoop® cluster.

The requirements are:

1. MATLAB Distributed Computing Server must be installed or available on the cluster nodes. See "Install Products and Choose Cluster Configuration" on page 3-2.
2. If the cluster is running in Kerberos authentication that requires the Java Cryptography Extension, you must download and install the Oracle version of this extension to each MATLAB Distributed Computing Server installation. You must also perform this step for the MATLAB client installation. To install the extension, place the Java Cryptography Extension jar files into the folder `${MATLABROOT}/sys/jre/${ARCH}/jre/lib/security`.
3. You must have a Hadoop installation on the MATLAB client machine, that can submit normal (non-MATLAB) jobs to the cluster.
4. The cluster must identify its user home directory as a valid location that the nodes can access. You must choose a local filesystem path and typically use a local folder such as `/tmp/hduserhome` or `/home/${USER}`. Set `yarn.nodemanager.user-home-dir` for Hadoop version 2.X.
5. There is one Hadoop property that must not be "final." (If properties are "final", they are locked to a fixed predefined value, and jobs cannot alter them.)

   The software needs to append a value to this property so that task processes are able to correctly run MATLAB. This property is passed as part of the job metadata given to Hadoop during job submission.

   This property is `mapred.child.env`, which controls environment variables for the job's task processes.
6. You must provide necessary information to the `parallel.cluster.Hadoop` object in the MATLAB client session. For example, see "Run mapreduce on a Hadoop Cluster" (Parallel Computing Toolbox) and "Use Tall Arrays on a Spark Enabled Hadoop Cluster" (Parallel Computing Toolbox).

1. Sender exports the profile.

   Export the sender profile to a `.settings` file. Navigate to **Home** > **Parallel** > **Manage Cluster Profiles** to open the Cluster Profile Manager. Select your profile in the list and click **Export**. Choose a name for your `.settings` file and click **Save**.

2 Sender distributes the exported profile.

   Send other users a copy of the `.settings` file containing the exported profile. The receiving users must save the file to a location of their choice.

3 Other users import the profile.

   Import the cluster admin profile into MATLAB using the `.settings` file. Navigate to **Home** > **Parallel** > **Manage Cluster Profiles** to open the Cluster Profile Manager and click **Import**. Select the `.settings` file and click **Open**. A copy of the sender profile appears in the profile list.

4 Other users validate their imported profile.

   Check that the profile works by selecting the profile in the Cluster Profile Manager and clicking **Validate**.

1 Sender exports the profile.

   Export the sender profile to a `.settings` file. Navigate to **Home** > **Parallel** > **Manage Cluster Profiles** to open the Cluster Profile Manager. Select your profile in the list and click **Export**. Choose a name for your `.settings` file and click **Save**.

2 Sender distributes the exported profile and integration scripts.

   Send other users a copy of

   - The `.settings` file containing the exported profile.
   - The **IntegrationScriptsLocation** folder and all the files contained within.

   The receiving users must save all files to a location of their choice.

3 Others users import the profile.

   Import the cluster admin profile into MATLAB using the `.settings` file. Navigate to **Home** > **Parallel** > **Manage Cluster Profiles** to open the Cluster Profile Manager and click **Import**. Select the `.settings` file and click **Open**. The cluster admin profile appears in your profile list.

4 Other users update the **IntegrationScriptsLocation** property of the profile.

   Select the sender profile in the Cluster Profile Manager and click **Edit**.
   Scroll down to the Scheduler Integration section of the profile and set the **IntegrationScriptsLocation** property to the folder containing your copy of the integration scripts.

5 Other users validate their imported profile.

Check that the profile works by selecting the profile in the Cluster Profile Manager and clicking **Validate**.

Hadoop Version Support

- MATLAB MapReduce is supported on Hadoop 1.x and Hadoop 2.x clusters. Note that support for Hadoop 1.x clusters will be removed in a future release, see the table.
- MATLAB Tall Array is supported on Spark® enabled Hadoop 2.x clusters.

Functionality	Result	Use Instead	Compatibility Considerations
Support for running MATLAB MapReduce on Hadoop 1.x clusters will be removed in a future release.	Warns	Use clusters that have Hadoop 2.x or higher 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

- “Install Products and Choose Cluster Configuration” 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)

4

Admin Center

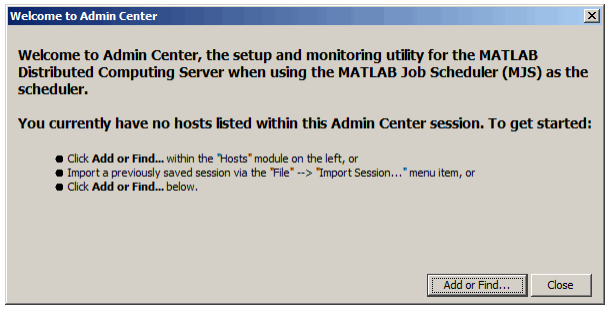
Start Admin Center

Admin Center is a graphical user interface with which you can control and monitor the MATLAB Distributed Computing Server processes of a MATLAB job scheduler (MJS) 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...

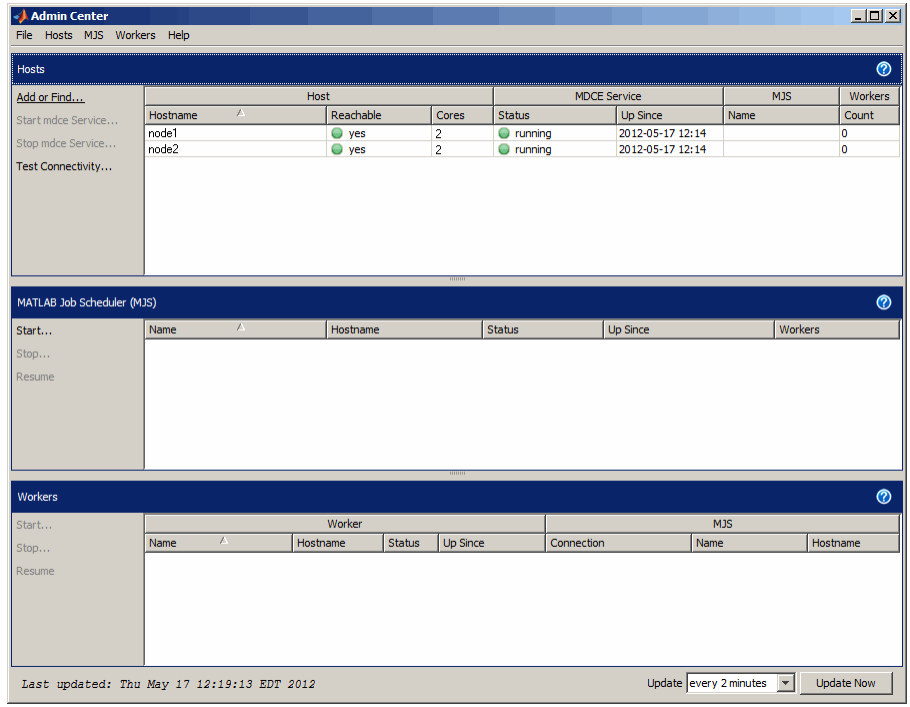
Add Hosts

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 (MJS), Admin Center automatically finds and lists all the hosts running workers registered with that MJS. Similarly, if you specify a host that is running a worker, Admin Center finds and lists the host running that worker’s MJS, and then also all hosts running other workers under that MJS.



Start mdce Service

A host must be running the mdce service if an MJS 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 Distributed Computing Server on your cluster, as described in the installation instructions available at “Installation”.

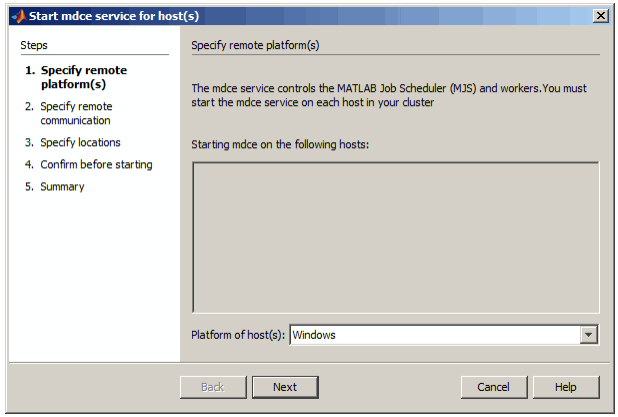
If you want to add or remove hosts to your cluster, Admin Center allows you to start and stop the mdce service on those hosts. To start the mdce service on a group of hosts with the same platform, select all those hosts in the Hosts module, and click **Start mdce Service** in the left column of the panel.

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

A dialog box leads you through the procedure of starting the mdce 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 mdce on multiple hosts at the same time, but they all must be the same platform. If you have a mixed platform cluster, run the mdce 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 `mdce_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:

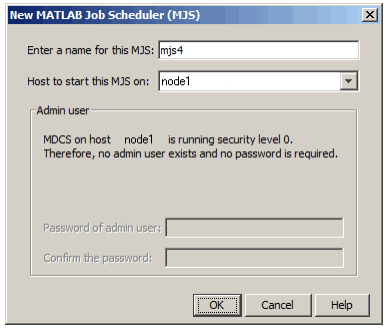


At each step, you can click **Help** to read detailed information about that step.

Start an MJS

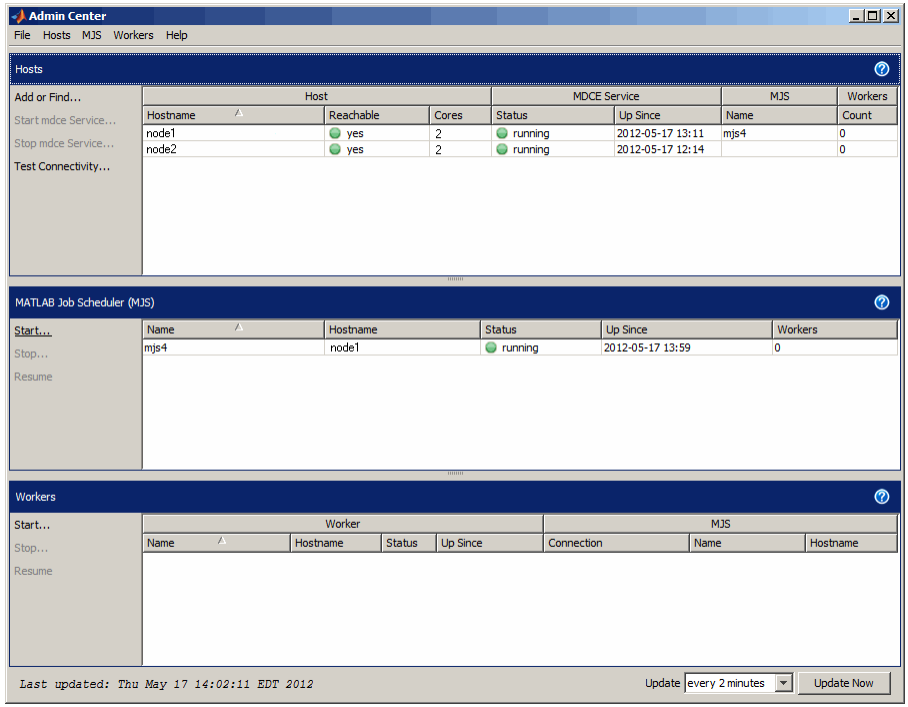
To start an MJS, click **Start** in the MJS module.

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



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

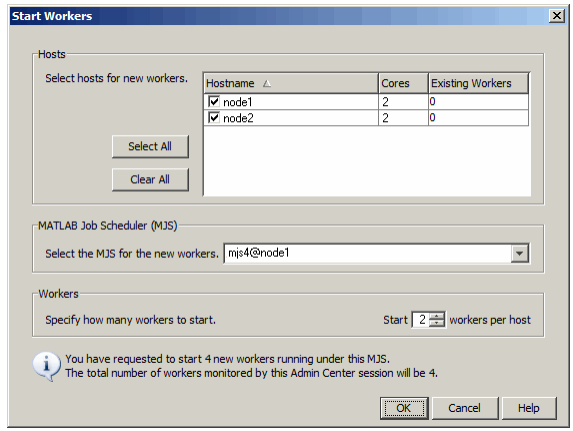
With an MJS running on your cluster, Admin Center might look like the following figure, with the MJS listed in the MJS module, as well as being listed by name in the Hosts module in the line for the host on which it is running.



Start Workers

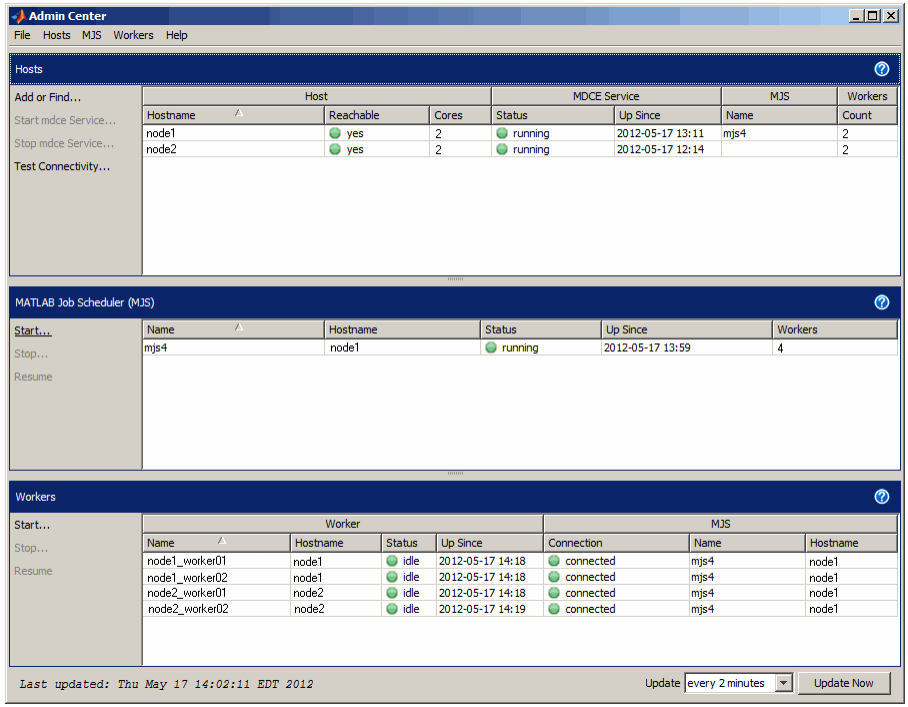
To start MATLAB workers, click **Start** in the Workers module.

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 MJS 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 MJS 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 MJS is listed in the MJS module, and the number of workers for each MJS is listed in the Hosts module.



To get more information on any host, MJS, 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**, **MJS**, and **Workers** drop-down menus.

Stop, Destroy, Resume, Restart Processes

You can **Stop** or **Destroy** the mdce service, MJSs, 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 mdce Service** to have mdce continue with existing data. Use **Resume** to have an MJS 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 Distributed Computing Server processes (mdce service, MJS, 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 MJS name.

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

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

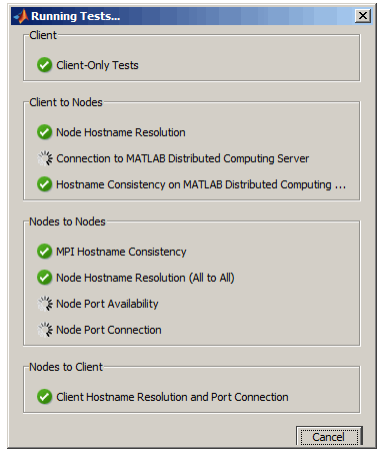
Admin Center lets you test communications between your MJS node, worker nodes, and the node where Admin Center is running.

The tests are divided into four categories:

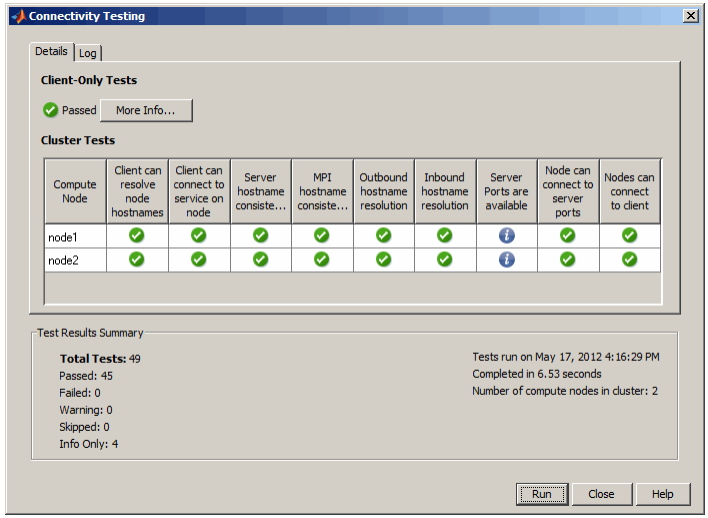
- **Client** — Verifies that the node running Admin Center is properly configured so that further cluster testing can proceed.
- **Client to Nodes** — Verifies that the node running Admin Center can identify and communicate with the other nodes in the cluster.
- **Nodes to Nodes** — Verifies that the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



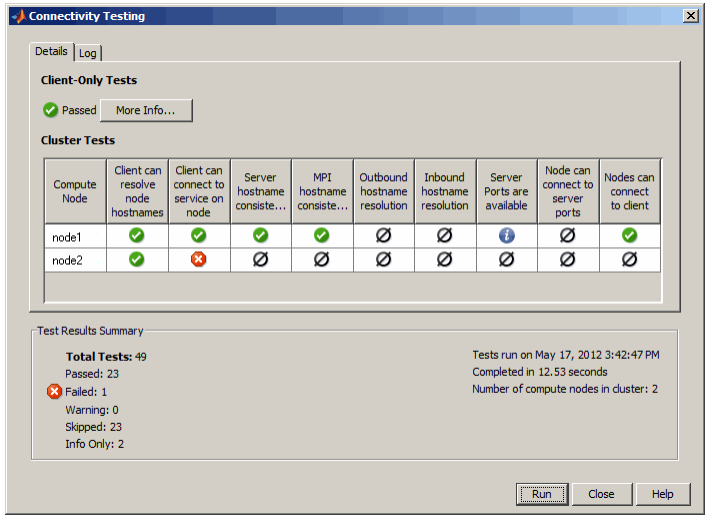
When the tests are complete, the Running Tests dialog box automatically closes, and Admin Center displays the test results in the Connectivity Testing dialog box.



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.



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 `.mdcs` 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 MJS name, MJS host, and number of workers. For more information about creating and using profiles, see “Discover Clusters and Use Cluster Profiles” (Parallel Computing Toolbox) in the Parallel Computing Toolbox documentation.

Control Scripts — Alphabetical List

admincenter

Start Admin Center GUI

Syntax

```
admincenter
```

Description

`admincenter` opens the MATLAB Distributed Computing Server Admin Center. When setting up or using a MATLAB job scheduler (MJS) 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 4-10

See Also

`mdce` | `nodestatus` | `remotemdce`

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 `mdce_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 `mdce_def` files set the parameter `SHARED_SECRET_FILE` to `/share/secret` before starting the mdce service on each.

See Also

`mdce`

mdce

Install, start, stop, or uninstall mdce service

Syntax

```
mdce install
mdce uninstall
mdce start
mdce stop
mdce console
mdce restart
mdce ... -mdcedef <mdce_defaults_file>
mdce ... -clean
mdce status
mdce -version
mdce -usemhlm
```

Description

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

The `mdce` 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.

`mdce install` installs the mdce 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.

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

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

`mdce stop` stops running the mdce 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 mdce service is started again.

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

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

`mdce ... -mdcedef <mdce_defaults_file>` uses the specified alternative mdce defaults file instead of the one found in *`matlabroot`*`/toolbox/distcomp/bin`. This file is provided for Linux (`mdce_def.sh`) and Windows (`mdce_def.bat`). Also see the Backwards Compatibility Note in "Install Products and Choose Cluster Configuration" on page 3-2.

`mdce ... -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.

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

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

`mdce -usemhlm` ensures that workers use MathWorks Hosted License Manager. Unless you specify –`usemhlm`, mdce uses FlexLM-based licensing.

See Also

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

nodestatus

Status of mdce processes running on node

Syntax

```
nodestatus
nodestatus -flags
```

Description

`nodestatus` displays the status of the mdce service and the processes which it maintains. The mdce 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
`-remotehost <hostname>`	Displays the status of the mdce service and the processes it maintains on the specified host. The default value is the local host.
`-infolevel <level>`	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.
`-baseport <port_number>`	Specifies the base port that the mdce service on the remote host is using. You need to specify this only if the value of `BASE_PORT` in the local `mdce_def` file does not match the base port being used by the mdce service on the remote host.

Flag	Operation
`-v`	Verbose mode displays the progress of the command execution.
`-json`	View the output in JavaScript Object Notation (JSON) format. Output in `json` format is easy to parse.

Examples

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

```
nodestatus
```

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

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

See Also

`mdce` | `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 rsh or 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
`-remoteplatform { unix | windows }`	Specify the platform of the remote hosts. This option is required only if different from the local platform.
`-quiet`	Prevent `remotecopy` from prompting for missing information. The command fails if all required information is not specified.
`-help`	Print the help information for this command.

`-protocol <type>`	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.
	
	The supported protocol types are `scp`, `sftp` and `rcp`.
	
	To get more information about one particular protocol type, enter
	
	`remotecopy -protocol <type> -help`
	
	For example:
	
	`remotecopy -protocol sftp -help`

<protocol options>	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 `mdce_def.sh` to two other machines. (Enter this command on a single line.)

```
remotecopy -local mdce_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\mdce\log
  -remotehost winHost1,winHost2
```

See Also

`remotemdce`

remotemdce

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

Syntax

```
remotemdce <mdce options> <flags> <protocol options>
```

Description

`remotemdce <mdce options> <flags> <protocol options>` allows you to execute the mdce service on one or more remote hosts.

For a description of the mdce service, see the `mdce` 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
<mdce options>	Options and arguments of the mdce command, such as `start`, `stop`, etc. See the `mdce` 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 mdce 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 mdce from prompting the user for missing information. The command fails if all required information is not specified.
`-help`	Print help information.

Flags and Options	Operation

`-protocol <type>`	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.
	
	The supported protocol types are `ssh`, `rsh`, and `winsc`.
	
	To get more information about one particular protocol type, enter
	
	`remotemdce -protocol <type> -help`
	
	For example:
	
	`remotemdce -protocol winsc -help`
	
	Using the `winsc` protocol requires that you log in as a user with admin privileges on the remote host.

<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\mdce_def.bat`, you should identify it as `C:/temp/mdce_def.bat`.

Examples

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

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

Start mdce 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:

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

See Also

mdce | remotecopy

pausejobmanager

Pause job manager process

Syntax

```
pausejobmanager
pausejobmanager -flags
```

Description

`pausejobmanager` pauses a job manager that is running under the mdce 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 `mdce_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 mdce service on the remote host is using. You need to specify this only if the value of `BASE_PORT` in the local `mdce_def` file does not match the base port being used by the mdce 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

`mdce` | `nodestatus` | `startjobmanager` | `stopjobmanager` | `resumejobmanager`

resumejobmanager

Resume job manager process

Syntax

```
resumejobmanager
resumejobmanager -flags
```

Description

`resumejobmanager` resumes a job manager that is running under the mdce 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 `mdce_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 mdce service on the remote host is using. You need to specify this only if the value of `BASE_PORT` in the local `mdce_def` file does not match the base port being used by the mdce 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

`mdce` | `nodestatus` | `startjobmanager` | `stopjobmanager` | `pausejobmanager`

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 mdce 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 mdce 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
`-name <job_manager_name>`	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 `DEFAULT_JOB_MANAGER_NAME` parameter in the `mdce_def` file.
`-remotehost <hostname>`	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 mdce service on the remote host is using. You need to specify this only if the value of `BASE_PORT` in the local `mdce_def` file does not match the base port being used by the mdce 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

`mdce` | `nodestatus` | `pausejobmanager` | `resumejobmanager` | `startworker` | `stopjobmanager` | `stopworker`

startworker

Start MATLAB worker session

Syntax

```
startworker
startworker -flags
```

Description

`startworker` starts a MATLAB worker process under the mdce service, which maintains it after that. The worker registers with the specified job manager, from which it will get tasks for evaluation. The mdce 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 `DEFAULT_WORKER_NAME` parameter in the `mdce_def` 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.
`-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 `DEFAULT_JOB_MANAGER_NAME` parameter in the `mdce_def` file.

Flag	Operation
`-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 `JOB_MANAGER_HOST` in the `mdce_def` 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 mdce service on the remote host is using. You only need to specify this if the value of `BASE_PORT` in the local `mdce_def` file does not match the base port being used by the mdce 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

mdce | nodestatus | startjobmanager | stopjobmanager | stopworker

stopjobmanager

Stop job manager process

Syntax

```
stopjobmanager
stopjobmanager -flags
```

Description

`stopjobmanager` stops a job manager that is running under the mdce 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 `mdce_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 mdce service on the remote host is using. You need to specify this only if the value of `BASE_PORT` in the local `mdce_def` file does not match the base port being used by the mdce 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

`mdce` | `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 mdce 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 `mdce_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.
`-baseport <port_number>`	Specifies the base port that the mdce service on the remote host is using. You need to specify this only if the value of `BASE_PORT` in the local `mdce_def` file does

Flag	Operation
	not match the base port being used by the mdce 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

`mdce` | `nodestatus` | `startjobmanager` | `startworker` | `stopjobmanager`

Glossary

CHECKPOINTBASE The name of the parameter in the `mdce_def` 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 `parfor`-loops and `spmd` 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 `spmd` 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 Distributed Computing Server software. For a list of products that are not eligible for use with Parallel Computing Toolbox software, see `http://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 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 communication 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 `mdce_def` file that defines the directory where logs are stored.

MATLAB client — See client.

MATLAB job scheduler (MJS) — The MathWorks process that queues jobs and assigns tasks to workers. Formerly known as a job manager.

MATLAB worker — See worker.

mdce — 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.

mdce_def file — The file that defines all the defaults for the mdce 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.