Working with the MCR

On this page…

About the MATLAB Compiler Runtime (MCR)

MATLAB Compiler uses the MATLAB Compiler Runtime (MCR), a standalone set of shared libraries that enables the execution of MATLAB files on computers without an installed version of MATLAB.

If you distribute your compiled components to end-users who do not have MATLAB installed on their systems, they must install the MATLAB Compiler Runtime (MCR) on their computers or know the location of a network-installed MCR. When you packaged your compiled component, you have the option of including the MCR in the package you distribute to users, or they can download it from the Web at `http://www.mathworks.com/products/compiler/mcr`. The MCR only needs to be installed once a user system.

> **Note:** There is no way to distribute your application with any subset of the files that are installed by the MCR Installer.

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 of MATLAB's 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 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. When you packaged your compiled component for distribution, you had the option to embed the MCR in your application package, or specify the network location of the MCR.

To install the MCR on any computer, perform these steps.

1. Start the MCR Installer. How you accomplish this depends on your computer.

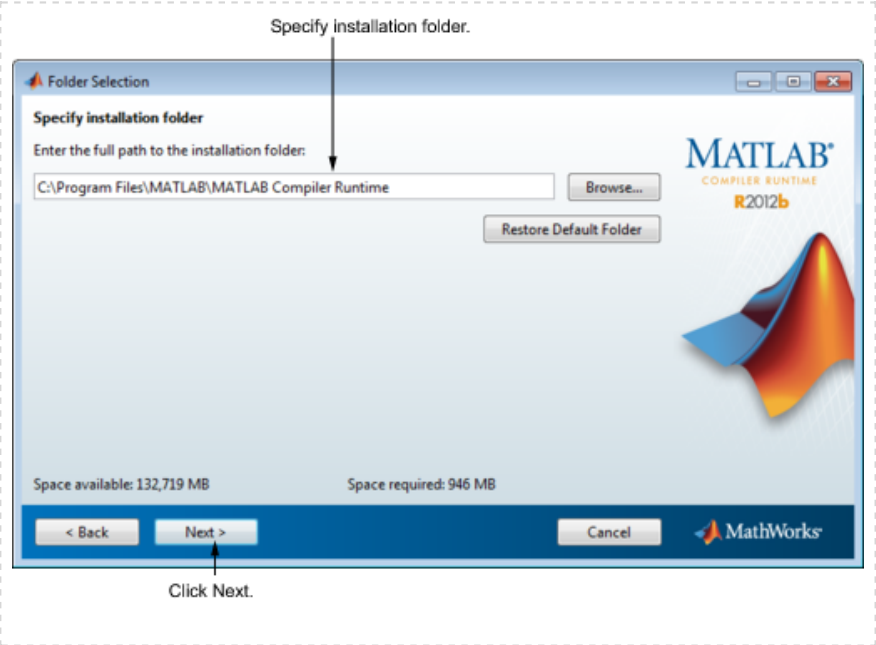
Computer	Steps
Windows	Double-click the compiled component package self-extracting archive file, typically named *my_program*`_pkg.exe`, where *my_program* is the name of the compiled component. This extracts the MCR Installer from the archive, along with all the files that make up the MCR. Once all the files have been extracted, the MCR Installer starts automatically.

Linux
Mac
	Extract the contents of the compiled component package, which is a Zip file on Linux systems, typically named, *my_program*`_pkg.zip`, where *my_program* is the name of the compiled component. Use the `unzip` command to extract the files from the package.
	`unzip MCRInstaller.zip`
	Run the MCR Installer script, from the directory where you unzipped the package file, by entering:
	`./install`
	For example, if you unzipped the package and MCR Installer in `\home\USER`, you run the `./install` from `\home\USER`.
	Note: On Mac systems, you may need to enter an administrator username and password after you run `./install`.

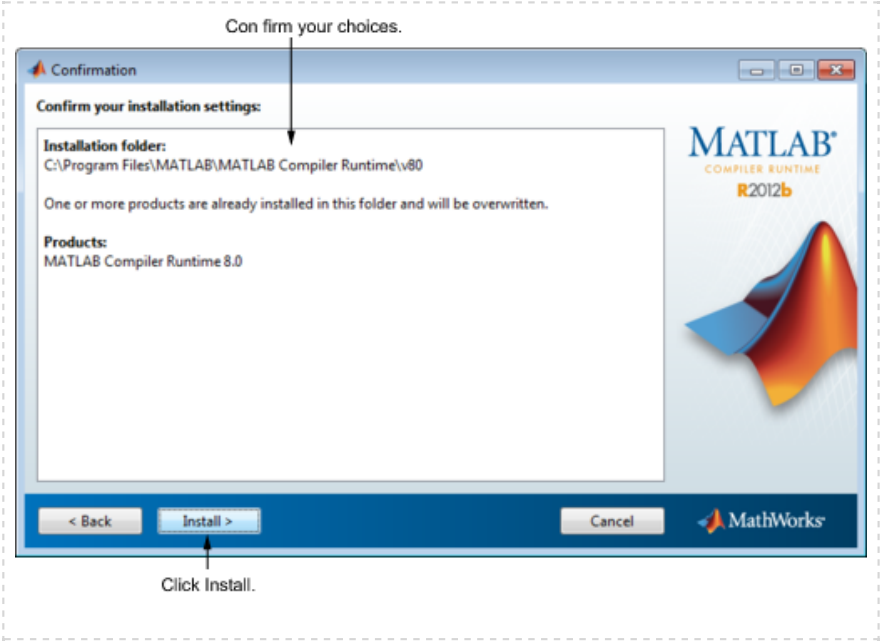
2. When the MCR Installer starts, it displays the following dialog box. Read the information and then click **Next** to proceed with the installation.

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

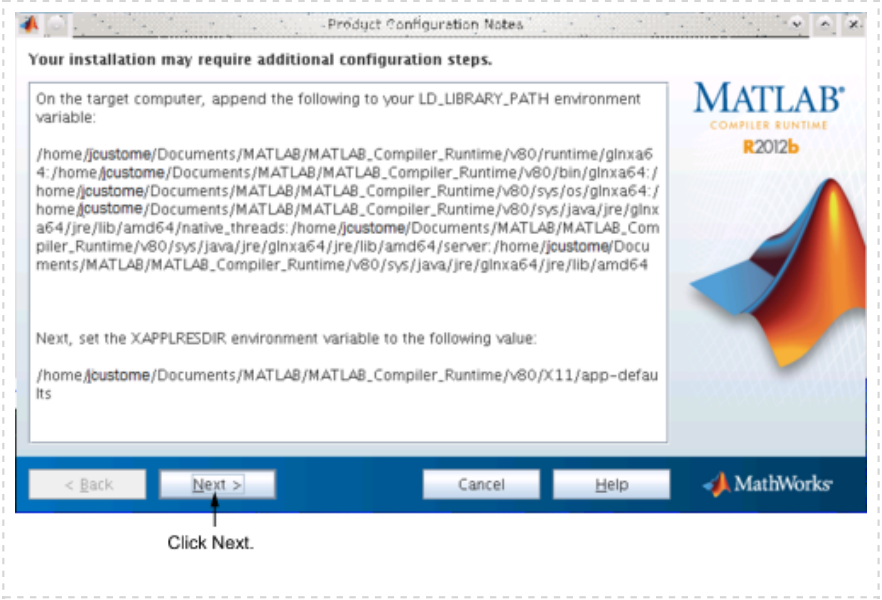
   Note: On Windows systems, you can have multiple installations of different versions of the MCR on your computer but only one installation for any particular version. If you already have an existing installation, the MCR Installer does not display the Folder Selection dialog box because you can only overwrite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



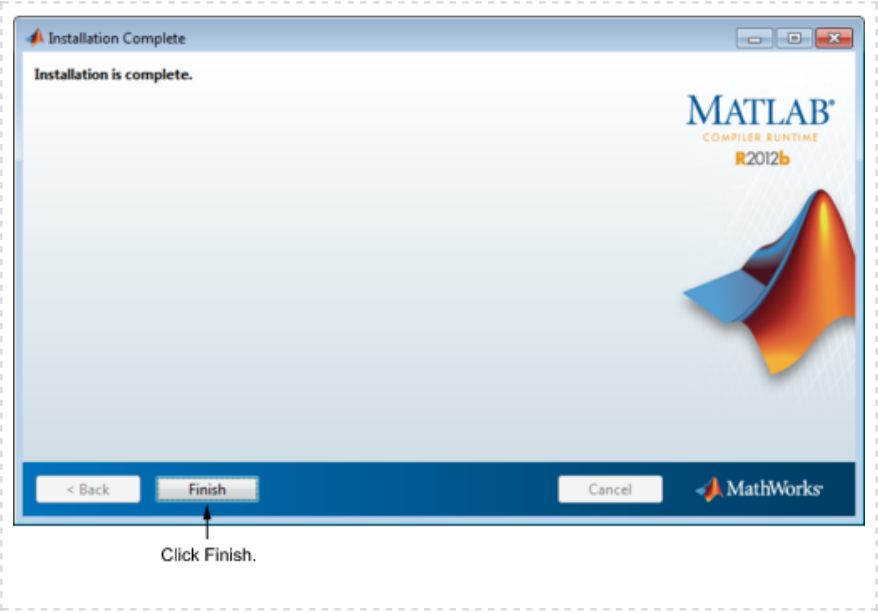
4. Confirm your choices and click **Next**. The MCR Installer starts copying files into the installation folder.



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**.



6. 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>`, and `XAPPLRESDIR` should point to `<mcr_root>/X11/app-defaults`. 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>`, and `XAPPLRESDIR` should point to `<mcr_root>/X11/app-defaults`.

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, and `XAPPLRESDIR` should point to *`matlabroot`*`/X11/app-defaults`.

> **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 (Silent Mode)

To install the MCR without having to interact with the installer dialog boxes, use one of the MCR installer non-interactive modes: silent or automated.

Mode	Description
`silent`	MCR installer runs as a background task and does not display any dialog boxes.
`automated`	MCR 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, such as the name of the destination folder. You can override these defaults by using MCR installer command-line options.

> **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.

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

   On Windows systems, double-click the MCR installer self-extracting archive file, `MCRinstaller.exe`. You might have to first extract the MCR installer from the compiled component archive, if you received a package file from the component developer.

   On Linux and Mac systems, use the `unzip` command:

   ```
   unzip MCRInstaller.zip
   ```

2. Run the MCR installer, specifying the `mode` option on the command line.

   The install script is created in the folder in which you have unzipped the MCR.

   Execute the MCR installer, specifying the mode argument.

   ```
   setup.exe -mode silent
   ```

   On a Linux or Mac computer, run the MCR installer script, specifying the mode argument.

   ```
   ./install -mode silent
   ```

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 Silent Installation

If you want to specify the installation folder or the name of the log file, use MCR installer command line options. For example, to specify the installation folder, use the `-destinationFolder` option, as follows:

```
setup.exe -mode silent -destinationFolder C:\my_folder
```

To specify the name and location of the installation log file, use the `-outputFile` option, as follows:

```
setup.exe -mode silent -destinationFolder C:\my_folder
                               -outputFile C:\my_log.txt
```

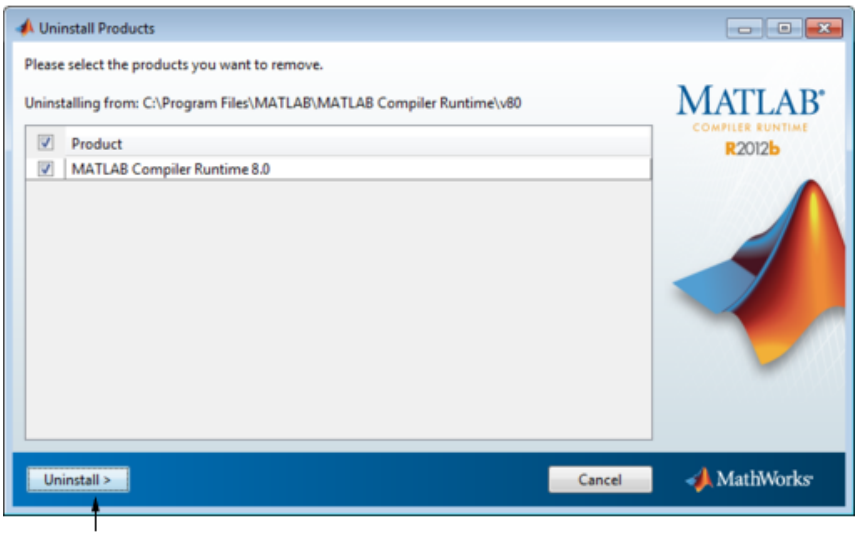
Removing (Uninstalling) the MCR

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

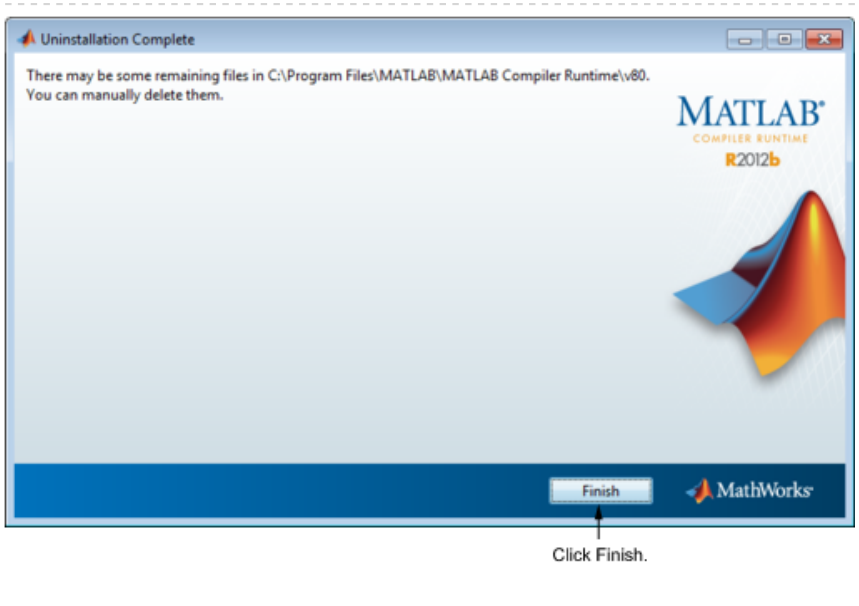
You can remove unwanted versions before or after installation of a more recent version of the MCR, as versions can be installed or removed in any order.

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**.



3. 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

1. Exit the application.
2. 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.

Retrieving MCR Attributes

Use these new 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
`bool mclIsMCRInitialized()`	Use `mclIsMCRInitialized()` to determine whether or not the MCR has been properly initialized.	Boolean (`true` or `false`). Returns `true` if MCR is already initialized, else returns `false`.
`bool mclIsJVMEnabled()`	Use `mclIsJVMEnabled()` to determine if the MCR was launched with an instance of a Java Virtual Machine (JVM).	Boolean (`true` or `false`). Returns `true` if MCR is launched with a JVM instance, else returns `false`.
`const char* mclGetLogFileName()`	Use `mclGetLogFileName()` to retrieve the name of the log file used by the MCR	Character string representing log file name used by MCR
`bool mclIsNoDisplaySet()`	Use `mclIsNoDisplaySet()` to determine if `-nodisplay` option is enabled.	Boolean (`true` or `false`). Returns `true` if `-nodisplay` is enabled, else returns `false`. **Note:** `false` is always returned on Windows systems since the `-nodisplay` option is not supported on Windows systems. **Caution** When running on Mac, if `-nodisplay` is used as one of the options included in `mclInitializeApplication`, then the call to `mclInitializeApplication` must occur before calling `mclRunMain`.

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

Retrieving Information from MCR State

```
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", mclIsJVMEnabled());
printf("Logfile name : %s\n", mclGetLogFileName());
printf("nodisplay set : %d\n", mclIsNoDisplaySet());
fflush(stdout);
```

Improving Data Access 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 wrapper code 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 `mxArray`s, 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 Improving Data Access Using the MCR User Data Interface for more information.

> **Tip** When calling the MATLAB functions `getmcruserdata` and `setmcruserdata`, remember the following:
>
> - These functions 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 `getmcruserdata` and `setmcruserdata`. For more information about the `isdeployed` function,

see the `isdeployed` reference page.

- The MATLAB functions `getmcruserdata` and `setmcruserdata` can be dragged and dropped (as you would any other MATLAB file), directly to the `deploytool` GUI.

Setting MCR Data for Standalone Executables

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

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

```
parallelapp.exe -mcruserdata
                ParallelConfigurationFile:config.mat
```

The argument following `-mcruserdata` 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 `getmcruserdata`.

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

Setting and Retrieving MCR Data for Shared Libraries

As mentioned in Improving Data Access 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 `setmcruserdata` .
4. After calling functions or performing other processing, retrieve the new MCR data with `getmcruserdata`.
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:
`mcc -R -startmsg`	Default start-up message `Initializing MATLAB Compiler Runtime version x.xx`
`mcc -R -startmsg,'user customized message'`	Default start-up message `Initializing MATLAB Compiler Runtime version x.xx` and *user customized message* for start-up
`mcc -R -completemsg,'user customized message'`	Default start-up message `Initializing MATLAB Compiler Runtime version x.xx` and *user customized message* for completion
`mcc -R -startmsg,'user customized message' -R -completemsg,'user customized message"`	Default start-up message `Initializing MATLAB Compiler Runtime version x.xx` and *user customized message* for both start-up and completion by specifying `-R` before each option
`mcc -R -startmsg,'user customized message',-completemsg,'user customized message'`	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.

Was this topic helpful? Yes No