

## X psLph kji uif N DS

### On this page...

[About the MATLAB Compiler Runtime \(MCR\)](#)

[The MCR Installer](#)

[Installing the MCR Non-Interactively](#)

[Uninstalling the MCR](#)

[MCR Startup Options](#)

[Using the MCR User Data Interface](#)

[Displaying MCR Initialization Start-Up and Completion Messages For Users](#)

### About the MATLAB Compiler Runtime (MCR)

The MATLAB<sup>®</sup> Compiler™ Runtime (MCR) is a standalone set of shared libraries that enables the execution of MATLAB files on computers without an installed version of MATLAB. Applications that use components built with MATLAB Compiler require access to an appropriate version of the MCR to run.

End-users of compiled components without access to MATLAB must install the MATLAB Compiler Runtime (MCR) on their computers or know the location of a network-installed MCR. The installers generated by the compiler app include the MCR installer. If you compiled your component using `mcc`, you should direct your end-users to download the MCR installer from the Web at

<http://www.mathworks.com/products/compiler/mcr>.

See [The MCR Installer](#) for more information.

### How is the MCR Different from MATLAB?

This MCR differs from MATLAB in several important ways:

- In the MCR, MATLAB files are securely encrypted for portability and integrity.
- MATLAB has a desktop graphical interface. The MCR has all the MATLAB functionality without the graphical interface.
- The MCR is version-specific. You must run your applications with the version of the MCR associated with the version of MATLAB Compiler with which it was created. For example, if you compiled an application using version 4.10 (R2009a) of MATLAB Compiler, users who do not have MATLAB installed must have version 7.10 of the MCR installed. Use `mcrversion` to return the version number of the MCR.
- The MATLAB and Java<sup>®</sup> paths in an MCR instance are fixed and cannot be changed. To change them, you must first customize them within MATLAB.

### Performance Considerations and the MCR

MATLAB Compiler was designed to work with a large range of applications that use the MATLAB programming language. Because of this, run-time libraries are large.

Since the MCR technology provides full support for the MATLAB language, including the Java programming language, starting a compiled application takes approximately the same amount of time as starting MATLAB. The amount of resources consumed by the MCR is necessary in order to retain the power and functionality of a full version of MATLAB.

The MCR makes use of thread locking so that only one thread is allowed to access the MCR at a time. As a result, calls into the MCR are threadsafe for MATLAB Compiler generated libraries, COM objects, and .NET objects. On the other hand, this can impact performance.

### The MCR Installer

Download the MCR from the Web at <http://www.mathworks.com/products/compiler/mcr>.

### Installing the MCR

To install the MCR, users of your component must run the MCR Installer.

**Note:** When packaging compiled components for distribution, MATLAB Compiler can include the Web-based MCR installer, or the actual MCR installer, in the distribution package.

To install the MCR:

1. Start the MCR Installer.

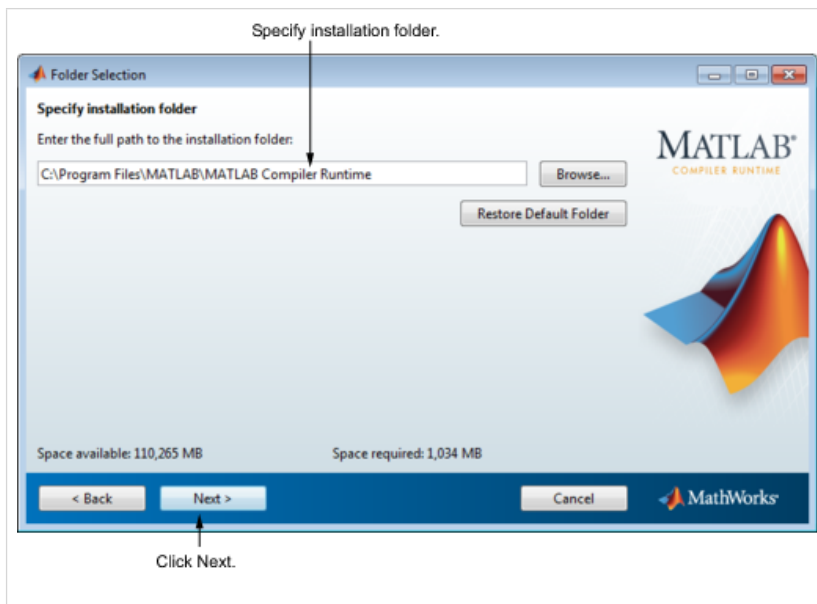
Computer	Steps
Windows®	Double-click the compiled component package self-extracting archive file, typically named <code>my_program_pkg.exe</code> , where <code>my_program</code> is the name of the compiled component. This extracts the MCR Installer from the archive, along with all the files that make up the MCR. Once all the files have been extracted, the MCR Installer starts automatically.
Linux®	Extract the contents of the compiled component package, which is a Zip file on Linux systems, typically named, <code>my_program_pkg.zip</code> , where <code>my_program</code> is the name of the compiled component. Use the <code>unzip</code> command to extract the files from the package.  <pre>unzip MCRInstaller.zip</pre> <p>Run the MCR Installer script, from the directory where you unzipped the package file, by entering:</p> <pre>./install</pre>
Mac	For example, if you unzipped the package and MCR Installer in <code>\home\USER</code> , you run the <code>./install</code> from <code>\home\USER</code> .  <div> <p><b>Note:</b> On Mac systems, you may need to enter an administrator username and password after you run <code>./install</code>.</p> </div>

2. When the MCR Installer starts, it displays the following dialog box. Read the information and then click **Next** to proceed with the installation.

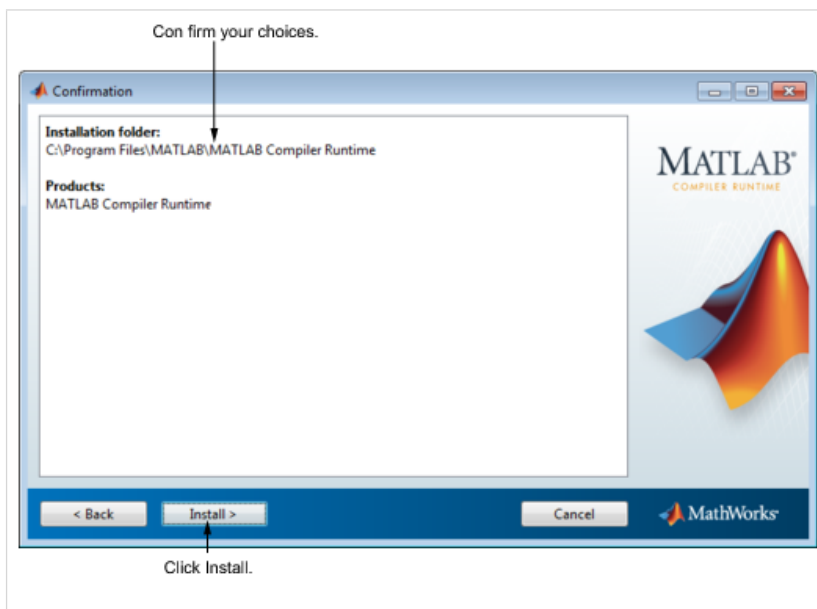


3. Specify the folder in which you want to install the MCR in the Folder Selection dialog box.

**Note:** On Windows systems, you can have multiple versions of the MCR on your computer but only one installation for any particular version. If you already have an existing installation, the MCR Installer does not display the Folder Selection dialog box because you can only overwrite the existing installation in the same folder. On Linux and Mac systems, you can have multiple installations of the same version of the MCR.



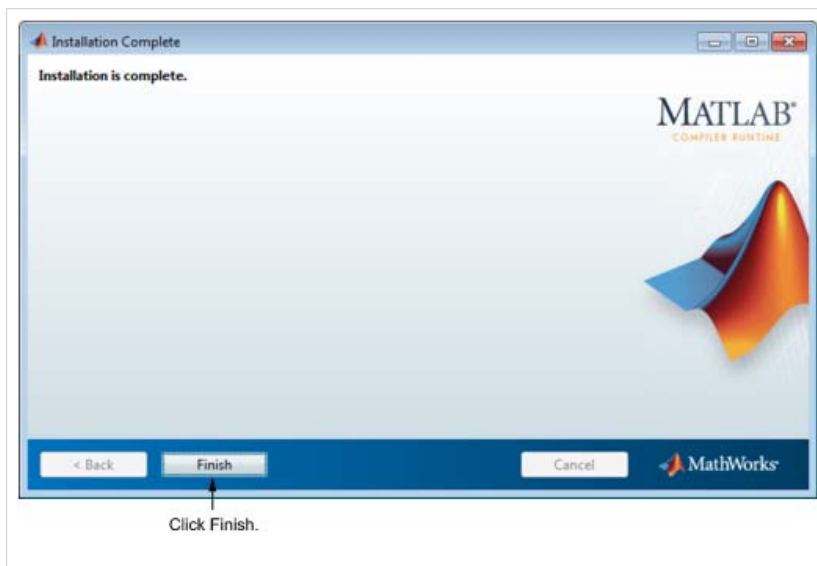
4. Confirm your choices and click **Next**. The MCR Installer starts copying files into the installation folder.



5. On Linux and Macintosh systems, after copying files to your disk, the MCR Installer displays the Product Configuration Notes dialog box. This dialog box contains information necessary for setting your path environment variables. Copy the path information from this dialog box and then click **Next**.



- Click **Finish** to exit the MCR Installer.



**MCR Installer Readme File.** A `readme.txt` file is included with the MCR Installer. This file, visible when the MCR Installer is expanded, provides more detailed information about the installer and the switches that can be used with it.

### Installing the MCR and MATLAB on the Same Machine

You do not need to install the MCR on your machine if your machine has both MATLAB and MATLAB Compiler installed. The version of MATLAB should be the same as the version of MATLAB that was used to create the deployed component.

You can, however, install the MCR for debugging purposes. See [Modifying the Path](#).

**Caution** There is a limitation regarding folders on your path. If the target machine has a MATLAB installation, the `<mcr_root>` folders must be first on the path to run the deployed application. To run MATLAB, the `matlabroot` folders must be first on the path. This restriction only applies to profiles involving an installed MCR and an installed MATLAB on the same machine.

**Modifying the Path.** If you install the MCR on a machine that already has MATLAB on it, you must adjust the library path according to your needs.

#### Windows

To run deployed components against the MCR install, `mcr_root\ver\runtime\win32|win64` must appear on your system path before `matlabroot\runtime\win32|win64`.

If `mcr_root\ver\runtime\arch` appears first on the compiled application path, the application uses the files in the MCR install area.

If `matlabroot\runtime\arch` appears first on the compiled application path, the application uses the files in the MATLAB Compiler installation area.

#### ■ UNIX®

To run deployed components against the MCR install, on Linux, Linux x86-64, or the `<mcr_root>/runtime/<arch>` folder must appear on your `LD_LIBRARY_PATH` before `matlabroot/runtime/<arch>`. See for the platform-specific commands.

To run deployed components on Mac OS X, the `<mcr_root>/runtime` folder must appear on your `DYLD_LIBRARY_PATH` before `matlabroot/runtime/<arch>`.

To run MATLAB on Mac OS X or Intel® Mac, `matlabroot/runtime/<arch>` must appear on your `DYLD_LIBRARY_PATH` before the `<mcr_root>/bin` folder.

**Note:** For detailed information about setting MCR paths on UNIX variants such as Mac and Linux, see [Using MATLAB Compiler on Mac or Linux](#) for complete deployment and troubleshooting information.

### Installing Multiple MCRs on One Machine

MCRInstaller supports the installation of multiple versions of the MCR on a target machine. This allows applications compiled with different versions of the MCR to execute side by side on the same machine.

If you do not want multiple MCR versions on the target machine, you can remove the unwanted ones. On Windows, run **Add or Remove Programs** from the Control Panel to remove any of the previous versions. On UNIX, you manually delete the unwanted MCR. You can remove unwanted versions before or after installation of a more recent version of the MCR, as versions can be installed or removed in any order.

**Note for Mac OS X Users** Installing multiple versions of the MCR on the same machine is not supported on Mac OS X. When you receive a new version of MATLAB, you must recompile and redeploy all of your applications and components. Also, when you install a new MCR onto a target machine, you must delete the old version of the MCR and install the new one. You can only have one version of the MCR on the target machine.

**Deploying a Recompiled Application.** Always run your compiled applications with the version of the MCR that corresponds to the MATLAB version with which your application was built. If you upgrade your MATLAB Compiler software on your development machine and distribute the recompiled application to your users, you should also distribute the corresponding version of the MCR. Users should upgrade their MCR to the new version. If users need to maintain multiple versions of the MCR on their systems, refer to [Installing Multiple MCRs on One Machine](#) for more information.

### Installing the MCR Non-Interactively

To install the MCR without having to interact with the installer dialog boxes, use one of the MCR installer's non-interactive modes:

- silent—the installer runs as a background task and does not display any dialog boxes
- automated—the installer displays the dialog boxes but does not wait for user interaction

When run in silent or automated mode, the MCR installer uses default values for installation options. You can override these defaults by using MCR installer command-line options or an installer control file.

**Note:** If you have already installed the MCR for a particular release in the default location, the installer overwrites the existing installation, when running in silent or automated mode.

### Running the Installer in Silent Mode

To install the MCR in silent mode:

1. Extract the contents of the MCR installer file to a temporary folder, called `$temp` in this documentation.

**Note:** On Windows systems, **manually** extract the contents of the installer file.

2. Run the MCR installer, specifying the `-mode` option and `-agreeToLicense yes` on the command line.

**Note:** On most platforms, the installer is located at the root of the folder into which the archive was extracted. On Windows 64, the installer is located in the `archives\bin` folder.

Platform	Command
Windows	<code>setup -mode silent -agreeToLicense yes</code>
Linux	<code>./install -mode silent -agreeToLicense yes</code>
Mac OS X	<code>./install -mode silent -agreeToLicense yes</code>

**Note:** If you do not include the `-agreeToLicense yes` the installer will not install the MCR.

3. View a log of the installation.

On Windows systems, the MCR installer creates a log file, named `mathworks_username.log`, where `username` is your Windows log-in name, in the location defined by your `TEMP` environment variable.

On Linux and Mac systems, the MCR installer displays the log information at the command prompt, unless you redirect it to a file.

### Customizing a Non-Interactive Installation

When run in one of the non-interactive modes, the installer will use the default values unless told to do otherwise. Like the MATLAB installer, the MCR installer accepts a number of command line options that modify the default installation properties.

Option	Description
<code>-destinationFolder</code>	Specifies where the MCR will be installed.
<code>-outputFile</code>	Specifies where the installation log file is written.
<code>-automatedModeTimeout</code>	Specifies how long, in milliseconds, that the dialog boxes are displayed when run in automatic mode.
<code>-inputFile</code>	Specifies an installer control file with the values for all of the above options.

**Note:** The MCR installer archive includes an example installer control file called `installer_input.txt`. This file contains all of the options available for a full MATLAB installation. Only the options listed in this section are valid for the MCR installer.

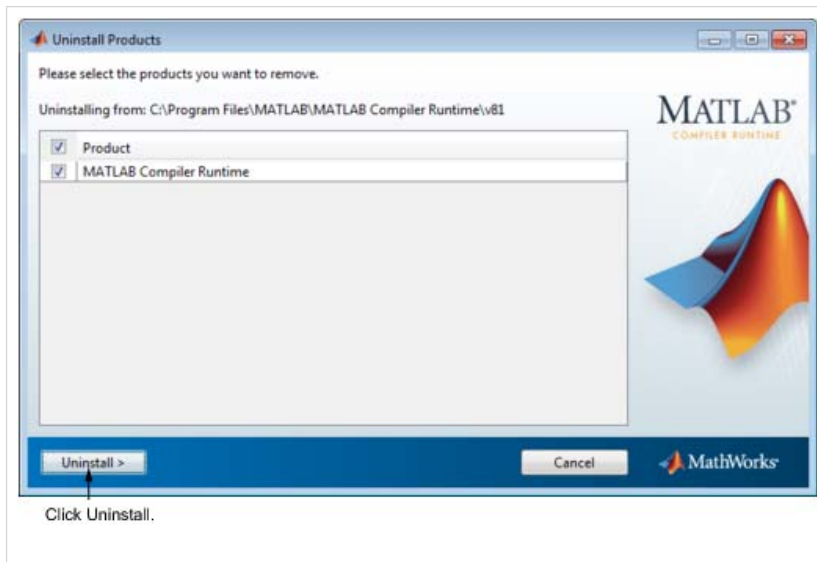
### Uninstalling the MCR

The method you use to uninstall the MCR from your computer varies depending on the type of computer.

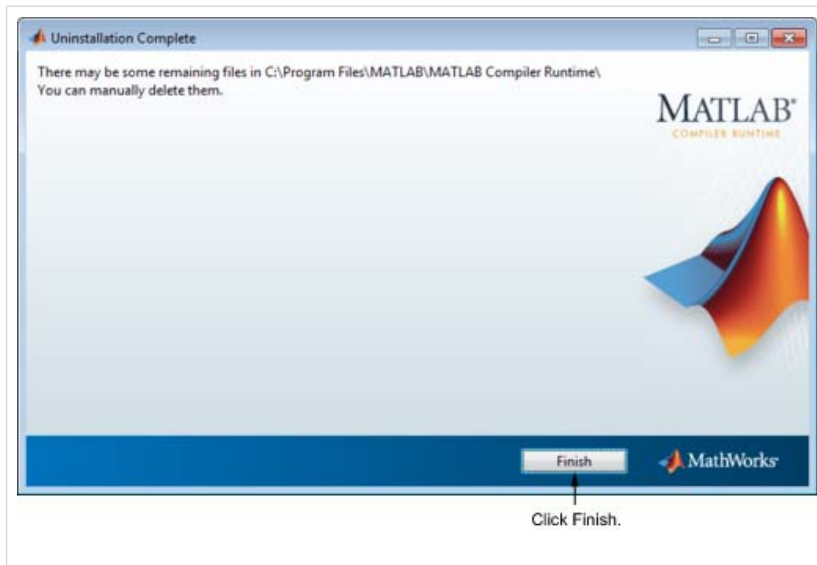
You can remove unwanted versions before or after installation of a more recent version of the MCR, as versions can be installed or removed in any order.

#### Windows

1. Start the uninstaller. From the Windows Start menu, search for the **Add or Remove Programs** control panel, and double-click MATLAB Compiler Runtime in the list. You can also launch the MCR Uninstaller from the `mcr_root\uninstall\bin\arch` folder, where `mcr_root` is your MCR installation folder and `arch` is an architecture-specific folder, such as `win64`.
2. Select the MATLAB Compiler Runtime from the list of products in the Uninstall Products dialog box and click **Next**.



- After the MCR uninstaller removes the files from your disk, it displays the Uninstallation Complete dialog box. Click **Finish** to exit the MCR uninstaller.



## Linux

- Exit the application.
- Enter this command at the Linux prompt:

```
rm -rf mcr_root
```

where *mcr\_root* represents the name of your top-level MATLAB installation folder.


## Mac

- Exit the application.
- Navigate to your MCR installation folder. For example, the installation folder might be named MATLAB\_Compiler\_Runtime.app in your Applications folder.  
where *mcr\_root* represents the name of your top-level MATLAB installation folder.
- Drag your MCR installation folder to the trash, and then select **Empty Trash** from the Finder menu.

## MCR Startup Options

### Setting MCR Options

Set MCR options, such as `-nojvm`, `-nodisplay`, or `-logfile` by performing either of the following tasks.

- Using the Deployment Tool, click the Settings icon (  ). Using the **Advanced** tab in the Project Settings dialog box, enter the options you want to select in the **MCR Startup Options** area.
- Using the `mcc` command, specify the `-R` switch.

**Using the Deployment Tool.** In the MCR Startup Options area of the Project Settings dialog box, you can set the following options.

#### Setting MCR Startup Options Using the Deployment Tool GUI

MCR Startup Option	This option...	Set the options by...
<code>-nojvm</code>	Disables the Java Virtual Machine, which is enabled by default. This can help improve MCR performance.	Select the <b>Disable JVM</b> checkbox.
<code>-nodisplay</code>	On Linux, launches the MCR without display functionality.	In the <b>MCR Options</b> box, enter <code>-nodisplay</code> .
<code>-logfile</code>	Writes information about MCR startup to a logfile.	Select the <b>Create log file</b> checkbox. You <i>must</i> enter the path to the logfile, including the logfile name, in the <b>Log File</b> box.

**Setting MCR Startup Options Using the `mcc` Command Line.** When you use the command line, specify the `-R` switch to invoke the MCR startup options you want to use.

Following are examples of using `mcc -R` to invoke `-nojvm`, `-nodisplay`, and `-logfile` when building a C standalone (designated by the `-m` switch).

- ▶ [Setting -nojvm](#)
- ▶ [Setting -nodisplay \(Linux Only\)](#)
- ▶ [Setting -logfile](#)
- ▶ [Setting -nojvm, -nodisplay, and -logfile With One Command](#)

#### Retrieving MCR Startup Options (Shared Libraries Only)

Use these functions to return data about MCR state when working with shared libraries (this does not apply to standalone applications).

Function and Signature	When to Use	Return Value
<code>bool mclIsMCRInitialized()</code>	Use <code>mclIsMCRInitialized()</code> to determine whether or not the MCR has been properly initialized.	Boolean (true or false). Returns <code>true</code> if MCR is already initialized, else returns <code>false</code> .
<code>bool mclIsJVMEEnabled()</code>	Use <code>mclIsJVMEEnabled()</code> to determine if the MCR was launched with an instance of a Java Virtual Machine (JVM).	Boolean (true or false). Returns <code>true</code> if MCR is launched with a JVM instance, else returns <code>false</code> .
<code>const char* mclGetLogFileName()</code>	Use <code>mclGetLogFileName()</code> to retrieve the name of the log file used by the MCR	Character string representing log file name used by MCR
<code>bool mclIsNoDisplaySet()</code>	Use <code>mclIsNoDisplaySet()</code> to determine if <code>-nodisplay</code> option is enabled.	Boolean (true or false). Returns <code>true</code> if <code>-nodisplay</code> is enabled, else returns <code>false</code> .  <b>Note:</b> <code>false</code> is always returned on Windows systems since the <code>-nodisplay</code> option is not supported on Windows systems.  <b>Caution</b> When running on Mac, if <code>-nodisplay</code> is used as one of the options included in



Function and Signature	When to Use	Return Value
		<div> <p><code>mclInitializeApplication</code>, then the call to <code>mclInitializeApplication</code> must occur before calling <code>mclRunMain</code>.</p> </div>

**Note:** All of these attributes have properties of write-once, read-only.

### Retrieving Information About MCR Startup Options.

```
const char* options[4];
options[0] = "-logfile";
options[1] = "logfile.txt";
options[2] = "-nojvm";
options[3] = "-nodisplay";
if( !mclInitializeApplication(options,4) )
{
    fprintf(stderr,
            "Could not initialize the application.\n");
    return -1;
}
printf("MCR initialized : %d\n", mclIsMCRInitialized());
printf("JVM initialized : %d\n", mclIsJVMEEnabled());
printf("Logfile name : %s\n", mclGetLogFileName());
printf("nodisplay set : %d\n", mclIsNoDisplaySet());
fflush(stdout);
```

### Using the MCR User Data Interface

The MCR User Data Interface lets you easily access MCR data. It allows keys and values to be passed between an MCR instance, the MATLAB code running on the MCR, and the host application that created the MCR. Through calls to the MCR User Data Interface API, you access MCR data by creating a per-MCR-instance associative array of `mxArrays`, consisting of a mapping from string keys to `mxArray` values. Reasons for doing this include, but are not limited to the following:

- You need to supply run-time profile information to a client running an application created with the Parallel Computing Toolbox. You supply and change profile information on a per-execution basis. For example, two instances of the same application may run simultaneously with different profiles. See [Deploy Applications Created Using Parallel Computing Toolbox](#) for more information.
- You want to set up a global workspace, a global variable or variables that MATLAB and your client can access.
- You want to store the state of any variable or group of variables.

The API consists of:

- Two MATLAB functions callable from within deployed application MATLAB code
- Four external C functions callable from within deployed application wrapper code

**Note:** The MATLAB functions are available to other modules since they are native to MATLAB. These built-in functions are implemented in the MCLMCR module, which lives in the standalone folder.

For implementations using .NET components, Java components, or COM components with Excel, see the MATLAB Builder™ NE, MATLAB Builder JA, and MATLAB Builder EX documentation, respectively.

### MATLAB Functions

Use the MATLAB functions `getmcruserdata` and `setmcruserdata` from deployed MATLAB applications. They are loaded by default only in applications created with the MATLAB Compiler or builder products. See [Using the MCR User Data Interface](#) for more information.

**Tip** `getmcrcruserdata` and `setmcrcruserdata` will produce an Unknown function error when called in MATLAB if the MCLMCR module cannot be located. This can be avoided by calling `isdeployed` before calling `getmcrcruserdata` and `setmcrcruserdata`. For more information about the `isdeployed` function, see the [isdeployed](#) reference page.

### Setting MCR Data for Standalone Executables

MCR data can be set for a standalone executable with the `-mcrcruserdata` command line argument.

The following example demonstrates how to set MCR user data for use with a Parallel Computing Toolbox profile:

```
parallelapp.exe -mcrcruserdata
    ParallelProfile:config.settings
```

The argument following `-mcrcruserdata` is interpreted as a key/value MCR user data pair, where the colon separates the key from the value. The standalone executable accesses this data by using `getmcrcruserdata`.

**Note:** A compiled application should set `mcrcruserdata ParallelProfile` *before* calling any Parallel Computing Toolbox™ code. Once this code has been called, setting `ParallelProfile` to point to a different file has no effect.

### Setting and Retrieving MCR Data for Shared Libraries

As mentioned in [Using the MCR User Data Interface](#), there are many possible scenarios for working with MCR Data. The most general scenario involves setting the MCR with specific data for later retrieval, as follows:

1. In your code, Include the MCR header file and the library header generated by MATLAB Compiler.
2. Properly initialize your application using `mclInitializeApplication`.
3. After creating your input data, write or "set" it to the MCR with `setmcrcruserdata`.
4. After calling functions or performing other processing, retrieve the new MCR data with `getmcrcruserdata`.
5. Free up storage memory in work areas by disposing of unneeded arrays with `mxDestroyArray`.
6. Shut down your application properly with `mclTerminateApplication`.

### Displaying MCR Initialization Start-Up and Completion Messages For Users

You can display a console message for end users that informs them when MCR initialization starts and completes.

To create these messages, use the `-R` option of the `mcc` command.

You have the following options:

- Use the default start-up message only (Initializing MATLAB Compiler Runtime version x.xx)
- Customize the start-up or completion message with text of your choice. The default start-up message will also display prior to displaying your customized start-up message.

Some examples of different ways to invoke this option follow:

This command:	Displays:
<code>mcc -R -startmsg</code>	Default start-up message Initializing MATLAB Compiler Runtime version x.xx
<code>mcc -R -startmsg,'user customized message'</code>	Default start-up message Initializing MATLAB Compiler Runtime version x.xx and user customized message for start-up
<code>mcc -R -completemsg,'user customized message'</code>	Default start-up message Initializing MATLAB Compiler Runtime version x.xx and user customized message for completion
<code>mcc -R -startmsg,'user customized message' -R -completemsg,'user customized message'</code>	Default start-up message Initializing MATLAB Compiler Runtime version x.xx and user customized message for both start-up and completion by specifying <code>-R</code> before each option

This command:	Displays:
<code>mcc -R -startmsg,'user customized message',-completemsg,'user customized message'</code>	Default start-up message Initializing MATLAB Compiler Runtime version x.xx and user customized message for both start-up and completion by specifying -R only once

### Best Practices

Keep the following in mind when using `mcc -R`:

- When calling `mcc` in the MATLAB Command Window, place the comma inside the single quote. For example:

```
mcc -m hello.m -R '-startmsg,"Message_Without_Space"'
```

- If your initialization message has a space in it, call `mcc` from the system console or use `!mcc` from MATLAB.