MATLAB® Distributed Computing Engine

For Use with MATLAB®

- Computation
- Visualization
- Programming

System Administrator's Guide

Version 2

How to Contact The MathWorks:

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

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

508-647-7000 Phone

508-647-7001 Fax

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

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

MATLAB Distributed Computing Engine 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)

Contents

1 Introduction

2 Network Administration

1

Introduction

This chapter provides an introduction to the concepts and terms of the Distributed Computing Toolbox and the MATLAB® Distributed Computing Engine.

What Are the Distributed Computing Products?

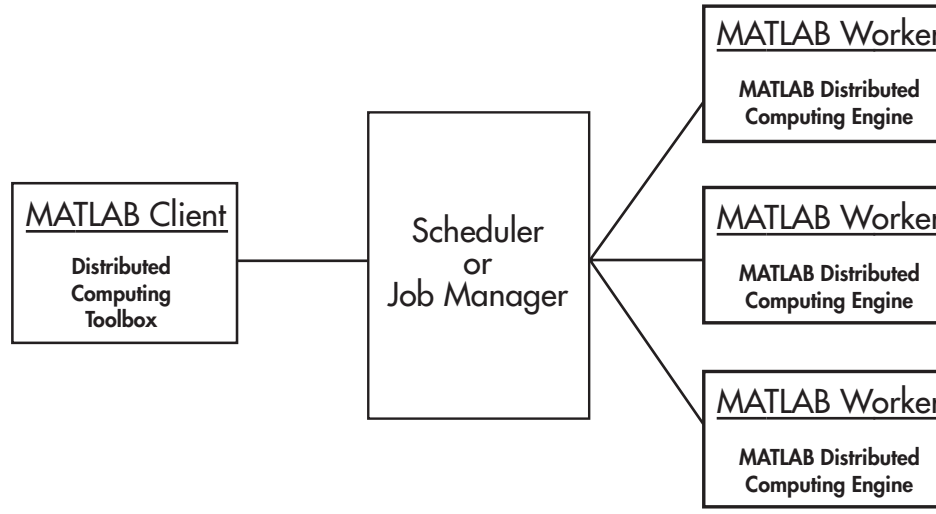
The Distributed Computing Toolbox and the MATLAB Distributed Computing Engine enable you to coordinate and execute independent MATLAB operations simultaneously on a cluster of computers, speeding up execution of large MATLAB jobs.

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 the Distributed Computing Toolbox to perform the definition of jobs and tasks. The MATLAB Distributed Computing Engine is the product that performs the execution of your job by evaluating each of its tasks and returning the result to your client session.

The *job manager* is the part of the engine that coordinates the execution of jobs and the evaluation of their tasks. The job manager distributes the tasks for evaluation to the engine's individual MATLAB sessions called *workers*. Use of the MathWorks job manager is optional; the distribution of tasks to workers can also be performed by a third-party scheduler, such as LSF.

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



Basic Distributed Computing Configuration

Determining Product Installation and Versions

To determine if the Distributed Computing Toolbox 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 application to determine what version of the MATLAB Distributed Computing Engine is installed on a worker machine. Note that the toolbox and engine must be the same version.

Toolbox and Engine Components

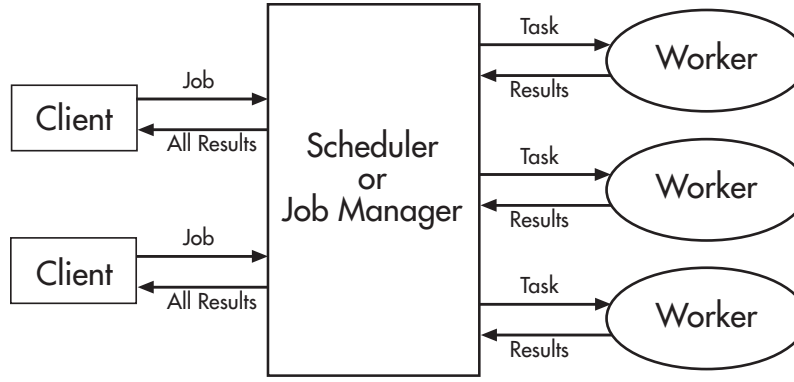
Job Managers, Workers, and Clients

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

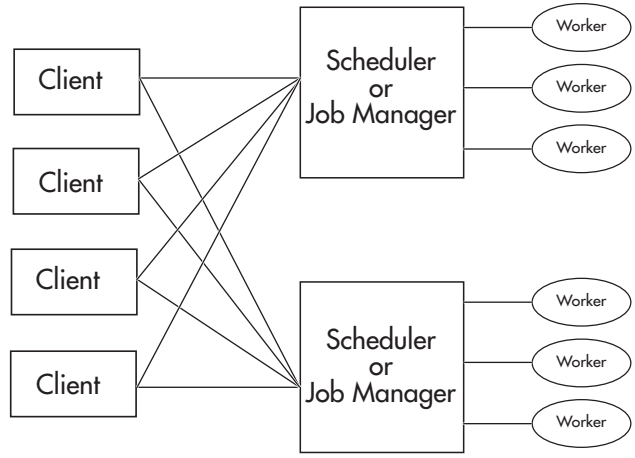
A MATLAB Distributed Computing Engine setup usually includes many workers that can all execute tasks simultaneously, speeding up execution of large MATLAB jobs. It is generally not important which worker executes a specific task. Each worker evaluates tasks one at a time, returning the results to the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



Interactions of Distributed Computing Sessions

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



Configuration with Multiple Clients and Job Managers

Third-Party Schedulers

As an alternative to using the MathWorks job manager, you can use a third-party scheduler. This could be an LSF scheduler or a generic scheduler.

Choosing Between a Scheduler and Job Manager

You should consider the following when deciding to use a scheduler or the MathWorks job manager for distributing your tasks:

- Does your cluster already have a scheduler?

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

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

  The MathWorks job manager is designed specifically for MathWorks distributed 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 MathWorks job manager can handle all file and data sharing necessary for your distributed computing applications. This might be helpful in configurations where shared access is limited.
- Are you interested in batch or interactive processing?

  When you use a job manager, worker processes usually remain running at all times, dedicated to their job manager. 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 job manager.
- 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

The Distributed Computing Toolbox and MATLAB Distributed Computing Engine are supported on Windows, UNIX, and Macintosh platforms. Mixed platforms are supported, so that the clients, job managers, and workers do not have to be on the same platform. The cluster can also be comprised of both 32-bit and 64-bit machines, so long as your data does not exceed the limitations posed by the 32-bit systems.

To run parallel applications that use inter-worker communication, all the cluster nodes running the application must have matching word sizes and processor endianness. Heterogeneous cluster configurations running these applications can comprise the following:

- Solaris with Macintosh
- 32-bit Windows with 32-bit Linux

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

MATLAB Distributed Computing Engine Service

If you are using the MathWorks job manager, every machine that hosts a worker or job manager session must also run the MATLAB Distributed Computing Engine (mdce) service.

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

Using the Distributed Computing Toolbox

A typical Distributed Computing Toolbox client session includes the following steps.

1. Find a Job Manager (or scheduler) — Your network may have one or more job managers available (but usually only one scheduler). The function you use to find a job manager or scheduler creates an object in your current MATLAB session to represent the job manager 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 job manager (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 job manager or scheduler. The job manager 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.

2

Network Administration

This chapter provides information useful for network administration of the Distributed Computing Toolbox and the MATLAB Distributed Computing Engine.

Preparing for Distributed Computing (p. 2-2)	Examines network requirements and limitations for running the Distributed Computing Toolbox and the MATLAB Distributed Computing Engine
Installing and Configuring (p. 2-5)	Where to find installation and configuration instructions
Shutting Down a Job Manager Configuration (p. 2-6)	Terminating distributed computing processes on your cluster
Customizing Engine Services (p. 2-10)	Overriding or modifying default parameters for scripts
Accessing Service Record Files (p. 2-14)	Accessing service logs and specifying their locations
Configuring mpiexec on Your Cluster (p. 2-16)	Setting up smpd and mpiexec on your cluster
Troubleshooting (p. 2-19)	Diagnosing and solving problems with your cluster

Preparing for Distributed Computing

Before You Start

Before attempting an installation of the Distributed Computing Toolbox and MATLAB Distributed Computing Engine, read Chapter 1, “Introduction” to familiarize yourself with the concepts and vocabulary of the products.

Planning Your Network Layout

Generally, there is not much difficulty in deciding 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 client session of MATLAB 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 distributed computing applications.

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

Session	Product	Processes
Client	Distributed Computing Toolbox	MATLAB with toolbox
Worker	MATLAB Distributed Computing Engine	mdce service; worker
Job manager	MATLAB Distributed Computing Engine	mdce service; job manager

The MATLAB Distributed Computing Engine (mdce) service or daemon is included in the engine software. It is separate from the worker and job manager processes, and it must be running on all machines that run worker or job manager sessions.

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

Network Requirements

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

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

Security Considerations

The distributed computing products do not provide any security measures. Therefore, you should 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 and as `LocalSystem` on Windows. 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 distributed 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 parallel 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 distributed computing. See "Defining the Script Defaults" on page 2-12.
- If your network supports multicast, the distributed computing processes accommodate multicast. However, because multicast is disabled on many networks for security reasons, you might require unicast communication between distributed computing processes. Most examples of MDCE scripts and Distributed Computing Toolbox functions in the documentation show unicast usage.
- If your organization is a member of the Internet Multicast Backbone (MBone), you need to ensure that your distributed computing cluster is isolated from MBone access if you are using multicast for distributed computing. This is generally the default condition. If you have any questions about MBone membership, contact your network administrator.

Installing and Configuring

To find the most up-to-date instructions for installing and configuring the current or past versions of the distributed computing products, visit the MathWorks Web site at

```
http://www.mathworks.com/support/product/DM/installation/ver_current/
```

Shutting Down a Job Manager Configuration

If you are done using the job manager and its workers, you might want to shut down the engine 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.

UNIX and Macintosh

Stopping the Job Manager and Workers

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

   ```
   cd $MATLAB/toolbox/distcomp/bin
   ```

   (Enter the following command on a single line.)

   ```
   stopjobmanager.sh -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.sh -help
   ```

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

   ```
   cd $MATLAB/toolbox/distcomp/bin
   stopworker.sh -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.sh -name worker1 -remotehost <worker hostname>
   stopworker.sh -name worker2 -remotehost <worker hostname>
   ```

   For a list of all options to the script, type

   ```
   stopworker.sh -help
   ```

Stopping and Uninstalling 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 Engine from a machine, you might want to uninstall the mdce daemon also, as you will not need it any longer.

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
   ```

Stopping the Daemon Manually. If you used the alternative manual startup of the mdce daemon, you can use the following commands to stop it manually.

```
cd $MATLAB/toolbox/distcomp/bin
mdce stop
```

Windows

Stopping the Job Manager and Workers

1 To shut down the job manager, enter the commands

```
cd $MATLAB\toolbox\distcomp\bin\win32
```

(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 $MATLAB\toolbox\distcomp\bin\win32
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
```

Stopping and Uninstalling 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 $MATLAB\toolbox\distcomp\bin\win32
mdce stop
```

If you plan to uninstall the MATLAB Distributed Computing Engine from a machine, you might want to uninstall the mdce service also, as you will not need it any longer.

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 $MATLAB\toolbox\distcomp\bin\win32
mdce uninstall
```

Customizing Engine Services

The scripts of the MATLAB Distributed Computing Engine run using several default parameters. You can customize the scripts, as described in the following sections:

- "Overriding the Script Defaults" on page 2-10
- "Defining the Script Defaults" on page 2-12

Overriding the Script Defaults

Specifying the Defaults File

The default parameters used by the mdce service, job managers, and workers are defined in the file `$MATLAB/toolbox/distcomp/bin/mdce_def.sh` (UNIX) or `$MATLAB\toolbox\distcomp\bin\win32\mdce_def.bat` (Windows). Before 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 it, and specify that this copy be used for the default parameters.

On UNIX or Macintosh,

```
startjobmanager.sh -mdcedef my_mdce_def.sh
startworker.sh -mdcedef my_worker_def.sh
```

On Windows,

```
startjobmanager -mdcedef my_mdce_def.bat
startworker -mdcedef my_worker_def.bat
```

For more information, see "Defining the Script Defaults" on page 2-12.

Starting 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 will not be destroyed or lost if the job manager machine crashes or if the job manager is inadvertently shut down. If you want to start up a job manager or worker from a clean state, with all history deleted, use the `-clean` flag on the `start` command.

On UNIX or Macintosh,

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

On Windows,

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

Defining the Script Defaults

The scripts for the engine 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 can be accessed 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 engine service scripts are defined in the file `$MATLAB\toolbox\distcomp\bin\win32\mdce_def.bat` (Windows), or `$MATLAB/toolbox/distcomp/bin/mdce_def.sh` (UNIX/Macintosh). To set the default parameters, edit this file before starting the mdce service.

Alternatively, you can make a copy of this file, modify it, and specify that this new copy be used for the service default parameters using the `-mdcedef` flag with the `mdce` script.

Note that 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 call.

For example, on Windows systems, you use the parameter file `my_mdce_def.bat` by typing

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

On UNIX or Macintosh systems, you use the parameter file `my_mdce_def.sh` by typing

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

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

Setting 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 system, set the value before starting the service; on a Windows system, set it before installing the service.
`MDCEPASS`	On a Windows system, set this parameter to specify the password for the user identified in the `MDCEUSER` parameter; otherwise, the system will prompt you for the password when the service is installed.

On UNIX 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`).

On Windows systems, when executing the `mdce start` script, the user defined by `MDCEUSER` must be listed among those who can log on as a service. To see the list of valid users, click the Windows **Start** > **Settings** > **Control Panel**. Double-click `Administrative Tools`, then `Local Security Policy`. In the tree, select `User Rights Assignment`, then in the right panel, double-click `Log on as a service`. This dialog must list the user defined for `MDCEUSER` in your `mdce_def.bat` file. If not, you can add the user to this dialog according to the instructions in the `mdce_def.bat` file, or when running `mdce start`, you can use another `mdce_def.bat` file that specifies a listed user.

Accessing Service Record Files

The services of the MATLAB Distributed Computing Engine generate various record files in the normal course of their operations. The mdce service, job manager, and worker sessions all generate such files. The types of information stored by the services are described in the following sections:

- "Locating Log Files" on page 2-14
- "Locating Checkpoint Directories" on page 2-15

Locating 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.

Platform	File Location
Windows	On Windows systems, 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`, then 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.
UNIX and Macintosh	On UNIX and Macintosh systems, 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.

Locating Checkpoint Directories

Checkpoint directories contain information related to persistence data, which the engine 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 will continue the old session, using all the same data.

A primary feature offered by the checkpoint directories is in crash recovery. This allows engine services to automatically resume their sessions after a system goes down and comes back up, without losing any data. (If a job manager crashes, its workers can take up to 2 minutes to reregister with it.)

Platform	File Location
Windows	On Windows systems, the default location of the checkpoint directories is `<TEMP>\MDCE\Checkpoint`, where `<TEMP>` is the value of the system `TEMP` variable. For example, if `TEMP` is set to `C:\TEMP`, then the checkpoint directories are placed in `C:\TEMP\MDCE\Checkpoint`. You can set alternative locations for the checkpoint directories by modifying the `CHECKPOINTBASE` setting in the `mdce_def.bat` file before starting the mdce service.
UNIX and Macintosh	On UNIX and Macintosh systems, the checkpoint directories are placed by default in `/var/lib/mdce/`. You can set alternative locations for the checkpoint directories by modifying the `CHECKPOINTBASE` setting in the `mdce_def.sh` file before starting the mdce service.

Configuring mpiexec on Your Cluster

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

For further information about mpiexec and smpd, see the MPICH2 home page and installer's guide at

```
http://www-unix.mcs.anl.gov/mpi/mpich2/
http://www-unix.mcs.anl.gov/mpi/mpich2/downloads/mpich2-doc-install.pdf
```

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

Starting the smpd Service

To use an mpiexec scheduler, the smpd service must be running on all nodes that will be used to run MATLAB workers.

Windows Clusters

For Windows cluster machines, the `smpd` executable is in `$MATLAB\bin\win32`.

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.

---

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

   ```
   $MATLAB\bin\win32\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. Repeat these steps on all Windows nodes in your cluster.

UNIX Clusters

For UNIX cluster machines, the `smpd` executable is found in `$MATLAB/bin/$arch`, where `$arch` is the architecture of the node.

1. Start smpd by typing in a shell

   ```
   cd $MATLAB/bin/$arch
   smpd -s -phrase MATLAB
   ```

   This command starts the service. You might want to configure this command to run at node boot time.

2. Repeat this step on all UNIX nodes in your cluster.

The passphrase authenticates jobs submitted to the scheduler. It corresponds to the passphrase specified by the `SubmitArguments` property of the scheduler object in the Distributed Computing Toolbox application. You can use any text you want for the passphrase; this example uses `MATLAB` to correspond to the examples in the Distributed Computing Toolbox User's Guide.

MATLAB workers on UNIX nodes run tasks as the user who started smpd. These workers must have permission to write to files in the directory specified by the scheduler's `DataLocation` property.

Configuring the Client Computer

Windows Clients with Windows Clusters

The `mpiexec` command uses credentials to authenticate users for running jobs. The client computer requires that you have credentials in the Windows registry for each user who will submit jobs. To set up your Windows client computer for submitting jobs to an smpd service,

1. Log in as the user you want to submit jobs as.
2. Enter your credentials in the registry by typing the following commands at a DOS command prompt.

   ```
   cd $MATLAB\bin\win32
   mpiexec -register
   ```

3. Provide your login name and password when requested. The information gets encrypted in the Windows registry.

You can later remove your credentials from the registry by running

```
mpiexec -remove
```

You must execute this command for each user whose credentials you want to remove. That is, it removes only those credentials of the currently logged in user.

UNIX Clients with Windows Clusters

UNIX clients do not have registry credentials to pass to the Windows cluster nodes, so password files provide identification for jobs. To send the password file from a UNIX client to a Windows cluster when running a job, you must include the following option as part of the scheduler object's `SubmitArguments` property in the application that defines the job.

```
-pwdfile <filename>
```

The password file is an unencrypted file containing a login name on the first line, and a password on the second line.

Troubleshooting

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. You can 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 the Distributed Computing Engine
  - 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 Distributed Computing Toolbox 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 will see the following information.

```
License Manager Error -15.
Cannot connect to license server
 The server (lmgrd) has not been started yet, or
 the wrong port@host or license file is being used, or the
 port or hostname in the license file has been changed.
Feature:       MATLAB_Distrib_Comp_Engine
Server name:   hostname
License path:  /matlab/etc/license.dat
FLEXlm error:  -15,570.  System Error: 111 "Connection refused"
For further information, refer to the FLEXlm End User Manual,
available at "www.macrovision.com".

MATLAB is unable to connect to the license server.
Make sure the SERVER line in your license file is correct.
Have your MATLAB administrator verify that the license manager is
running and validate network services.

For more information, see The MathWorks Support page at
http://www.mathworks.com/support and search for
"license manager error -15"
```

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

Verifying Multicast Communications

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. For an in-depth discussion of multicast, see the multicast FAQ at `http://www.multicasttech.com/multicast_faq.html`.

The main tools for investigating this type of packet are `tcpdump` or the equivalent on Windows (usually called `winpcap` and `ethereal`), and a Java class included with DCT/MDCE Version 2.0.

The 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. The class can be used both inside MATLAB and from a call to Java.

Inside MATLAB, the class would be used as follows.

```
m = com.mathworks.toolbox.distcomp.test.MulticastTester('239.1.1.1', 9999);
m.startSendingThread;
m.startListeningThread;
 0 : host1name : 0
 1 : host2name : 0
```

From a shell prompt, you would type (assuming that Java is on your path)

```
java -cp distcomp.jar com.mathworks.toolbox.distcomp.test.MulticastTester
0 : host1name : 0
1 : host2name : 0
```

Memory Errors on UNIX

If the number of threads created by the engine services on a UNIX machine exceeds the limitation set by the `maxproc` value, the services will fail and generate an out-of-memory error. You can check your `maxproc` value on UNIX with the `limit` command. (Different versions of UNIX might have different names for this property instead of `maxproc`, such as `descriptors` on Solaris.)

Running MDCE Processes from a Windows Network Installation

Because the worker is installed as a service running under `LocalSystem` by default, it does not have access to mapped network drives. You must use the full UNC path when installing the mdce service if you want to run from a network share. In the following sections, use a full UNC path for your `$MATLAB`, such as

```
\\MyDomain\Host52\Applications\MATLAB
```

Do **NOT** use a mapped drive for your `$MATLAB`, such as

```
P:\Applications\MATLAB
```

Many networks are configured not to allow `LocalSystem` (the default user for MDCE) access to UNC or mapped network shares. In this case, run MDCE under a different user with rights to log on as a service. See “Setting the User” on page 2-13.

3

Control Script Reference

This chapter describes the MATLAB Distributed Computing Engine control scripts that you use to start, stop, and configure the engine processes.

On Windows systems, these scripts are in

```
$MATLAB\toolbox\distcomp\bin\win32
```

On Windows computers, you execute the scripts by specifying the root word of the script name, such as `mdce`, `startjobmanager`, etc.

On UNIX systems, these scripts are in

```
$MATLAB/toolbox/distcomp/bin
```

On UNIX computers you must include the `.sh` extention in the call to the scripts, such as `mdce.sh`, `stopworker.sh`, etc.

Control Scripts — By Category

MDCE Control Scripts	Scripts for controlling mdce service
Job Manager Control Scripts	Scripts for controlling job manager
Worker Control Scripts	Scripts for controlling MATLAB workers

MDCE Control Scripts

`mdce`	Install, start, stop, or uninstall mdce service
`nodestatus`	Status of mdce service and its subordinate services

Job Manager Control Scripts

`startjobmanager`	Start job manager process
`stopjobmanager`	Stop job manager process

Worker Control Scripts

`startworker`	Start MATLAB worker
`stopworker`	Stop MATLAB worker

Control Scripts — Alphabetical List

This section contains detailed descriptions of the MATLAB Distributed Computing Engine control scripts. Each reference page contains some or all of the following information:

- The script name
- The script purpose
- The script syntax
- A description of each function syntax, including command flags
- Examples of usage
- References to related scripts

mdce

Purpose

Install, start, stop, or uninstall mdce service

Syntax

```
mdce install
mdce start
mdce stop
mdce uninstall
```

Description

```
mdce:           Start the base service associated with the MATLAB
                Distributed Computing Engine.
                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.

Usage: mdce [ install | uninstall | start | stop | console ]
            [ -mdcedef mdce_defaults_file ]

install         Install the mdce service in the Windows Service Control
                Manager.  This causes the service to automatically start
                when Windows boots up.  The service must be installed
                before it is started.

uninstall       Uninstall 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.

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

stop            Stop 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.

console         Start the mdce service as a process in the current DOS
                window rather than as a service running in the
                background.

-mdcedef        Specify an alternative mdce defaults file instead of
                the one found in MATLABROOT/toolbox/distcomp/bin/win32.
```

See also: nodestatus, startjobmanager, stopjobmanager, startworker, stopworker

nodestatus

Purpose

Status of MDCE processes running on node

Syntax

```
nodestatus
```

Description

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


Usage:  nodestatus [ -remotehost hostname ]
                   [ -infolevel level ]
                   [ -baseport port_number ]
                   [ -v ]

-remotehost     Display the status of the mdce service and the
                processes it maintains on the specified host.
                The default value is the local host.

-infolevel      Specify 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       Specify 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              Be verbose.  Display the progress of the command
                execution.

Examples:       1) Display basic information about the local host.

                nodestatus.sh

                2) Display detailed information about the status of the
                host node27.

                nodestatus.sh -remotehost node27 -infolevel 2


See also:       mdce, startjobmanager, stopjobmanager, startworker,
                stopworker
```

Purpose

Start job manager process

Syntax

```
startjobmanager
```

Description

```
startjobmanager:Start 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.

Usage:  startjobmanager [ -name job_manager_name ]
                        [ -remotehost hostname ]
                        [ -clean ]
                        [ -multicast ]
                        [ -baseport port_number ]
                        [ -v ]

-name           Specify 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     Specify the name of the host where you want to start
                the job manager and the job manager lookup process.
                If omitted, they are started on the local host.

-clean          Delete all checkpoint information stored on disk from
                previous instances of this job manager before starting.
                This will clean the job manager so that it will
                initialize with no jobs or tasks.

-multicast      Override the use of unicast to contact the job manager
                lookup process. It is recommended that you not use
                -multicast unless you are certain that multicast works
                on your network. This overrides the setting of
                JOB_MANAGER_HOST in the mdce_def file on the remote
                host, which would have the job manager use unicast.
                If this flag is omitted and JOB_MANAGER_HOST is empty,
                the job manager uses unicast to contact the job manager
                lookup process running on the same host.
```

```
-baseport       Specify 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              Be verbose.  Display the progress of the command
                execution.

Examples:       1) Start the job manager MyJobManager on the local
                host.

                startjobmanager.sh -name MyJobManager

                2) Start the job manager MyJobManager on the host
                JMHost.

                startjobmanager.sh -name MyJobManager -remotehost JMHost

See also:       mdce, nodestatus, stopjobmanager, startworker,
                stopworker
```

Purpose

Start MATLAB worker session

Syntax

```
startworker
```

Description

```
   startworker:    Start 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.

   Usage:  startworker [ -name worker_name ]
                       [ -jobmanager job_manager_name ]
                       [ -jobmanagerhost jmhost | -multicast ]
                       [ -remotehost hostname ]
                       [ -clean ]
                       [ -baseport port_number ]
                       [ -v ]

   -name           Specify the name of the MATLAB worker.  The default is
                   the value of the DEFAULT_WORKER_NAME parameter in the
                   mdce_def file.

   -jobmanager     Specify 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.

   -jobmanagerhost Specify the host on which the job manager is running
                   by using -jobmanagerhost.  The worker will then use
                   unicast to contact the job manager lookup process on
                   that host in  order to register with the job manager.
                   This overrides the setting of JOB_MANAGER_HOST in the
                   mdce_def file on the worker computer, which would also
                   have the worker use unicast.
   -multicast      If you are certain that multicast works on your
                   network, you can force the worker to use multicast to
                   locate the job manager lookup process by specifying
                   -multicast.
                   NOTE: If you are using this flag to change the settings
                   of and restart a stopped worker, then you should also
                   use the -clean flag.
```

```
-remotehost     Specify the name of the computer where you want to
                start the MATLAB worker.  If omitted, the worker is
                started on the local computer.

-clean          Delete all checkpoint information associated with this
                worker name before starting.

-baseport       Specify 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              Be verbose.  Display the progress of the command
                execution.

Examples:       1) Start a worker on the local host, using the default
                worker name, registering with the job manager
                MyJobManager on the host JMHost.

                startworker.sh -jobmanagername MyJobManager
                    -jobmanagerhost JMHost

                2) Start a worker on the host WorkerHost, using the
                default worker name, and registering with the job
                manager MyJobManager on the host JMHost.

                startworker.sh -jobmanagername MyJobManager
                    -jobmanagerhost JMHost -remotehost WorkerHost

                3) Start two workers, named worker1 and worker2, on the
                host WorkerHost, registering with the job manager
                MyJobManager on the host JMHost.  Note that in order to
                start two workers on the same computer, we have to give
                them different names.

               startworker.sh -name worker1 -jobmanagername MyJobManager
                    -jobmanagerhost JMHost -remotehost WorkerHost
               startworker.sh -name worker2 -jobmanagername MyJobManager
                    -jobmanagerhost JMHost -remotehost WorkerHost

See also:       mdce, nodestatus, startjobmanager, stopjobmanager,
                stopworker
```

Purpose

Stop job manager process

Syntax

```
stopjobmanager
```

Description

```
stopjobmanager: Stop a job manager that is running under the mdce
                service.

Usage:  stopjobmanager [ -name job_manager_name ]
                       [ -remotehost hostname ]
                       [ -baseport port_number ]
                       [ -v ]

-name           Specify the name of the job manager to stop.  The
                default is the value of DEFAULT_JOB_MANAGER_NAME
                parameter the mdce_def file.

-remotehost     Specify 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.

-baseport       Specify 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              Be verbose.  Display the progress of the command
                execution.

Examples:       1) Stop the job manager MyJobManager on the local host.

                stopjobmanager.sh -name MyJobManager

                2) Stop the job manager MyJobManager on the host JMHost.

                stopjobmanager.sh -name MyJobManager -remotehost JMHost

See also:       mdce, nodestatus, startjobmanager, startworker,
                stopworker
```

stopworker

Purpose

Stop MATLAB worker session

Syntax

```
stopworker
```

Description

```
   stopworker:     Stop a MATLAB worker process that is running under the
                   mdce service.

   Usage:  stopworker [ -name worker_name ]
                      [ -remotehost hostname ]
                      [ -baseport port_number ]
                      [ -v ]

   -name           Specify 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     Specify the name of the host where you want to stop the
                   MATLAB worker.  The default value is the local host.

   -baseport       Specify 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              Be verbose.  Display the progress of the command
                   execution.

   Examples:       1) Stop the worker with the default name on the local
                   host.

                   stopworker.sh

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

                   stopworker.sh -remotehost WorkerHost

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

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

See also: mdce, nodestatus, startjobmanager, stopjobmanager, startworker

Glossary

CHECKPOINT-BASE	The name of the parameter in the `mdce_def` file that defines the location of the job manager and worker checkpoint directories.
checkpoint directory	Location where job manager checkpoint information and worker checkpoint information is stored.
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.
cluster	A collection of computers that are connected via a network and intended for a common purpose.
computer	A system with one or more processors.
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.
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. Distributed applications can be either coarse-grained or find-grained.
distributed computing	Computing with distributed applications, running the application on several nodes simultaneously.
distributed computing demos	Demonstration programs that use the Distributed Computing Toolbox, as opposed to sequential demos.
DNS	Domain Name System. A system that translates Internet domain names into IP addresses.
head node	Usually, the node of the cluster designated for running the job manager 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.
job	The complete large-scale operation to perform in MATLAB, composed of a set of tasks.

job manager	The MathWorks process that queues jobs and assigns tasks to workers. A third-party process that performs this function is called a scheduler. The general term “scheduler” can also refer to a job manager.
job manager checkpoint information	Snapshot of information necessary for the job manager to recover from a system crash or reboot.
job manager database	The database that the job manager uses to store the information about its jobs and tasks.
job manager lookup process	The process that allows clients, workers, and job managers to find each other. It starts automatically when the job manager starts.
lab	When workers start, they work independently by default. They can then connect to each other and work together as peers, and are then referred to as labs.
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 manager	See job manager.
MATLAB worker	See worker.
mdce	The service that has to run on all machines before they can run a job manager or worker. This is the engine foundation process, making sure that the job manager and worker processes that it controls are always running. Note that the program and service name is all lower-case 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 labs 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 labs simultaneously, with communication, shared data, or synchronization points between the labs.
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 job manager are started and the worker contacts job manager.
scheduler	The process, either third-party or the MathWorks job manager, that queues jobs and assigns tasks to workers.
task	One segment of a job to be evaluated by a worker.
worker	The MATLAB process 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.

Index