MATLAB® Compiler SDK™ Release Notes

MATLAB®

MathWorks®

How to Contact MathWorks

Latest news:	`www.mathworks.com`
Sales and services:	`www.mathworks.com/sales_and_services`
User community:	`www.mathworks.com/matlabcentral`
Technical support:	`www.mathworks.com/support/contact_us`
Phone:	508-647-7000

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

MATLAB® Compiler SDK™ Release Notes

Contents

R2021a

R2020b

R2023a

.NET Assembly Integration: Deploy MATLAB functions within .NET applications from Linux and macOS systems

You can deploy MATLAB functions within .NET applications from Linux® and macOS systems using the `compiler.build.dotNETAssembly` function or the `mcc` command. This functionality is supported only if you use the MATLAB Data API for .NET to exchange data between the deployed MATLAB functions and the .NET application, and not the MWArray API.

For details, see “Deploy MATLAB Code Within .NET Application Using MATLAB Data API for .NET”.

Python Version Support for Python Package Integration

Support	**Python® Version**	**Platform**
Removed	Python 2.7	All platforms

R2022b

MATLAB Runtime: Installation path updated to use MATLAB release name

MATLAB Runtime now installs to a folder whose name uses the corresponding MATLAB release name.

Previously, the default path to an installation of MATLAB Runtime included the MATLAB Runtime version, such as `v912` in R2022a. The new installer uses the MATLAB release name in the installation path, for example, `C:\Program Files\MATLAB\MATLAB Runtime\R2022b`.

Application Deployment: Create a Deployment Script using the createDeploymentScript function

You can create a MATLAB deployment script from a MATLAB Compiler SDK project file (`.prj`) using the `createDeploymentScript` function. This function converts the project deployment settings to the corresponding `compiler.build` commands. The generated MATLAB script creates a compiled component using the same settings as the MATLAB Compiler SDK project.

Packaging: Obfuscate all MATLAB code prior to packaging

You can use the `mcc` command with the `-j` option to automatically convert all `.m` files to P-files before packaging. This option generates a P-code file with a `.p` extension for each `.m` file included in the `mcc` command. P-code files are an obfuscated, execute-only form of MATLAB code. For more details, see `pcode`.

Packaging: Specify encryption key for packaging MATLAB code

You can use the `mcc` command with the `-k` option to specify an AES encryption key and a MEX-file loader interface to retrieve the decryption key at runtime. If you do not specify any arguments after `-k`, `mcc` generates a 256-bit AES key and a loader MEX-file.

Python Package Integration: Deployed functions accept any object using Python buffer protocol

Deployed functions in a compiled Python package now accept any object that satisfies the Python buffer protocol as an input argument. For details, see `docs.python.org/3/c-api/buffer.html`.

For example, `sumwrap` is a deployed wrapper function that calls the built-in MATLAB function `sum`, and the input `bufferarray` is a NumPy ndarray.

```
import matlab
# sumwrap(arr, dim) is a deployed function that calls the built-in
# function sum(arr, dim)
import sumwrap
import numpy
sumwrap_mod_handle = sumwrap.initialize()
bufferarray = numpy.array([[1, 2, 3], [4, 5, 6]], dtype='uint16')

# Supported in R2022a and earlier: must initialize a matlab.uint16 from
# the numpy array and pass it into the function
array_as_matlab_uint16 = matlab.uint16(bufferarray)
```

```
sum_of_elems = sumwrap_mod_handle.sumwrap(array_as_matlab_uint16, 1, 'native')
print(sum_of_elems) # prints 'matlab.uint16([[5,7,9]])'

# Supported in R2022b and later: can pass the numpy array
# directly into the function
sum_of_elems = sumwrap_mod_handle.sumwrap(bufferarray, 1, 'native')
print(sum_of_elems) # prints 'matlab.uint16([[5,7,9]])'
```

Microservice Integration: Create microservice Docker images on Windows and macOS

In addition to Linux, you can create microservice Docker® images on Windows® and macOS operating systems from the MATLAB command prompt using the `compiler.package.microserviceDockerImage` function. This function, along with the `compiler.package.MicroserviceDockerImageOptions` function, provides an interface to specify various options associated with creating Docker images. The microservice image provides an HTTP or HTTPS endpoint to access MATLAB code.

NET Assembly Integration: Deploy MATLAB functions within .NET applications using the MATLAB Compiler SDK API for .NET and MATLAB Data API for .NET

You can create an archive containing MATLAB functions using the `compiler.build.dotNETAssembly` function and deploy the archive within a .NET application using the MATLAB Compiler SDK API for .NET. This API enables .NET applications to launch MATLAB Runtime instances and evaluate deployed MATLAB functions with arguments. The API supports asynchronous task execution for MATLAB library calls, and leverages modern .NET constructs for writing streamlined application code. Use the MATLAB Data API for .NET to handle the data exchange between a .NET application and deployed MATLAB code. For details, see Deploy to .NET Applications Using MATLAB Data API for .NET.

Compatibility Considerations

- To deploy MATLAB functions within .NET applications using the MATLAB Compiler SDK API for .NET and MATLAB Data API for .NET, we recommend you install .NET 5.0 or higher. This version of .NET ensures that the applications you develop can be run across multiple platforms. Although you can use MATLAB Compiler SDK API for .NET and MATLAB Data API for .NET with .NET Framework and .NET Core, consider the following limitations:
  - Deployed applications developed using .NET Framework can run only on Windows systems and not across multiple platforms.
  - Only the following .NET Framework versions are supported: `4.6.1`, `4.6.2`, `4.7`, `4.7.1`, `4.7.2`, `4.8`
  - According to Microsoft®, .NET Framework 4.8 is the final version of .NET Framework that will be released.
  - .NET Framework does not have a command-line interface for development.
  - As of R2022b, .NET Core versions `2.0` to `3.0` are already out of support from Microsoft, and .NET Core `3.1` is in a maintenance phase. According to Microsoft, .NET 5.0 and higher represent the successor to .NET Core and are part of the company's plan for a unified .NET stack and ecosystem.

- If you are using Visual Studio® as your development environment, you must use version 16.8 or higher.

MathWorks recommends using the MATLAB Compiler SDK API for .NET and MATLAB Data API for .NET with .NET 5.0 or higher to support deployment across the broadest array of platforms.

.NET Assembly Integration: Support for .NET 6.0

You can create a .NET 6.0 assembly using the **Library Compiler** app and integrate it with a .NET 6.0 application. For details, see the Microsoft documentation.

Python Version Support for Python Package Integration

Support	Python Version	Platform
Added	Python 3.10	All platforms

Functionality Being Removed or Changed

MATLAB Runtime Installer: Automated Mode removed

The option to install MATLAB Runtime noninteractively using the `-mode automated` option has been removed.

You can install MATLAB Runtime in silent mode by using installer command-line options or an installer control file. In silent mode, the installer runs as a background task and does not display any dialog boxes.

MATLAB Compiler SDK Application Installers: Automated Mode removed

The option to install packaged MATLAB Compiler SDK applications noninteractively using the `-mode automated` option has been removed.

You can install applications in silent mode by using installer command-line options or an installer control file. In silent mode, the installer runs as a background task and does not display any dialog boxes.

MATLAB Compiler SDK Application Installers: Automated Mode removed

The option to install packaged MATLAB Compiler SDK applications noninteractively using the `-mode automated` option has been removed.

You can install applications in silent mode by using installer command-line options or an installer control file. In silent mode, the installer runs as a background task and does not display any dialog boxes.

Python 2.7 support will be removed
Still runs

Support for Python 2.7 will be removed in a future release. To build Python packages, use Python 3.7 or newer.

-build and -package options in the productionServerCompiler function will be removed
Warns

The `-build` and `-package` options in the `productionServerCompiler` function will be removed in a future release. To generate deployable archives, use the `compiler.build.productionServerArchive` function, or the `mcc` command, or the **Production Server Compiler** app. The other options of the `productionServerCompiler` function will not be changed.

The **Production Server Compiler** app will also not be changed. You can continue to the use the app to generate deployable archives.

-build and -package options in the deploytool function will be removed
Warns

The `-build` and `-package` options in the `deploytool` function will be removed in a future release. To build applications, use the `mcc` command. To package and create an installer, use the `compiler.package.installer` function. The other options of the `deploytool` function will not be changed.

R2022a

Microservice Integration: Create a microservice Docker image using the compiler.package.microserviceDockerImage function

You can create microservice Docker images on Linux operating systems from the MATLAB command prompt using the `compiler.package.microserviceDockerImage` function. This function, along with the `compiler.package.MicroserviceDockerImageOptions` function, provides an interface to specify various options associated with creating Docker images. The microservice image provides an HTTP or HTTPS endpoint to access MATLAB code.

.NET Assembly Integration: Support for .NET 5.0

You can create a .NET 5.0 assembly using the **Library Compiler** app and integrate it with a .NET 5.0 application. These applications can be created on Windows and run on Linux and macOS systems. Microsoft.NET 5.0 is the successor to .NET Core 3.1 and does not replace .NET Framework. For details, see the Microsoft documentation.

For an example, see Build .NET 5.0 Application That Runs on Linux and macOS.

Compatibility Considerations

- To use this functionality, you must have Visual Studio and .NET 5.0 or higher.
- If you have version 16.8.0 of Visual Studio 2019 installed, then you do not need to install .NET 5.0 or higher separately.
- Microsoft Visual Studio is required to generate a .NET 5.0 assembly using the **Library Compiler** app in MATLAB Compiler SDK.
- Microsoft Visual Studio 2019 is not required for building .NET 5.0 applications. Microsoft .NET 5.0 comes with its own command line tools that let you create, build, and run .NET 5.0 applications.
  - For details on using Visual Studio Code with .NET 5.0, see the Microsoft documentation.
  - For details on using Visual Studio 2019 with .NET 5.0, see the Microsoft documentation.

Python Package Integration: Improved performance with large multidimensional arrays in Python

The Python multidimensional array component used by MATLAB Compiler SDK shows improved performance when:

- Converting data from Python sequences to the data types defined by the `matlab` module
- Transferring data back and forth between Python and MATLAB

In both cases, the improvement is noticeable when operating on arrays with at least 10 elements. When transferring data back and forth, the improvement increases as the size of the array increases.

For example, this Python code measures the execution times of two operations:

1. Converting a Python array of size 10^8 to a MATLAB `double` array
2. Summing the elements of a MATLAB array in a function deployed using MATLAB Compiler SDK

The first operation in bold is about 11x faster than in the previous release, and the second operation in bold is about 260x faster than in the previous release:

```
import random
import time
import matlab

# sumwrap(arr, dim) is a deployed passthrough function to
# the builtin function sum(arr, dim)
import sumwrap

sumwrap_mod_handle = sumwrap.initialize()
rand_array = [random.random() for i in range(10**8)]
s0 = time.perf_counter()
array_md = matlab.double(rand_array, size=(1, 10**8))
s1 = time.perf_counter() - s0
print('conversion to matlab.double(): {} seconds'.format(s1))
s0 = time.perf_counter()
sum_of_elems = sumwrap_mod_handle.sumwrap(array_md, 1)
s1 = time.perf_counter() - s0
print('sum(): {} seconds'.format(s1))
```

The approximate execution times for the first operation are:

- **R2021b:** 41 s
- **R2022a:** 3.7 s

The approximate execution times for the second operation are:

- **R2021b:** 210 s
- **R2022a:** 0.82 s

The code was timed on a Windows 10, Intel® Xeon® CPU E5-1650 v4 @ 3.60 GHz test system by using Python `perf_counter()` statements.

Python Version Support for Python Package Integration

Support	Python Version	Platform
Discontinued	Python 3.7	All platforms

Functionality Being Removed or Changed

-build and -package options in the libraryCompiler function will be removed
Warns

The `-build` and `-package` options in the `libraryCompiler` function will be removed in a future release. To build applications, use one of the `compiler.build` family of functions or the `mcc` command. To package and create an installer, use the `compiler.package.installer` function. There are no changes to the other options of the `libraryCompiler` function.

Also, there are no changes to the **Library Compiler** app. You can continue to the use the app to generate libraries.

-build and -package options in the productionServerCompiler function will be removed
Warns

The `-build` and `-package` options in the `productionServerCompiler` function will be removed in a future release. To generate deployable archives, use the `compiler.build.productionServerArchive` function, or the `mcc` command, or the **Production Server Compiler** app. There are no changes to the other options of the `productionServerCompiler` function.

Also, there are no changes to the **Production Server Compiler** app. You can continue to the use the app to generate deployable archives.

-build and -package options in the deploytool function will be removed
Warns

The `-build` and `-package` options in the `deploytool` function will be removed in a future release. To build applications, use the `mcc` command. To package and create an installer, use the `compiler.package.installer` function. There are no changes to the other options of the `deploytool` function.

R2021b

Packaging: Obfuscate file structures and file names

You can use the `mcc` command with the `-s` option to customize your archive for all deployment targets. This option obfuscates folder structures and file names in the deployable archive (`.ctf` file) from the end user.

Support Packages: Additional options for specifying support packages

You can use two new options to manually include support packages for all deployment targets.

- Use the `mcc` command with the `-Z` option to specify the method of adding support packages to the deployable archive.
- Use a `compiler.build` command with the `SupportPackages` option to specify the method of adding support packages.

To list installed support packages or those used by a specific file, see `compiler.codetools.deployableSupportPackages`.

Excel Add-In for MATLAB Production Server: Create an Excel add-in for MATLAB Production Server using the excelClientForProductionServer function

You can create Excel® add-ins for MATLAB Production Server™ from the MATLAB command prompt using the `compiler.build.excelClientForProductionServer` function. This function, along with the `compiler.build.ExcelClientForProductionServerOptions` function, provides an improved interface to specify various options associated with creating Excel add-ins. You can deploy the generated Excel add-in to MATLAB Production Server. These functions are in addition to the existing Production Server Compiler (MATLAB Production Server) app you use to create Excel add-ins.

C++ Shared Library Integration: Support for strongly typed MATLAB code

When creating C++ shared libraries from MATLAB functions or classes, you can stipulate how MATLAB data types should be represented in C++ application code using standard and custom data type mappings between MATLAB and C++. The data type requirements are specified using an `arguments` block within a MATLAB function or using a `properties` block and `arguments` block within a MATLAB class. The strongly typed MATLAB function, class, or package is compiled using the `mcc` command to generate a C++ shared library header (`.hpp` file) and a deployable archive (`.ctf` file). The generated header file (`.hpp` file) is included in the C++ application code using a `#include` directive. You can then compile and run the application.

For details, see C++ MATLAB Data API Shared Library Support for Strongly Typed MATLAB Code.

Python Version Support for Python Package Integration

Support	Python Version	Platform
Added	Python 3.9	All platforms

Support	Python Version	Platform
Discontinued	Python 3.6, as of R2021a	All platforms

Functionality Being Removed or Changed

-build and -package options in the libraryCompiler function will be removed
Warns

The `-build` and `-package` options in the `libraryCompiler` function will be removed in a future release. To build applications, use one of the `compiler.build` family of functions or the `mcc` command. To package and create an installer, use the `compiler.package.installer` function. There are no changes to the other options of the `libraryCompiler` function.

Also, there are no changes to the **Library Compiler** app. You can continue to the use the app to generate libraries.

-build and -package options in the productionServerCompiler function will be removed
Warns

The `-build` and `-package` options in the `productionServerCompiler` function will be removed in a future release. To generate deployable archives, use the `compiler.build.productionServerArchive` function, or the `mcc` command, or the **Production Server Compiler** app. There are no changes to the other options of the `productionServerCompiler` function.

Also, there are no changes to the **Production Server Compiler** app. You can continue to the use the app to generate deployable archives.

-build and -package options in the deploytool function will be removed
Warns

The `-build` and `-package` options in the `deploytool` function will be removed in a future release. To build applications, use the `mcc` command. To package and create an installer, use the `compiler.package.installer` function. There are no changes to the other options of the `deploytool` function.

R2021a

COM Component Integration: Create a COM component using the compiler.build.comComponent function

You can create COM components from the MATLAB command prompt using the `compiler.build.comComponent` function. This function, along with the `compiler.build.COMComponentOptions` function, provides an improved interface to specify various options associated with creating COM components. You can then integrate the generated COM component into any COM-compliant application. These functions are in addition to the existing **Library Compiler** app you use to create COM components.

C++ Integration: Create a C++ shared library using the compiler.build.cppSharedLibrary function

You can create a C++ shared library from the MATLAB command prompt using the `compiler.build.cppSharedLibrary` function. This function, along with the `compiler.build.CppSharedLibraryOptions` function, provides an improved interface to specify various options associated with creating C++ shared libraries. You can then integrate the generated C++ shared library into any C++ application. These functions are in addition to the existing **Library Compiler** app you use to create shared libraries.

C Integration: Create a C shared library using the compiler.build.cSharedLibrary function

You can create a C shared library from the MATLAB command prompt using the `compiler.build.cSharedLibrary` function. This function, along with the `compiler.build.CSharedLibraryOptions` function, provides an improved interface to specify various options associated with creating C shared libraries. You can then integrate the generated C shared library into any C application. These functions are in addition to the existing **Library Compiler** app you use to create shared libraries.

.NET Integration: Create a .NET assembly using the compiler.build.dotNETAssembly function

You can create a .NET assembly from the MATLAB command prompt using the `compiler.build.dotNETAssembly` function. This function, along with the `compiler.build.DotNETAssemblyOptions` function, provides an improved interface to specify various options associated with creating a .NET assembly. These functions are in addition to the existing **Library Compiler** app you use to create .NET assemblies.

Java: Create a Java package using the compiler.build.javaPackage function

You can create a Java® package from the MATLAB command prompt using the `compiler.build.javaPackage` function. This function, along with the `compiler.build.JavaPackageOptions` function, provides an improved interface to specify various options associated with creating Java packages. You can then integrate the generated Java package into any Java application. These functions are in addition to the existing **Library Compiler** app you use to create Java packages.

Python integration: Create a Python package using the compiler.build.pythonPackage function

You can create a Python package from the MATLAB command prompt using the `compiler.build.pythonPackage` function. This function, along with the `compiler.build.PythonPackageOptions` function, provides an improved interface to specify various options associated with creating Python packages. You can then integrate the generated Python package into any Python application. These functions are in addition to the existing **Library Compiler** app you use to create Python packages.

Functionality Being Removed or Changed

-build and -package options in the libraryCompiler function will be removed
Warns

The `-build` and `-package` options in the `libraryCompiler` function will be removed in a future release. To build applications, use one of the `compiler.build` family of functions or the `mcc` command. To package and create an installer, use the `compiler.package.installer` function. There are no changes to the other options of the `libraryCompiler` function.

Also, there are no changes to the **Library Compiler** app. You can continue to the use the app to generate libraries.

-build and -package options in the productionServerCompiler function will be removed
Warns

The `-build` and `-package` options in the `productionServerCompiler` function will be removed in a future release. To generate deployable archives, use the `compiler.build.productionServerArchive` function, or the `mcc` command, or the **Production Server Compiler** app. There are no changes to the other options of the `productionServerCompiler` function.

Also, there are no changes to the **Production Server Compiler** app. You can continue to the use the app to generate deployable archives.

-build and -package options in the deploytool function will be removed
Warns

The `-build` and `-package` options in the `deploytool` function will be removed in a future release. To build applications, use the `mcc` command. To package and create an installer, use the `compiler.package.installer` function. There are no changes to the other options of the `deploytool` function.

R2020b

Java Package Integration: Support for MATLAB string array

You can now create Java packages from MATLAB code that consists of string arrays.

MATLAB Production Server: Create deployable archives using the compiler.build.productionServerArchive function

You can create deployable archives from the MATLAB command prompt using the `compiler.build.productionServerArchive` function. This function, along with the `compiler.build.ProductionServerArchiveOptions` function, provides an improved interface to specify various options associated with creating deployable archives. You can then copy or upload the deployable archives to the server. These functions are in addition to the existing `mcc` command you use to create deployable archives.

Python Version Support for Python Package Integration

Support	Python Version	Platform
Added	Python 3.8	All platforms

Functionality Being Removed or Changed

-build and -package options in the libraryCompiler function will be removed
Warns

The `-build` and `-package` options in the `libraryCompiler` function will be removed in a future release. To build applications, use the `mcc` command. To package and create an installer, use the `compiler.package.installer` function. There are no changes to the other options of the `libraryCompiler` function.

Also, there are no changes to the **Library Compiler** app. You can continue to the use the app to generate libraries.

-build and -package options in the productionServerCompiler function will be removed
Warns

The `-build` and `-package` options in the `productionServerCompiler` function will be removed in a future release. To generate deployable archives, use the `mcc` command or **Production Server Compiler** app. There are no changes to the other options of the `productionServerCompiler` function.

Also, there are no changes to the **Production Server Compiler** app. You can continue to the use the app to generate deployable archives.

-build and -package options in the deploytool function will be removed
Warns

The `-build` and `-package` options in the `deploytool` function will be removed in a future release. To build applications, use the `mcc` command. To package and create an installer, use the `compiler.package.installer` function. There are no changes to the other options of the `deploytool` function.

R2020a

.NET Assembly Integration: Support for MATLAB string array

You can now create .NET assemblies from MATLAB code that consists of string arrays.

C Shared Library Integration: Support for MATLAB string array

You can now create C shared libraries from MATLAB code that consists of string arrays.

C++ Shared Library Integration: Support for MATLAB string array

You can now create C++ shared libraries from MATLAB code that consists of string arrays.

Functionality Being Removed or Changed

-build and -package options in the libraryCompiler function will be removed
Still runs

The `-build` and `-package` options in the `libraryCompiler` function will be removed in a future release. To build applications, use the `mcc` command. To package and create an installer, use the `compiler.package.installer` function. There are no changes to the other options of the `libraryCompiler` function.

Also, there are no changes to the **Library Compiler** app. You can continue to the use the app to generate libraries.

-build and -package options in the productionServerCompiler function will be removed
Still runs

The `-build` and `-package` options in the `productionServerCompiler` function will be removed in a future release. To generate deployable archives, use the `mcc` command or **Production Server Compiler** app. There are no changes to the other options of the `productionServerCompiler` function.

Also, there are no changes to the **Production Server Compiler** app. You can continue to the use the app to generate deployable archives.

R2019b

.NET Assembly Integration: Run .NET assemblies integrated into .NET applications on Linux and macOS using .NET Core

Previously, MATLAB functions packaged into .NET assemblies and integrated with .NET applications could only run on a Windows platform. Starting in R2019b, you can build these .NET applications on a Windows platform and run them on Linux and macOS using .NET Core. To use this functionality, you must have .NET Core 2.0 or higher installed. For an example, see Create a .NET Core Application That Runs on Linux and macOS.

Compatibility Considerations

- To use this functionality, you must have Visual Studio and .NET Core 2.0 or higher.
- If you have version 15.8.2 of Visual Studio 2017 installed, then you do not need to install .NET Core 2.0 or higher separately.
- Visual Studio 2017 is not required for building .NET Core applications. .NET Core comes with its own command line tools that let you create, build, and run .NET Core applications. For more information, see the .NET Core documentation. However, you will need Visual Studio to generate a .NET assembly using the `Library Compiler` app in MATLAB Compiler SDK.

File Versioning: Generate system-level file versioning on Windows systems for files created using deployment apps

On Windows systems, you can now generate target files with system-level version numbers using deployment apps. System-level versioning of files is supported for the following targets:

- C shared libraries
- C++ shared libraries
- .NET assemblies
- COM components

You can specify the version number in the version section of any of the deployment apps by using the `mcc` command. For more information, see `mcc`.

Compatibility Considerations

System-level file versioning for files created using deployment apps is supported on Windows systems only.

R2019a

Library Compiler: Generate programming language specific sample driver code

When you package MATLAB code into a programming language specific component using the **Library Compiler** app, you can simultaneously generate sample driver code that can be used to build and run that component.

To generate sample driver code:

1. Open the **Library Compiler** app from the **Apps** tab in the toolstrip.
2. Select the type of component you want to generate and add the MATLAB files you want to package.
3. Expand the Samples section in the app and click **Create sample**.
4. In the resulting context menu, select the MATLAB file for which you want to generate sample code. This will automatically create a sample MATLAB script that calls the MATLAB function in the selected file. You can edit this file or leave it as it is.

   Alternatively, if you have an existing MATLAB script that can serve as a sample, add that file instead of creating a sample.

   If you do not create or add a sample MATLAB script, sample driver code will not be generated.
5. Click the **Package** button after creating or adding a sample MATLAB script. The **Library Compiler** app will create the corresponding programming language specific driver code.

   The generated sample driver code can be found in the `.../for_redistribution_files_only/samples` folder. If you install the application, you can find the generated sample code in the `...`*`<installLocation>`*`/`*`<applicationName>`*`/application/samples` folder.

Sample driver code generation is supported for the following targets: C++, Java, Microsoft .NET, and Python.

Sample driver code generation is *not* supported for the following targets: C, COM, and Microsoft Excel.

Functionality Being Removed

Functionality	Result
WebFigures: Support for exporting figures to a deployed Java or .NET application has been removed. The `webfigure` function will no longer work.	Error

Compatibility Considerations

To deploy MATLAB apps to the web, consider using Web Apps (MATLAB Compiler).

R2018b

Version: 6.6

Bug Fixes

R2018a

C++ Shared Library Integration: Generate C++ shared libraries using MATLAB Data API

You can generate C++ shared libraries using the new MATLAB Data API that leverages modern C++ semantics and design patterns. The **Library Compiler** app automatically generates shared libraries using both the old `mwArray` API and the new MATLAB Data API. The new shared library files can be found in a folder named `v2` in the project folder.

For more information, see Integrate a C++ MATLAB Data API Shared Library into an Application.

Compatibility Considerations

Shared libraries created with the `mwArray` API cannot be interchangably used with files generated using the MATLAB Data API.

Compiler Support for C/C++ Shared Library Integration

Support	Compiler	Platform
Added	GNU® gcc 6.x	Linux
Added	Xcode 9.x	macOS

For an up-to-date list of supported compilers, see the Supported and Compatible Compilers website.

Python Version Support for Python Package Integration

Support	Python Version	Platform
Added	Python 3.6, as of R2017b	All platforms
Discontinued	Python 3.4	All platforms

RESTful API for Service Discovery: Discover MATLAB functions deployed on MATLAB Production Server

You can get information about the archives deployed to MATLAB Production Server and the MATLAB functions contained within those deployed archives using the discovery service API. You can use the API to find out the names of the deployed functions and the number and size and of the inputs and outputs.

For more information, see RESTful API.

JSON Encode and Decode Functions: Convert data between MATLAB and JSON for MATLAB Production Server

You can convert data in both directions between MATLAB and JSON, including server requests and responses using MATLAB Production Server JSON schema. You can use these functions when making RESTful API calls to the server. For more information, see the following reference pages:

- mps.json.encode

- mps.json.decode
- mps.json.encoderequest
- mps.json.decoderesponse

R2017b

Functionality Being Removed or Changed

Functionality	Result
WebFigures: Support for exporting figures to a deployed Java or .NET application will be removed in a future release.	Warns

Compiler Support: Create C/C++ shared libraries and generic COM components using the freely available MinGW-w64 compiler on 64-bit Windows

You can use the MinGW-w64 version 5.3.0 compiler with MATLAB Compiler SDK to generate C/C++ shared libraries and generic COM components. The compiler can be installed as an add-on using the Add-On Explorer in the MATLAB desktop. Use the search term "MinGW-w64" in the Add-On Explorer to find and install the compiler.

You will also need to install the Windows 10 SDK to generate generic COM components using the MinGW-w64 compiler. You can download the Windows 10 SDK from Microsoft.

For an up-to-date list of supported compilers, see the Supported and Compatible Compilers website.

Python Package Integration: Support for Python 3.6

You can use Python 3.6 with MATLAB Compiler SDK to generate Python packages that can be integrated with Python applications.

R2017a

Version: 6.3.1

Bug Fixes

R2016b

Neural Network Training Models: Deploy Neural Network Toolbox functions that train a model

Use MATLAB Runtime to deploy functions that can train a model. To deploy MATLAB code that trains neural networks, see Create Standalone Application from Command Line. The following features are not supported in deployed mode:

- Training progress dialog, `nntraintool`
- `genFunction` and `gensim` to generate MATLAB code or Simulink® blocks
- `view` method
- `nctool`, `nftool`, `nnstart`, `nprtool`, `ntstool`
- Plot functions

R2016a

HTTPS Support for Excel Add-ins: Securely integrate Excel Add-ins with MATLAB Production Server using HTTPS

Excel add-ins can securely connect to MATLAB Production Server using HTTPS and evaluate deployed MATLAB functions. Using HTTPS provides transmission-layer encryption. A server certificate must be must be installed when using HTTPS, and can be either self-signed or signed by a trusted Certificate Authority (CA). Self-signed certificates need to be installed in the certificate store of the local machine running the Excel add-in.

For more information, see Change the Server Configuration.

RESTful API and JSON: Develop clients for MATLAB Production Server in any programming language that supports HTTP

The MATLAB Production Server RESTful API enables you to evaluate MATLAB functions on remote servers using JSON representation of MATLAB data types. You can create client programs in any programming language with an HTTP library.

Client code that uses the MATLAB Production Server RESTful API and JSON representation of MATLAB data types can be written in web-based languages such as JavaScript® and embedded in HTML pages. These web pages can then be used to send and retrieve requests from a MATLAB Production Server instance.

For more information, see RESTful API and JSON.

R2015aSP1

Version: 6.0.1

Bug Fixes

R2015b

Python Integration: Deploy MATLAB components as native Python packages, for integration with applications written in Python

MATLAB Compiler SDK packages MATLAB functions into software components that you integrate into native Python applications. MATLAB Compiler SDK provides APIs to execute compiled MATLAB functions on the MATLAB Runtime.

For more information, see Python Package Integration.

R2015a

Packaging of your MATLAB programs as C/C++ shared libraries, Microsoft .NET assemblies, and Java classes

MATLAB Compiler SDK packages MATLAB functions into software components that you integrate into applications. MATLAB Compiler SDK includes APIs to integrate the packaged components using C, C++, Microsoft .NET, or Java.

Royalty-free distribution of software components to users who do not need MATLAB

MATLAB functions compiled with MATLAB Compiler SDK execute on the MATLAB Runtime. The MATLAB Runtime is a freely distributable execution engine made up of shared libraries that MATLAB uses to run MATLAB files on systems without an installed version of MATLAB.

Development and test framework for MATLAB Production Server for integration with web and enterprise systems

MATLAB Compiler SDK includes everything you need to develop client applications for MATLAB Production Server:

- Client APIs execute MATLAB functions on the server.
- Excel add-in to execute MATLAB functions on the server.
- Production Server Compiler app tests the integration between clients and compiles MATLAB functions for deployment to the server.
- Test server tests client code and server configuration together.

Encryption of MATLAB code to protect your intellectual property

MATLAB Compiler SDK encrypts your MATLAB code so that it cannot be read or copied.