AutomationDesk

Accessing MATLAB

For AutomationDesk 6.5

Release 2021-A – May 2021

dSPACE

How to Contact dSPACE

Mail:	dSPACE GmbH Rathenaustraße 26 33102 Paderborn Germany
Tel.:	+49 5251 1638-0
Fax:	+49 5251 16198-0
E-mail:	info@dspace.de
Web:	http://www.dspace.com

How to Contact dSPACE Support

If you encounter a problem when using dSPACE products, contact your local dSPACE representative:

- Local dSPACE companies and distributors: http://www.dspace.com/go/locations
- For countries not listed, contact dSPACE GmbH in Paderborn, Germany.
  Tel.: +49 5251 1638-941 or e-mail: support@dspace.de

You can also use the support request form: http://www.dspace.com/go/supportrequest. If you are logged on to mydSPACE, you are automatically identified and do not need to add your contact details manually.

If possible, always provide the relevant dSPACE License ID or the serial number of the CmContainer in your support request.

Software Updates and Patches

dSPACE strongly recommends that you download and install the most recent patches for your current dSPACE installation. Visit http://www.dspace.com/go/patches for software updates and patches.

Important Notice

Contents

About This Document

Content

This document gives you information on how to access MATLAB via AutomationDesk.

Required knowledge

Working with AutomationDesk requires:

- Basic knowledge in handling the PC and the Microsoft Windows operating system.
- Basic knowledge in developing applications or tests.
- Basic knowledge in handling the external device, which you control remotely via AutomationDesk.

dSPACE provides trainings for AutomationDesk. For more information, refer to https://www.dspace.com/go/trainings.

Symbols

dSPACE user documentation uses the following symbols:

Symbol	Description
⚠ DANGER	Indicates a hazardous situation that, if not avoided, will result in death or serious injury.
⚠ WARNING	Indicates a hazardous situation that, if not avoided, could result in death or serious injury.
⚠ CAUTION	Indicates a hazardous situation that, if not avoided, could result in minor or moderate injury.
NOTICE	Indicates a hazard that, if not avoided, could result in property damage.
Note	Indicates important information that you should take into account to avoid malfunctions.
Tip	Indicates tips that can make your work easier.
⍰	Indicates a link that refers to a definition in the glossary, which you can find at the end of the document unless stated otherwise.

Symbol	Description
📖	Precedes the document title in a link that refers to another document.

Naming conventions

dSPACE user documentation uses the following naming conventions:

%name% Names enclosed in percent signs refer to environment variables for file and path names.

< > Angle brackets contain wildcard characters or placeholders for variable file and path names, etc.

Special folders

Some software products use the following special folders:

Common Program Data folder A standard folder for application-specific configuration data that is used by all users.

`%PROGRAMDATA%\dSPACE\<InstallationGUID>\<ProductName>`

or

`%PROGRAMDATA%\dSPACE\<ProductName>\<VersionNumber>`

Documents folder A standard folder for user-specific documents.

`%USERPROFILE%\Documents\dSPACE\<ProductName>\<VersionNumber>`

Local Program Data folder A standard folder for application-specific configuration data that is used by the current, non-roaming user.

`%USERPROFILE%\AppData\Local\dSPACE\<InstallationGUID>\<ProductName>`

Accessing dSPACE Help and PDF Files

After you install and decrypt dSPACE software, the documentation for the installed products is available in dSPACE Help and as PDF files.

dSPACE Help (local) You can open your local installation of dSPACE Help:

- On its home page via Windows Start Menu
- On specific content using context-sensitive help via **F1**

dSPACE Help (Web) You can access the Web version of dSPACE Help at www.dspace.com/go/help.

To access the Web version, you must have a *mydSPACE* account.

PDF files You can access PDF files via the icon in dSPACE Help. The PDF opens on the first page.

Basics and Instructions

Introduction

AutomationDesk provides the MATLAB Access library to access MATLAB and data stored in MAT files.

Where to go from here

Information in this section

MATLAB Access Basics

Introduction

To use the MATLAB Access library with its sublibraries, you have to know some basics.

Sublibraries

The MATLAB Access library consists of two sublibraries: MATLAB and MATFile.

MATLAB With the automation blocks of the MATLAB sublibrary, you can create a MATLAB instance via AutomationDesk. The sublibrary provides a MATLAB data object to create a MATLAB instance via AutomationDesk. The automation blocks access this data object and can be used for data exchange between MATLAB and AutomationDesk and for executing MATLAB commands via AutomationDesk.

Note

Do not access MATLAB via AutomationDesk and via ControlDesk at the same time. This will cause errors.

MATFile With the MATFile sublibrary and its automation blocks, you can access, read, write, and delete data to or from MAT files. The sublibrary provides a MATFile data object to specify a MAT file. You can create a new MAT file, read data from an existing MAT file, or modify an existing MAT file by using the automation blocks. You do not need a MATLAB instance to work with MAT files.

Supported MATLAB versions

To work with MATLAB via AutomationDesk, you need a correctly installed MATLAB instance. AutomationDesk supports only MATLAB releases compatible with the current dSPACE Release.

MATLAB data object

To create a MATLAB instance, your AutomationDesk project must contain a MATLAB data object. This object handles the MATLAB instance and can be used by the automation blocks of the MATLAB sublibrary. Each automation block must reference a project-specific MATLAB data object. If this data object is created in the Project manager before building a sequence, the automation blocks reference the first MATLAB data object automatically. If you use several MATLAB data objects in your project, you must note AutomationDesk's referencing mechanism. For further information, refer to Scope of Data Object References (AutomationDesk Basic Practices 🕮).

The MATLAB data object provides a data conversion setting for integer data types. The default is no data type conversion.

MATFile data object

To work with a MAT file, your AutomationDesk project must contain a MATFile data object. The automation blocks of the MATFile sublibrary need a MATFile data object. You have to specify the file name of the MAT file you want to work with, the access mode, and the data conversion for integer data types. If you use several MATFile data objects in your project, you must note AutomationDesk's referencing mechanism. For further information, refer to Scope of Data Object References (AutomationDesk Basic Practices 🕮).

Related topics

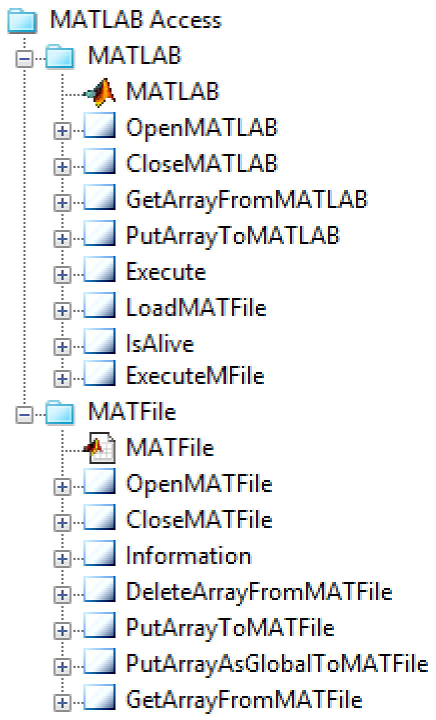
Basics

Scope of Data Object References (AutomationDesk Basic Practices 🕮)

Overview of the MATLAB Access Library Elements

Introduction

The MATLAB Access library contains the two sublibraries MATLAB and MATFile.



MATLAB Access
- MATLAB
  - MATLAB
  - OpenMATLAB
  - CloseMATLAB
  - GetArrayFromMATLAB
  - PutArrayToMATLAB
  - Execute
  - LoadMATFile
  - IsAlive
  - ExecuteMFile
- MATFile
  - MATFile
  - OpenMATFile
  - CloseMATFile
  - Information
  - DeleteArrayFromMATFile
  - PutArrayToMATFile
  - PutArrayAsGlobalToMATFile
  - GetArrayFromMATFile

MATLAB

The MATLAB sublibrary provides functions to open and close MATLAB via AutomationDesk, and to send data to it and retrieve data from it.

MATLAB The MATLAB data object creates the connection to MATLAB. With the ConvertToDouble setting, you can specify how integer values are to be converted when exchanged. For further information, refer to MATLAB on page 35.

OpenMATLAB The OpenMATLAB automation block opens MATLAB. For further information, refer to OpenMATLAB on page 36.

CloseMATLAB The CloseMATLAB automation block closes MATLAB. For further information, refer to CloseMATLAB on page 29.

GetArrayFromMATLAB The GetArrayFromMATLAB automation block reads data from the MATLAB workspace and makes it accessible in AutomationDesk. For further information, refer to GetArrayFromMATLAB on page 32.

PutArrayToMATLAB The PutArrayFromMATLAB automation block writes AutomationDesk data to the MATLAB workspace. For further information, refer to PutArrayToMATLAB on page 37.

Execute With the Execute automation block, you can execute MATLAB commands via AutomationDesk. For further information, refer to Execute on page 30.

LoadMATFile With the LoadMATFIle automation block, you can load a MAT file into MATLAB's workspace via AutomationDesk. For further information, refer to LoadMATFile on page 34.

IsAlive With the IsAlive automation block, you can check whether an access to MATLAB is still possible. For further information, refer to IsAlive on page 33.

ExecuteMFile The ExecuteMFile automation block lets you execute an M file in MATLAB via AutomationDesk. For further information, refer to ExecuteMFile on page 31.

MATFile

With the elements of the MATFile sublibrary, you can read data from and write data to files in a MATLAB format.

MATFile The MATFile data object handles a MAT file. You can specify the file name, the file access mode, and the ConvertToDouble settings, which you can use to specify how integer values are to be converted when exchanged. For further information, refer to MATFile on page 43.

OpenMATFile The OpenMATFile automation block opens a MAT file via AutomationDesk. For further information, refer to OpenMATFile on page 45.

CloseMATFile The CloseMATFile automation block closes a MAT file via AutomationDesk. For further information, refer to CloseMATFile on page 39.

Information The Information automation block displays information about the MAT file contents in AutomationDesk. For further information, refer to Information on page 42.

DeleteArrayFromMATFile The DeleteArrayFromMATFile automation block lets you delete a specified array from a MAT file via AutomationDesk. For further information, refer to DeleteArrayFromMATFile on page 40.

PutArrayToMATFile The PutArrayToMATFile automation block writes data to a MAT file. The data is saved to the local workspace when MATLAB loads the file. For further information, refer to PutArrayToMATFile on page 47.

PutArrayAsGlobalToMATFile The PutArrayAsGlobalToMATFile automation block writes data to the MAT file. The data is saved to the global workspace when MATLAB loads the file. For further information, refer to PutArrayAsGlobalToMATFile on page 46.

GetArrayFromMATFile The GetArrayFromMATFile automation block reads data from a MAT file. For further information, refer to GetArrayFromMATFile on page 41.

Related topics

References

Conversion Rules for Exchanging Data Between AutomationDesk and MATLAB

Introduction

When you use the blocks and data objects of the MATLAB Access library, the data type conversion for AutomationDesk and MATLAB data types is done automatically. For further data processing you need information on the underlying conversion rules.

Conversion rules

MATLAB basically works with array classes. If you want to exchange data between MATLAB and AutomationDesk, these array classes must be mapped to Python data types used in AutomationDesk. Data type conversion is required for reading from a MAT file or the MATLAB workspace, and writing to a MAT file or the MATLAB workspace.

Note

Note the following restrictions:
- AutomationDesk stores the retrieved MATLAB data in a Variant data object, which can handle each MATLAB data type. If the converted Python data type has no equivalent in AutomationDesk, for example, the complex data type, you must use Python functions for postprocessing.
- The MATLAB cell array data type is retrieved in a Variant data object, but it cannot be stored to the AutomationDesk project. If you save the project, the value is reset to None.

Data types

The following table shows the conversion rules for the data types:

MATLAB Data Type	Python Data Type
int8	int
int16	
int32	
uint8	
uint16	
uint32	long[1)]

MATLAB Data Type	Python Data Type
single	float
double	
double (complex)	complex
char	string
	unicode (used by AutomationDesk's String data object)
struct	dictionary
cell	string representation of the cell array (cannot be stored in the AutomationDesk project)

1) If the value of long ≤ 2^31-1, it is converted to int32 MATLAB type.

Note

The data type mapping for the integer types additionally depends on the **Convert to double** setting of the MATLAB and MATFile data object.

Array dimensions

The following table shows the conversion rules according to the array dimensions:

MATLAB Array Dimensions	Python Equivalent
0x0 array	Empty list
1x1 array	Scalar
1xN array	List of related Python type
MxN array	List of list of related Python type
MxNx... array	Nested lists of related Python type

Example

The following table shows some examples of data type mapping:

MATLAB	AutomationDesk
1x1 int8 array	Scalar of int, for example: 1
1x3 double array	List of float, for example: [1.0, 2.0, 3.0]
3x1 int16 array	List of lists of int, for example: [[1],[2],[3]]
2x3 int32 array	List of lists of int, for example:[[1,2,3],[4,5,6]]

Related topics

HowTos

How to Retrieve Data from MATLAB.. 19

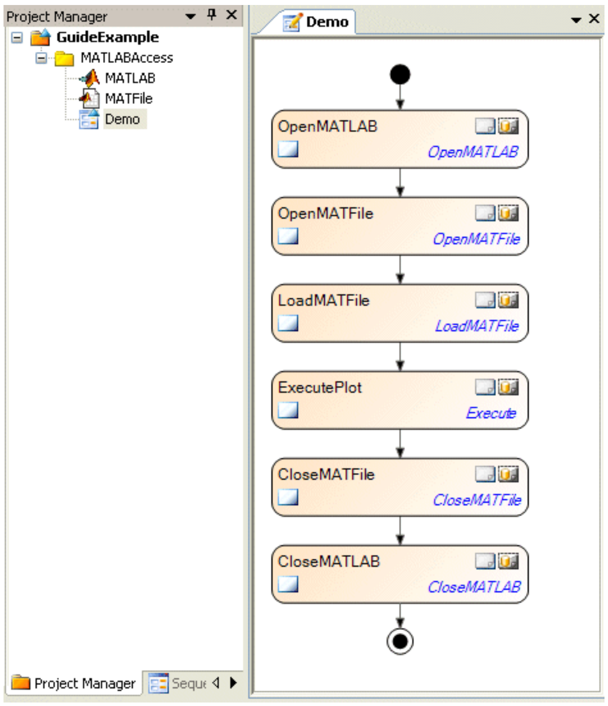
Example of a MATLAB Access Sequence

Introduction

The following automation sequence shows you a simple program for access to MATLAB and for reading and writing data. It automates only:

- Opening MATLAB via AutomationDesk
- Opening a MAT file via AutomationDesk
- Loading the MAT file into MATLAB's workspace
- Executing a MATLAB command
- Closing the MAT file and MATLAB

The illustration below shows what the example sequence looks like:



Preconditions

You have to specify some block-specific values in the Data Object Editor:

- For the OpenMATFile and the LoadMATFile automation blocks, you have to specify the name and location of the MAT file.
- For the Execute automation block, you have to specify the MATLAB command you want to execute, for example, `plot`.

Result

When you execute the sequence, you open MATLAB and a MAT file. The MAT file is loaded into MATLAB's workspace. A MATLAB command is executed and the MAT file and MATLAB are then closed.

Demo projects

Further AutomationDesk demo projects can be found at `<DocumentsFolder>\MATLAB Access Library`.

Related topics

References

How to Work with a MATLAB Instance

Objective

To work with MATLAB, your AutomationDesk project must contain a MATLAB data object representing the MATLAB instance for the automated access.

MATLAB parameters

Communication with a MATLAB instance is specified by:

- Conversion of Python int data types to MATLAB float data types and vice versa. For further information, refer to Conversion Rules for Exchanging Data Between AutomationDesk and MATLAB on page 12

MATLAB referencing

The automation blocks of the MATLAB sublibrary need a MATLAB data object as an input parameter:

- If you have not specified a MATLAB configuration in the Project Manager, the default configuration will be used.
- If you have specified one MATLAB configuration in the Project Manager, this data object is referenced automatically from the MATLAB automation blocks.
- If you have specified more than one MATLAB configuration in the Project Manager, the MATLAB automation blocks use the first configuration data object of their hierarchy level or the explicitly referenced one. This behavior is the same as for any other project-specific data object. For further information, refer to Scope of Data Object References (AutomationDesk Basic Practices 📖).

Preconditions

- To work with MATLAB via AutomationDesk you need an installed MATLAB on your PC. For further information on supported MATLAB versions, refer to MATLAB Access Basics on page 8.
- Make sure that MATLAB runs without errors and is configured according to your needs before you open it as an instance of AutomationDesk.

- You opened AutomationDesk and an AutomationDesk project with an empty sequence beforehand.

Method

To work with a MATLAB instance

1. Drag a MATLAB data object from the Library Browser to the Project Manager. You can place it under the project element or under a folder.
2. Double-click the data object to open the MATLAB Configuration dialog.
3. Specify the conversion mode for integer values.
4. Close the dialog.
5. Drag an OpenMATLAB automation block from the Library Browser to the Sequence Builder. The block's MATLAB data object is automatically referencing the MATLAB data object specified in the Project Manager.
6. Drag a CloseMATLAB automation block from the Library Browser to the Sequence Builder. The block's MATLAB data object is automatically referencing the MATLAB data object specified in the Project Manager.

Result

When you execute the sequence, you have opened and closed a MATLAB instance via AutomationDesk.

Next step

When you have configured your MATLAB instance, you should proceed with How to Send Data to MATLAB on page 16.

Related topics

Basics

Scope of Data Object References (AutomationDesk Basic Practices 🕮)

References

CloseMATLAB	29
MATLAB	35
OpenMATLAB	36

How to Send Data to MATLAB

Objective

If you want MATLAB to do some calculations for your AutomationDesk project, you have to send data to it.

Data conversion

AutomationDesk and MATLAB do not use the same data types. The data has to be converted before being exchanged between them. When you use the blocks of the MATLAB access library, conversion is done automatically. Data type conversion is required for reading from a MAT file or the MATLAB workspace, and writing to a MAT file or the MATLAB workspace. For further information, refer to Conversion Rules for Exchanging Data Between AutomationDesk and MATLAB on page 12.

The following instruction shows, as an example, sending data from a List data object. In general, you can send every data type to MATLAB.

Preconditions

Before you can send data to MATLAB, you have to ensure that you have opened MATLAB via AutomationDesk. Refer to How to Work with a MATLAB Instance on page 15.

Method

To send data to MATLAB

1. Drag a PutArrayToMATLAB automation block from the Library Browser to the sequence.
2. Create a new List data object and place it under the folder in the Project Manager.
3. Rename and edit the List data object.
4. Double-click the List data object to open the Value Editor. Configure the list according to your needs.
5. Specify either the value of the PutArrayToMATLAB block (for example, by referencing to the List data object) or the Reference Name of the PutArrayToMATLAB block.

Result

The data stored in the List data object is sent to MATLAB when you execute the sequence. The block's MATLAB data object is automatically referencing the MATLAB data object specified in the Project Manager.

Next step

When you have sent data to MATLAB you should continue with How to Execute MATLAB Commands on page 18.

Related topics

HowTos

How to Work with a MATLAB Instance 15

References

PutArrayToMATLAB 37

How to Execute MATLAB Commands

Objective

You can execute MATLAB commands and MATLAB M files containing MATLAB functions and scripts via AutomationDesk.

Preconditions

Before you can execute MATLAB commands and M files via AutomationDesk, you have to ensure that you have opened a MATLAB instance via AutomationDesk. Refer to How to Work with a MATLAB Instance on page 15.

Possible Methods

AutomationDesk provides two different blocks for executing MATLAB commands:

- If you want to execute a single MATLAB command, you can use the Execute automation block. The MATLAB command is stored in the automation block. Refer to Method 1 on page 18.
- If you want to execute an M file, you can use the ExecuteMFile automation block. The executed file is not stored by AutomationDesk. Refer to Method 2 on page 18.

Method 1

To execute a MATLAB command

1. Drag an Execute automation block from the Library Browser to the sequence.
2. Enter the MATLAB command in the Value field of the Command data object in the Data Object Editor.

Method 2

To execute an M File

1. Drag an ExecuteMFile automation block from the Library Browser to the sequence.
2. Enter the path where the M file is stored on your computer in the Value field of the MFile data object in the Data Object Editor.

Result

When you execute the sequence, the specified command or M file is sent via AutomationDesk to MATLAB and then the command is executed by MATLAB.

Further steps

When you have executed MATLAB commands via AutomationDesk, you should proceed with How to Retrieve Data from MATLAB on page 19.

Related topics

HowTos

References

How to Retrieve Data from MATLAB

Objective

To get the results of the executed MATLAB commands or M files, you have to receive data from MATLAB in AutomationDesk.

Data exchange between AutomationDesk and MATLAB

The data object used to store the data to be exchanged is a variant. It is mapped automatically according to the conversion rules for data type mapping between AutomationDesk and MATLAB. For further information, refer to Conversion Rules for Exchanging Data Between AutomationDesk and MATLAB on page 12.

You should also use the data type variant, if you create a data object in the Project Manager to be used for data exchange. If you want to process the data in your AutomationDesk sequence later on, you have to specify the right data type as described in the data type mapping.

If you want to retrieve data from MATLAB or read it from a MAT file, ensure that the specified array exists in MATLAB or the MAT file. Otherwise this will cause an error.

Preconditions

Before you can receive data from MATLAB, you have to ensure that:

- For efficient MATLAB access it is recommended to have a MATLAB instance opened before executing this automation block
- MATLAB's workspace contains data which should be sent to AutomationDesk.

Method

To retrieve data from MATLAB

1 Drag a GetArrayFromMATLAB automation block from the Library Browser to the sequence.

2 Create and configure a new Variant data object. Reference it to the Array data object of the GetArrayFromMATLAB automation block.

3 Enter the name of the array which you want to receive from MATLAB in the Value column of the ArrayName data object in the Data Object Editor. The specified name has to be identical to the name of the array in MATLAB.

Result

When you execute the sequence, the specified array is sent from MATLAB to AutomationDesk. The received value is mapped automatically to the corresponding Python data type.

Next Steps

You can now proceed with processing the data in your AutomationDesk sequence.

Related topics

References

How to Work with MATFile Data Objects

Objective

AutomationDesk lets you create, save, and load MAT files to and from MATLAB to exchange data.

MATFile objects

The MATFile data objects can be used to:

- Create, open, and close MAT files
- Load MAT files and assign all their arrays to Python variables
- Write arrays to MAT files
- Read arrays from MAT files
- Delete arrays from MAT files

MATFile parameters

Data exchange with MATLAB via MAT files is specified by:

- Conversion of Python int data types to MATLAB float data types and vice versa. For further information, refer to Conversion Rules for Exchanging Data Between AutomationDesk and MATLAB on page 12.

MATFile referencing

The automation blocks of the MATFile sublibrary need a MATFile data object as an input parameter:

- If you have not specified a MATFile data object in the Project Manager, the default configuration will be used.
- If you have specified a MATFile data object in the Project Manager before creating the sequence, this data object is referenced automatically from the MATLAB automation blocks.

- If you have specified more than one MATFile data object in the Project Manager, the MATFile automation blocks use the first configuration data object of their hierarchy level or the explicitly referenced one. This behavior is the same as for any other project-specific data object. For further information, refer to Scope of Data Object References (AutomationDesk Basic Practices 📖).

Access mode

You can access your MAT file in different modes. You can select one of the following access modes for the MATFile data object in AutomationDesk:

- Access with write protection
- Creating a new file or overwriting an existing file
- Appending data to an existing file
- Compatibility options for several MAT file formats

For further information on the different modes, refer to MATFile on page 43.

Preconditions

The following preconditions must be fulfilled:

- You opened AutomationDesk and an AutomationDesk Project with an empty sequence beforehand.

Method

To work with MATFile data objects

1. Drag a **MATFile** data object from the Library Browser to the Project Manager. You can place it under the project element or under a folder.
2. Double-click the data object to open the **MATFile Configuration** dialog.
3. Specify the file name and the path containing the MAT file to be used, the mode, in which you will work with the MAT file, and the conversion mode of integer data types.
4. Close the dialog.
5. Drag an **OpenMATFile** automation block from the Library Browser to the Sequence Builder. The **MATFile** data object of the **OpenMATFile** block is automatically referenced to the project-specific **MATFile** data object. When the sequence is executed, the specified MAT file is opened or created according to its **Mode** setting. You can now process the data of the MAT file with the **GetArrayFromMATFile** or **PutArrayToMATFile** automation blocks.
6. Drag a **CloseMATFile** automation block from the Library Browser to the Sequence Builder. The MAT file is closed and saved, according to the access mode you specified.

Result

When you execute the sequence, you open a MAT file, and can get information about the MAT file and close it.

Note

If you want detailed information on the variables stored in the MAT file, you can place the **Information** automation block in your sequence.

Next Steps

When you can handle MATFile data objects, you should proceed with How to Write Data to a MAT File on page 22.

Related topics

References

How to Write Data to a MAT File

Objective

If you want to work with test results from AutomationDesk in MATLAB, you can store these results in a MAT file permanently.

Possible methods

AutomationDesk provides two different blocks for writing data to a MAT file:

- If you want MATLAB to load the MAT file contents into its local workspace, use the **PutArrayToMATFile** automation block.
- If you want MATLAB to load the MAT file contents into its global workspace, use the **PutArrayAsGlobalToMATFile** automation block.

Preconditions

The following preconditions must be fulfilled to write data to a MAT file:

- Before you can write data to a MAT file, you have to ensure that you have created a MAT file data object in your AutomationDesk project. Refer to How to Work with MATFile Data Objects on page 20.
- In the access mode of the MATFile data object, you specified whether the data to be written is appended to existing data or the data overwrites existing data. Refer to MATFile on page 43.
- If the access mode is **u - Update**, the specified MAT file must exist.

- Ensure that you work with a sequence that contains an OpenMATFile and a CloseMATFile automation block. You can use the sequence built in How to Work with MATFile Data Objects on page 20 for this, for example.

Method

To write data that is predefined to be loaded to the local workspace of MATLAB

1 Drag a PutArrayToMATFile or a PutArrayAsGlobalToMATFile automation block to the sequence.

2 Specify the value for the ArrayName and the value for the Array in the Data Object Editor.

Result

When you execute the sequence, the specified data is written to the MAT file, and when the MAT file is loaded, the data is stored in the local or global workspace of MATLAB.

Next Steps

When you have written data to a MAT file, you should continue with How to Read Data from a MAT File on page 23.

Related topics

References

How to Read Data from a MAT File

Objective

Before you can work with data from MAT files, the data must be extracted from the MAT files.

Preconditions

Before you can read data from MAT files, you have to ensure that:

- You have created a MATFile data object in your AutomationDesk project.
- The MAT file contains data to be sent to AutomationDesk.
- The sequence contains an OpenMATFile and a CloseMATFile automation block. The access mode to be specified can be one of the read-only access modes.

Method

To read data from a MAT file

1 Drag a GetArrayFromMATFile automation block from the Library Browser to the sequence.

2 Create and configure a new Variant data object. Reference it to the GetArrayFromMATFile automation block.

3 Enter the name of the array to be read from the MAT file in the Value entry of the ArrayName in the Data Object Editor. The specified name has to be identical to the name of the array in the MAT file.

Result

When executing the sequence, AutomationDesk reads the specified data from the MAT file.

Next Steps

If you have read data from MAT files, you should proceed with How to Delete Data from a MAT File on page 24.

Related topics

References

How to Delete Data from a MAT File

Objective

AutomationDesk lets you delete data from MAT files.

Preconditions

Before you can delete data from MAT files, you have to ensure that:

- You have created a MATFile data object in your AutomationDesk project.
- The MAT file contains data to be deleted via AutomationDesk.

Note

If you try to delete data from an empty MAT file, this will cause an error when the sequence is executed.

- The sequence contains an OpenMATFile and a CloseMATFile automation block. The access mode to be specified must be one of the write access modes.

Method

To delete data from a MAT file

1. Drag a **DeleteArrayFromMATFile** automation block from the Library Browser to the sequence.
2. Enter the name of the array to delete from the MAT file in the **Value** entry of the **ArrayName** in the Data Object Editor. The specified name has to be identical to the name of the array in the MAT file.

Result

When the sequence is executed, the specified data is deleted from the MAT file via AutomationDesk.

Related topics

References

Reference Information

Where to go from here

Information in this section

Automation Blocks

Where to go from here

Information in this section

MATLAB

Introduction

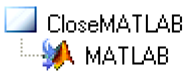
AutomationDesk provides data objects and automation blocks to access MATLAB data via MATLAB instance.

Where to go from here

Information in this section

CloseMATLAB

Graphical representation



Purpose

To close a MATLAB instance that was opened in an automation sequence.

Description

The CloseMATLAB automation block is used to close a MATLAB instance that was created beforehand in either the same or another sequence. For example, if you start an execution containing several sequences with MATLAB access blocks, you can start MATLAB in the first sequence using the OpenMATLAB block, and stop MATLAB in the last sequence using the CloseMATLAB block. In this way, you can save the time for closing and reloading MATLAB in the other sequences.

Note

- With this block, you cannot close a MATLAB instance, that you interactively created using the Open MATLAB command.
- If MATLAB is not already running, this block opens MATLAB before closing it.

Data objects

This automation block provides the following data object:

Name	In / Out	Data Type	Default Value	Description
MATLAB	In	MATLAB	None	Contains the instantiated MATLAB data object.

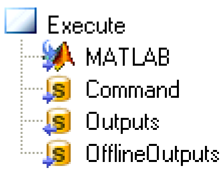
Related topics

Basics

References

Execute

Graphical representation



Purpose

To execute a MATLAB command.

Description

The Execute automation block is used to send a MATLAB command to the MATLAB instance. If it is a valid command, it is executed, otherwise an error message is displayed in the Message Viewer.

Because "Execute" is used as an internal identifier, an instance of this block is automatically named "Execute1".

Note

If MATLAB is not already running, this block opens MATLAB. It is automatically closed when the execution has finished.

Tip

- If you want to execute M files containing MATLAB scripts and functions, you can use the ExecuteMFile automation block.
- If you want to load a MAT file, you can use the LoadMATFile automation block.

Data objects

This automation block provides the following data objects:

Name	In / Out	Data Type	Default Value	Description
MATLAB	In	MATLAB	None	Contains the instantiated MATLAB data object.
Command	In	String	" "	Specifies the command to be executed in MATLAB.
Outputs	Out	String	" "	Returns the outputs of the executed command.
OfflineOutputs	In	String	" "	Lets you specify the string to be used in offline operation mode.

Related topics

Basics

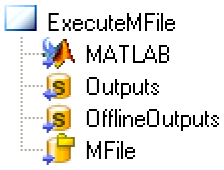
Basics and Instructions... 7
Executing Sequences Using Different Operation Modes (AutomationDesk Basic Practices 🕮)

References

ExecuteMFile.. 31
LoadMATFile.. 34
MATLAB... 35

ExecuteMFile

Graphical representation



Purpose

To execute an M file.

Description

The ExecuteMFile automation block is used to execute MATLAB functions and scripts saved to an M file.

Note

If MATLAB is not already running, this block opens MATLAB. It is automatically closed when the execution has finished.

Data objects

This automation block provides the following data objects:

Name	In / Out	Data Type	Default Value	Description
MATLAB	In	MATLAB	None	Contains the instantiated MATLAB data object.
MFile	In	File	" "	Specifies the M file to be executed in MATLAB.
Outputs	Out	String	" "	Returns the outputs of the executed M file.
OfflineOutputs	In	String	" "	Lets you specify the string to be used in offline operation mode.

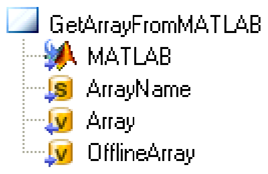
Related topics

Basics

References

GetArrayFromMATLAB

Graphical representation



Purpose

To get an array from the MATLAB workspace.

Description

The GetArrayFromMATLAB automation block is used to retrieve data from MATLAB. The data is specified by its name. The name must correspond to an existing MATLAB workspace variable, otherwise this block returns an error message. The values are stored in a Variant data object.

For further information on data type mapping, refer to Conversion Rules for Exchanging Data Between AutomationDesk and MATLAB on page 12.

Note

If MATLAB is not already running, this block opens MATLAB. It is automatically closed when the execution has finished.

Data objects

This automation block provides the following data objects:

Name	In / Out	Data Type	Default Value	Description
MATLAB	In	MATLAB	None	Contains the instantiated MATLAB data object.
ArrayName	In	String	" "	Specifies the name of the workspace variable you want to retrieve.
Array	Out	Variant	None	Returns the data of the corresponding MATLAB variable.
OfflineArray	In	Variant	None	Lets you specify the array to be used in offline operation mode.

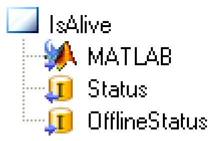
Related topics

Basics

References

IsAlive

Graphical representation



Purpose

To check whether a MATLAB instance is alive.

Description

The IsAlive automation block is used to determine the status of the specified MATLAB instance. It checks whether MATLAB reacts to an access from AutomationDesk.

> **Note**
>
> If MATLAB is not already running, this block opens MATLAB. It is automatically closed when the execution has finished.

Data objects

This automation block provides the following data objects:

Name	In / Out	Data Type	Default Value	Description
MATLAB	In	MATLAB	None	Contains the instantiated MATLAB data object.
Status	Out	Int	0	Returns the alive state of MATLAB: ▪ 0 - MATLAB is not alive ▪ 1 - MATLAB is alive
OfflineStatus	In	Int	0	Lets you specify the value to be used in offline operation mode.

Related topics

Basics

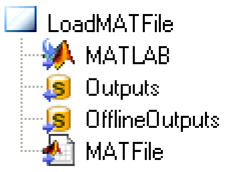
Basics and Instructions.......... 7
Executing Sequences Using Different Operation Modes (AutomationDesk Basic Practices 🕮)

References

MATLAB.......... 35

LoadMATFile

Graphical representation


LoadMATFile
MATLAB
Outputs
OfflineOutputs
MATFile

Purpose

To load a MAT file to the MATLAB workspace.

Description

The LoadMATFile automation block is used to load the specified MAT file to the MATLAB workspace. Its content is accessible via the workspace variables.

You can also use the Execute block to load a MAT file, but then it is more complicated to specify dynamically the file to be loaded.

Note

If MATLAB is not already running, this block opens MATLAB. It is automatically closed when the execution has finished.

Data objects

This automation block provides the following data objects:

Name	In / Out	Data Type	Default Value	Description
MATLAB	In	MATLAB	None	Contains the instantiated MATLAB data object.
Outputs	Out	String	" "	Returns the outputs of the MATLAB load command.
OfflineOutputs	In	String	" "	Lets you specify the string to be used in offline operation mode.
MATFile	In	MATFile	None	Specifies the MAT file to be loaded to the MATLAB workspace.

Related topics

Basics

Basics and Instructions.......... 7
Executing Sequences Using Different Operation Modes (AutomationDesk Basic Practices 🕮)

References

MATFile.......... 43
MATLAB.......... 35

MATLAB

Graphical representation

MATLAB

Purpose

To specify a MATLAB instance.

Description

The MATLAB data object is used to handle the MATLAB instance that you work with. If you have created a MATLAB data object in the Project Manager, this data object represents a MATLAB instance that can be used by the automation blocks of the MATLAB Access library. You can also open and close this MATLAB instance interactively by using the **Open MATLAB** and **Close MATLAB** commands from the data object's context menu.

Each automation block in the MATLAB folder of the MATLAB Access library must reference a project-specific MATLAB data object in the Project Manager. If you create the MATLAB data object in the Project Manager before you start building your automation sequence, the automation blocks automatically reference the MATLAB data object.

The MATLAB data object provides the **Convert to double** setting, which you can set using the edit command. By default, no data type conversion for integer types is done.

Note

You can only use one MATLAB instance at the same time.
If you created a second MATLAB data object, the blocks use the workspace from the already opened MATLAB instance anyway.

Related topics

Basics

References

OpenMATLAB

Graphical representation

Purpose

To open MATLAB.

Description

The OpenMATLAB automation block is used to prepare the connection to MATLAB. If there is no MATLAB instance already instantiated by AutomationDesk, it opens a new one, otherwise it accesses the MATLAB instance already running.

Data objects

This automation block provides the following data object:

Name	In / Out	Data Type	Default Value	Description
MATLAB	In	MATLAB	None	Contains the instantiated MATLAB data object.

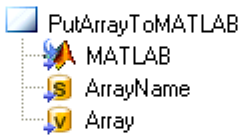
Related topics

Basics

References

PutArrayToMATLAB

Graphical representation



Purpose

To put an array in the MATLAB workspace.

Description

The PutArrayToMATLAB automation block is used to send data from AutomationDesk to MATLAB. The data is specified by a name and its values. A new variable is created in the MATLAB workspace according to these inputs. If the **Convert to double** setting of the MATLAB data object is specified as `TRUE`, values of integer Python types, like int and long, are converted to the double MATLAB type.

For further information on data type mapping, refer to Conversion Rules for Exchanging Data Between AutomationDesk and MATLAB on page 12.

Note

If MATLAB is not already running, this block opens MATLAB. It is automatically closed when the execution has finished.

Data objects

This automation block provides the following data objects:

Name	In / Out	Data Type	Default Value	Description
MATLAB	In	MATLAB	None	Contains the instantiated MATLAB data object.
ArrayName	In	String	" "	Specifies the name of the MATLAB workspace variable to be created.
Array	In	Variant	None	Specifies the data that you want to send to MATLAB.

Related topics

Basics

References

MATFile

Introduction

AutomationDesk provides data objects and automation blocks to access MAT files.

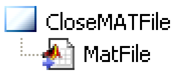
Where to go from here

Information in this section

Automation blocks

CloseMATFile

Graphical representation



Purpose

To close an opened MAT file.

Description

The CloseMATFile automation block is used to close a MAT file that you opened before using the OpenMATFile automation block. Whether the file is saved, and how, depends on the specified file access mode of the referenced MATFile data object. For further information, refer to MATFile on page 43.

Data objects

This automation block provides the following data object:

Name	In / Out	Data Type	Default Value	Description
MATFile	In	MATFile	None	Contains the instantiated MATFile data object.

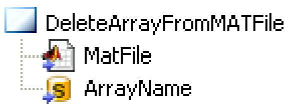
Related topics

Basics

References

DeleteArrayFromMATFile

Graphical representation



Purpose

To delete an array from a MAT file.

Description

The DeleteArrayFromMATFile automation block is used to delete the specified array from a MAT file.

Data objects

This automation block provides the following data objects:

Name	In / Out	Data Type	Default Value	Description
MATFile	In	MATFile	None	Contains the instantiated MATFile data object.
ArrayName	In	String	" "	Name of the array to be deleted.

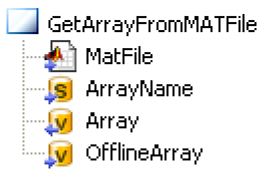
Related topics

Basics

References

GetArrayFromMATFile

Graphical representation



Purpose

To get an array from a MAT file.

Description

The GetArrayFromMATFile automation block is used to read the array, which is specified by its name, from the MAT file. The Python representations of the values are stored in the block's Array data object.

For further information on data type mapping, refer to Conversion Rules for Exchanging Data Between AutomationDesk and MATLAB on page 12.

Data objects

This automation block provides the following data objects:

Name	In / Out	Data Type	Default Value	Description
MATFile	In	MATFile	None	Contains the instantiated MATFile data object.
ArrayName	In	String	" "	Specifies the name of the array to be read.
Array	Out	Variant	None	Returns the data of the specified array.
OfflineArray	In	Variant	None	Lets you specify the array to be used in offline operation mode.

Related topics

Basics

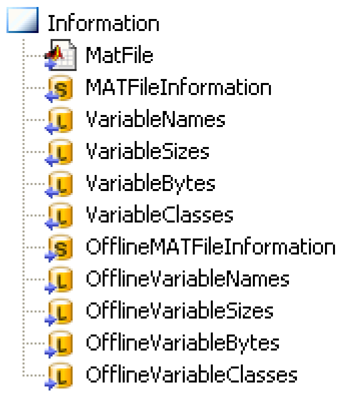
Basics and Instructions.. 7
Executing Sequences Using Different Operation Modes (AutomationDesk Basic Practices 🕮)

References

MATFile.. 43
PutArrayToMATFile.. 47

Information

Graphical representation



Purpose

To get information about the variables stored in a MAT file.

Description

The Information automation block is used to get detailed information on the variables stored in the specified MAT file. It corresponds to the `whos` command, but additionally, the automation block provides separate data objects for the names, sizes, bytes, and classes of the variables. If the MAT file is empty, a message is displayed.

It is not required to execute an OpenMATFile block before executing an Information block.

Data objects

This automation block provides the following data objects:

Name	In / Out	Data Type	Default Value	Description
MATFile	In	MATFile	None	Contains the instantiated MATFile data object.
MATFileInformation	Out	String	" "	Returns the entire information on the file as a single string.
VariableNames	Out	List	[ ]	Returns the names of the variables as a list of strings.
VariableSizes	Out	List	[ ]	Returns the sizes of the variables as a list of strings.
VariableBytes	Out	List	[ ]	Returns the bytes of the variables as a list of strings.
VariableClasses	Out	List	[ ]	Returns the classes of the variables as a list of strings.
OfflineMATFileInformation	In	String	" "	Lets you specify the string to be used in offline operation mode.

Name	In / Out	Data Type	Default Value	Description
OfflineVariableNames	In	List	[ ]	Lets you specify the strings to be used in offline operation mode.
OfflineVariableSizes	In	List	[ ]	Lets you specify the strings to be used in offline operation mode.
OfflineVariableBytes	In	List	[ ]	Lets you specify the strings to be used in offline operation mode.
OfflineVariableClasses	In	List	[ ]	Lets you specify the strings to be used in offline operation mode.

Related topics

Basics

Basics and Instructions..7
Executing Sequences Using Different Operation Modes (AutomationDesk Basic Practices)

References

MATFile..43

MATFile

Graphical representation

MATFile

Purpose

To specify a MAT file.

Description

The MATFile data object is used to create and configure a project-specific MATFile data object in the Project Manager. This can be referenced by a block's MATFile data object.

You can specify a **MATFile** data object in two different ways:

- By specifying a file name and its absolute or relative path.
- By specifying the name of a file that you added before to the project's or custom library's attachment folder. Refer to the **`Manage Attachments`** command.

  Files in the attachment folder of custom libraries are specified in the following format:

  ```
  <CustomLibraryName>::<FileName>
  ```

For both ways of specifying a **MATFile** data object, you can use its edit command. Refer to Edit (MATFile) on page 49.

The MATFile data object provides the following settings which you can specify by using its edit command.

FileName Lets you specify the path and name of the file you want to work with. You can choose between an absolute or relative path to be used.

Absolute path Lets you specify the path as an absolute path in the project.

Relative path Lets you specify the path as a relative path in the project. It is then a shortened path relating to the AutomationDesk project file. The path will not be changed to a relative path if the project and the specified file are saved to different drives.

Additionally, you can specify the following:

Mode Lets you specify the file access mode:

Value	Meaning
r (default)	Read The opened file can be read but not modified. The version of the MAT file is determined and will be preserved.
u	Update (read and write) The file to be updated must exist. New input is appended to the existing content. The version of the MAT file is determined and will be preserved.
w	Write If the file does not exist, it will be created. Existing contents are deleted. The HDF5-based file format can be read with MATLAB version 7.3 and later.
w4	Write Level 4 MAT file If the file does not exist, a MAT file is created that can be read with MATLAB version 4 and earlier. Existing contents are deleted.
wL	Write character data using the default character set for your system If the file does not exist, a MAT file is created that can be read with MATLAB version 6 or 6.5. Existing contents are deleted.
wz	Write compressed data If the file does not exist, a MAT file is created that can be read with MATLAB version 7 and later. Existing contents are deleted.

Convert to double Lets you select the type conversion behavior. The following values are provided:

Value	Meaning
True	If you use a PutArray block for integer Python types like int and long, the values are converted to the double MATLAB type.
False (default)	No data type conversion is done.

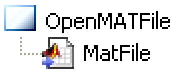
Related topics

Basics

References

OpenMATFile

Graphical representation



Purpose

To open a MAT file.

Description

The OpenMATFile automation block is used to open the specified MAT file. The file access behavior depends on the Mode and Convert to double settings. When you add this block to your sequence, if you created a MATFile data object in the Project Manager beforehand, the block's MATFile data object is automatically referenced to the MATFile data object in the Project Manager.

Data objects

This automation block provides the following data object:

Name	In / Out	Data Type	Default Value	Description
MATFile	In	MATFile	None	Contains the instantiated MATFile data object.

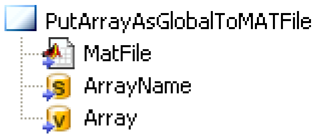
Related topics

Basics

References

PutArrayAsGlobalToMATFile

Graphical representation



Purpose

To write an array with global variables to the MAT file.

Description

The PutArrayAsGlobalToMATFile automation block is used to write data from AutomationDesk to the specified MAT file. If you load this data to MATLAB, it is stored in the global workspace instead of the local workspace.

By setting the access mode of the MATFile data object, you can specify whether the data to be written is appended to existing data or the data overwrites existing data. Refer to MATFile on page 43.

For further information on data type mapping, refer to Conversion Rules for Exchanging Data Between AutomationDesk and MATLAB on page 12.

Data objects

This automation block provides the following data objects:

Name	In / Out	Data Type	Default Value	Description
MATFile	In	MATFile	None	Contains the instantiated MATFile data object.
ArrayName	In	String	" "	Specifies the name of the variable to be written to the MAT file.
Array	In	Variant	None	Specifies the data to be written to the MAT file.

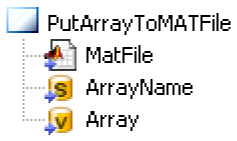
Related topics

Basics

References

PutArrayToMATFile

Graphical representation



Purpose

To write an array to a MAT file.

Description

The PutArrayToMATFile automation block is used to write data to the specified MAT file.

By setting the access mode of the MATFile data object, you can specify whether the data to be written is appended to existing data or the data overwrites existing data. Refer to MATFile on page 43.

If the **Convert to double** setting of the MATFile data object is specified as `TRUE`, values of integer Python types, like int and long, are converted to the double MATLAB type.

For further information on data type mapping, refer to Conversion Rules for Exchanging Data Between AutomationDesk and MATLAB on page 12.

Data objects

This automation block provides the following data objects:

Name	In / Out	Data Type	Default Value	Description
MATFile	In	MATFile	None	Contains the instantiated MATFile data object.
ArrayName	In	String	" "	Specifies the name of the variable to be written to the MAT file.
Array	In	Variant	None	Specifies the data to be written to the MAT file.

Related topics

Basics

References

Commands And Dialogs

Where to go from here

Information in this section

Close MATLAB

Access

You can access this command via:

Ribbon	None
Context menu of	MATLAB data object in the Project Manager, Sequence Hierarchy Browser, and Data Object Editor
Shortcut key	None
Icon	None

Purpose

To close an opened MATLAB instance interactively.

Description

With this command, you can close the MATLAB instance that you opened before using the **Open MATLAB** command. A MATLAB instance that was opened by an automation sequence cannot be closed by this command.

Note

A MATLAB instance that you opened in AutomationDesk (interactively or by automation) must be closed before you exit your AutomationDesk session.

Related topics

References

Edit (MATFile)

Access

You can access this command via:

Ribbon	None
Context menu of	MATFile data object in the Project Manager, Sequence Hierarchy Browser, and Data Object Editor
Shortcut key	None
Icon	None
Others	▪ Double-click MATFile data object ▪ Press `Enter` with MATFile data object selected

Purpose

To configure the MATFile data object.

Description

With this command, you can configure the name and path of the MAT file, the access mode, and the data type conversion for data exchange between AutomationDesk (using Python data types) and MATLAB.

Dialog settings

The edit command of the MATFile data object opens the MATFile Configuration dialog.

File/Attachment These two options let you select whether to specify the file via its name and path or via its name in the attachment folder.

If the File option is selected:

File name You can use the Browse button to open a standard Open or Save As dialog for specifying the MAT file you want to open or create. The dialog type depends on the selected Mode parameter.

Absolute path Lets you specify the path as an absolute path in the project.

Relative path Lets you specify the path as a relative path in the project. It is then a shortened path relating to the AutomationDesk project file. The path will not be changed to a relative path if the project and the specified file are saved to different drives.

If the Attachment option is selected:

File name Lets you specify the name of a file in the project's or custom library's attachment folder. Files in custom libraries are specified in the following format:

```
<CustomLibraryName>::<FileName>
```

Alternatively, you can select the file name from the list of files that are already attached to the current project or to one of the open custom libraries.

Add Opens a file browser that lets you specify a file to be added to the project's or library's attachment folder. The name of this file is automatically selected in the File name input field.

Independent from the selected File/Attachment option, you can specify the following:

Mode Lets you select the file access mode. The following values are provided:

Value	Meaning
r (default)	Read The opened file can be read but not modified. The version of the MAT file is determined and will be preserved.
u	Update (read and write) The file to be updated must exist. New input is appended to the existing content. The version of the MAT file is determined and will be preserved.
w	Write If the file does not exist, it will be created. Existing contents are deleted. The HDF5-based file format can be read with MATLAB version 7.3 and later.
w4	Write Level 4 MAT file If the file does not exist, a MAT file is created that can be read with MATLAB version 4 and earlier. Existing contents are deleted.
wL	Write character data using the default character set for your system If the file does not exist, a MAT file is created that can be read with MATLAB version 6 or 6.5. Existing contents are deleted.
wz	Write compressed data If the file does not exist, a MAT file is created that can be read with MATLAB version 7 and later. Existing contents are deleted.

Convert to double Lets you select the type conversion behavior. The following values are provided:

Value	Meaning
True	If you use a PutArray block for integer Python types like int and long, the values are converted to the double MATLAB type.
False (default)	No data type conversion is done.

Related topics

References

Edit (MATLAB)

Access

You can access this command via:

Ribbon	None
Context menu of	MATLAB data object in the Project Manager, Sequence Hierarchy Browser, and Data Object Editor
Shortcut key	None
Icon	📝
Others	▪ Double-click MATLAB data object ▪ Press **Enter** with MATLAB data object selected

Purpose

To configure the MATLAB data object.

Description

With this command, you can configure the data type conversion for data exchange between AutomationDesk (using Python data types) and MATLAB.

Dialog settings

The edit command of the MATLAB data object opens the MATLAB Configuration dialog.

Convert to double Lets you select the type conversion behavior. The following values are provided:

Value	Meaning
True	If you use a PutArray block for integer Python types like int and long, the values are converted to the double MATLAB type.
False (default)	No data type conversion is done.

Related topics

References

Insert (MATLAB Access Elements)

Access

You can access the data objects of the MATLAB Access library via:

Ribbon	Home - Insert - Data Objects
Context menu of	None
Shortcut key	None
Icon	▪ MATLAB ▪ MATFile

Purpose

To insert data objects of the MATLAB Access library using the Home ribbon.

Description

The ribbon commands are enabled or disabled according to context.

You can add data objects to your project in the Project Manager and to specific automation blocks in the Sequence Builder, such as Exec automation blocks. The data object is added to the selected element.

Related topics

References

Open MATLAB

Access

You can access this command via:

Ribbon	None
Context menu of	MATLAB data object in the Project Manager, Sequence Hierarchy Browser, and Data Object Editor
Shortcut key	None
Icon	None

Purpose

To open MATLAB interactively.

Description

With this command, you can open the MATLAB instance that AutomationDesk will start using the specified MATLAB data object. To save execution time, you can start MATLAB only once before executing several sequences that contain automated access to MATLAB.

Note

- Each MATLAB data object in the Project Manager uses the same MATLAB instance. Only one MATLAB instance can be opened at the same time.
- A MATLAB instance that you have opened interactively must also be closed interactively using the Close MATLAB command.

Related topics

References

Automation

Basics on Automating the Access to MATLAB

Introduction

AutomationDesk provides a COM-based API to automate the handling of AutomationDesk.

Related information

The AutomationDesk COM API provides the following objects for configuring the access to MATLAB:

- MATLAB (AutomationDesk Automation 🕮)
- MATFile (AutomationDesk Automation 🕮)

For basic information and instructions, refer to Basics and Instructions (AutomationDesk Automation 🕮).