AutomationDesk

Automation

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

Introduction

AutomationDesk provides a COM-based application programming interface (API) with which you can create a COM server that accesses either AutomationDesk with its standard user interface (UI), or the UI-free AutomationDesk - Automation Server. This reference describes the objects, properties, methods, and events accessible via the AutomationDesk API.

The syntax descriptions and example codes in this reference are implemented in Python, but the AutomationDesk API can be used in any programming language.

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.
📖	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.

New Features

Introduction	Information on enhancements and new features of AutomationDesk.

New Features For the Automation of AutomationDesk

New features and migration	For information on new features of the current version of AutomationDesk, refer to New Features of AutomationDesk 6.5 (New Features and Migration 🕮).

Basics and Instructions

Where to go from here

Information in this section

General Information on the AutomationDesk API

Where to go from here

Information in this section

Overview of the AutomationDesk API

Architecture

The AutomationDesk API is implemented as a COM object model. The Microsoft Component Object Model (COM) is based on the client-server principle. It supports communication between objects from different applications. For further information, refer to Overview of the AutomationDesk API on page 32.

AutomationDesk and Automation Server

You can develop client applications which access AutomationDesk. For further information, refer to General Information on the AutomationDesk API on page 31.

Programming language

The COM objects can be used by any COM-compatible applications, regardless of the programming language in which they were developed. The client application can be implemented in a programming language such as C#, Python, Visual Basic, or any other COM-compatible programming language. The syntax descriptions and the examples in this document are written in Python. You can immediately work with the Python source codes provided, because the dSPACE installation includes the Python interpreter and several Python modules. To develop Python scripts, you can use PythonWin, which you can find in the Windows Start menu (Start - Programs - dSPACE Python 3.6 - PythonWin).

As of dSPACE Release 2018-B, Python 3.6 is supported. For information on changes and the migration of Python scripts in dSPACE products, refer to the dSPACE website: http://www.dspace.com/go/Python36Migration.

For more information on the main differences between using the AutomationDesk COM API from Python or from other languages, refer to Translating Code Into Other Programming Languages on page 78.

Demo applications

For some demo applications, refer to `<DocumentsFolder>\API`.

For detailed instructions how to use the API, refer to Programming Instructions on page 35.

Related topics

Basics

Overview of the Data Types Used

Information on the data types

The following data types are used by the API:

Data Type	Description
boolean	False (=0) or True (≠0)[1)]
int	16-bit signed integer
long	32-bit signed integer
double	64-bit floating-point number
string	Arbitrary text in unicode characters
date	Date and time in the format: YYYY/MM/DD, HH-MM-SS, for example, 2006/10/11, 13-35-45.
variant	A parameter of this type can be used for any data types.

[1)] The COM interface interprets `True` as `-1`.

Note

The integer value used with the COM API is restricted to 32 bits (long data type). In AutomationDesk, an Int data object is represented by a Python integer with unlimited precision.

Overview of API Constants

Information on the API constants

The following constants are provided by the API:

Alignment Enumeration	
Member	**Description**

adLeft = 0
adCenter = 1
adRight = 2
	Specifies the horizontal alignment of the logo in a report. The default setting depends on the registry entry.

The element type of an object is specified by the ElementType enumeration. You can get an object's element type by using its Type property.

ElementType Enumeration	
Member	**Description**

adProject = 0
adFolder = 1
adSequence = 2
adResult = 7
adReport = 8
adLibraryFolder = 9
adCustomLibraryFolder = 10
	Enumerations used for elements in the Standard library.

adMainLibraryInt = 3
adMainLibraryFloat = 4
adMainLibraryString = 5
adMainLibraryFile = 6
adMainLibraryCondition = 11
adMainLibraryVariant = 12
adMainLibraryDataContainer = 13
adMainLibraryList = 14
adMainLibraryTuple = 15
adMainLibraryDictionary = 16
adMainLibraryVerdict = 56
	Enumerations used for elements in the Main Library.

adReportLibraryColor = 17	Enumerations used for elements in the Report library.

ElementType Enumeration	
Member	**Description**

adRemoteCalibrationCOMSystem = 22
adRemoteCalibrationCOMProject = 23
adRemoteCalibrationCOMLogicalLink = 24
adRemoteCalibrationCOMCharacteristic = 25
adRemoteCalibrationCOMCollector = 26
adRemoteCalibrationCOMMeasurement = 27
adRemoteCalibrationCOMCollectors = 48
adRemoteCalibrationCOMCharacteristics = 49
	Enumerations used for elements in the Remote Calibration (COM) library.

adRS232Configuration = 28	Enumerations used for elements in the RS232 library.

adRemoteDiagnosticsCOMSystem = 29
adRemoteDiagnosticsCOMProject = 30
adRemoteDiagnosticsCOMVehicleInformation = 31
adRemoteDiagnosticsCOMLogicalLink = 32
adRemoteDiagnosticsCOMControlPrimitive = 33
adRemoteDiagnosticsCOMService = 34
adRemoteDiagnosticsCOMSingleJob = 35
adRemoteDiagnosticsCOMResults = 36
adRemoteDiagnosticsCOMControlPrimitives = 50
adRemoteDiagnosticsCOMServices = 51
adRemoteDiagnosticsCOMSingleJobs = 52
	Enumerations used for elements in the Remote Diagnostics (COM) library.

adMatlab = 44
adMatFile = 45
	Enumerations used for elements in the MATLAB Access library.

adSignal = 47	Enumerations used for elements in the Evaluation library.
adDeprecated = 55	Enumeration used for elements of discontinued libraries.

ElementType Enumeration	
Member	**Description**

adXilApiBaseValue = 58
adXilApiCapture = 59
adXilApiCaptureResult = 60
adXilApiCaptureResultReader = 61
adXilApiCaptureResultWriter = 62
adXilApiCaptureState = 63
adXilApiDataType = 64
adXilApiMaPort = 65
adXilApiMaPortConfiguration = 66
adXilApiSignalDescription = 67
adXilApiSignalDescriptionSet = 68
adXilApiSignalDescriptionsReader = 69
adXilApiSignalDescriptionsWriter = 70
adXilApiSignalGenerator = 71
adXilApiSignalGeneratorReader = 72
adXilApiSignalGeneratorWriter = 73
adXilApiSignalGroupValue = 74
adXilApiSignalSegment = 75
adXilApiSignalValue = 76
adXilApiSymbol = 77
adXilApiWatcher = 78
adXilApiMapping = 79
adXilApiTestbenchFactory = 80
adXilApiTestbench = 81
adXilApiPortConfig = 82
adXilApiTaskInfo = 83
adXilApiVariableInfo = 84
adXilApiSymbolFactory = 85
adXilApiCapturingFactory = 86
adXilApiSignalGeneratorFactory = 87
adXilApiSignalFactory = 88
adXilApiValueFactory = 89
adXilApiAttributes = 90
adXilApiWatcherFactory = 91
adXilApiMAPortFactory = 92
adXilApiEESPort = 93
	Enumerations used for elements in the XIL API library.

ElementType Enumeration	
Member	**Description**

adXilApiEESPortFactory = 94
adXilApiErrorConfiguration = 95
adXilApiEESConfigurationReader = 96
adXilApiEESConfigurationWriter = 97
adXilApiErrorSet = 98
adXilApiBaseErrorBuilder = 99
adXilApiErrorFactory = 100
adXilApiBaseError = 101
adXilApiSpecificErrorFactory = 102
adXilApiSpecificErrorFactory2 = 103
adMainLibraryLabeledValue = 104
adMainLibraryBool = 105	

adPythonModule = 106
adPythonPackage = 107
	Enumerations used for elements in a custom library.

FileFormats Enumeration	
Member	**Description**

adZip = 0
adXML = 1
	If you import or export a project or a custom library, you can use the constants of the FileFormats to specify whether to use a ZIP archive or XML files.

FileOptions Enumeration	
Member	**Description**

adCancel = 0
adOverWrite = 1
	If you create, import, export, or rename a project or a custom library, you can use the constants of the FileOptions to specify whether an existing project is overwritten or the instruction is canceled.

InOutStateConstant Enumeration	
Member	**Description**

adInDOB = 0
adOutDOB = 1
adInOutDOB = 2
	Enumeration used by the InOutState property to specify a data object as input data object, output data object or input/output data object.

LogMessageSeverity Enumeration	
Member	**Description**

Trace = 0
Info = 1
Warning = 2
	With the LogMessageSeverity constants, you can specify the output format of a message that is written to the message log.

LogMessageSeverity Enumeration	
Member	**Description**

Error = 3
SevereError = 4
SystemError = 5
Question = 6
Advice = 7
	The names have no leading "ad", because the enumeration is provided by a common component of dSPACE software.

OperationModeConstant Enumeration	
Member	**Description**

adOnline = 0
adOnlineRecording = 1
adOffline = 2
	Before you start an execution, you can specify the operation mode of the built-in libraries to online, online recording or offline operation mode. In offline operation mode, the required hardware and external devices are not connected. The previously parameterized offline data objects by recording or manual editing are used during execution.

RecordDepth Enumeration	
Member	**Description**

adRecordHighAndMedium = 0
adRecordHigh = 1
adRecordNone = 2
	The result of an execution has three possible record depths. If the result has the record depth "adRecordNone", no results of any objects are added to the report. If the record depth is "adRecordHigh", only objects with a high result level are added to the report. If the record depth is "adRecordHighAndMedium", all objects with "high" and "medium" result levels are added to the report.

ReportType Enumeration	
Member	**Description**

adHTML = 0
adPDF = 1
	With the ReportType constants, you can specify the output format of a report as PDF or HTML.

ResultLevel Enumeration	
Member	**Description**

adResultHigh = 0
adResultMedium = 1
adResultNone = 2
	The combination of result level and record depth determines the report content. By default, an object's result level is specified as "high", and a data object as "medium". If you set an object to "None" it will not appear in a report at all.

VerdictConstant Enumeration	
Member	**Description**

adVerdictExecuted = 0
adVerdictPassed = 1
adVerdictUndefined = 2
adVerdictFailed = 3
adVerdictError = 4
	The VerdictConstant constants are used to provide a verdict for automation elements.

Using constants

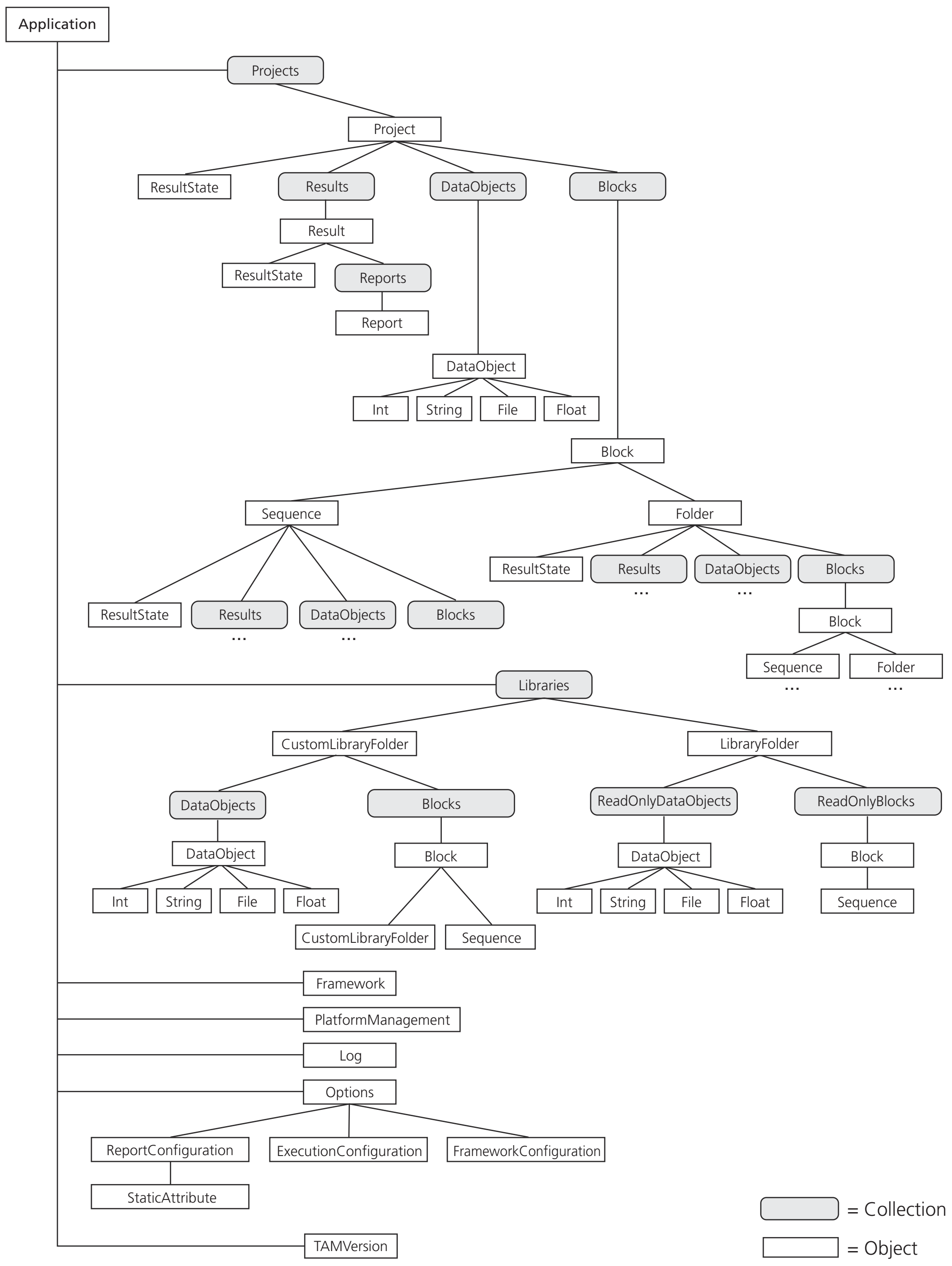
Note

If you want to use constants in your application, you have to note some specific programming instructions. For further information, refer to Using API Constants on page 67.

Overview of API Object Dependencies

Information on the dependency

The figure on the next page shows you an **extract** of the dependencies of the objects which are used in the API to handle automation tasks. For further objects look at the library-specific descriptions.


Application
Projects
Project
ResultState
Results
DataObjects
Blocks
Result
ResultState
Reports
Report
DataObject
Int
String
File
Float
Block
Sequence
Folder
ResultState
Results
DataObjects
Blocks
ResultState
Results
DataObjects
Blocks
Block
Sequence
Folder
Libraries
CustomLibraryFolder
LibraryFolder
DataObjects
Blocks
ReadOnlyDataObjects
ReadOnlyBlocks
DataObject
Block
DataObject
Block
Int
String
File
Float
Int
String
File
Float
Sequence
CustomLibraryFolder
Sequence
Framework
PlatformManagement
Log
Options
ReportConfiguration
ExecutionConfiguration
FrameworkConfiguration
StaticAttribute
TAMVersion
= Collection
= Object

Using the AutomationDesk API

Introduction

AutomationDesk provides a COM-based application programming interface (API) with which you can implement client applications accessing AutomationDesk.

Where to go from here

Information in this section

General Information on the AutomationDesk API

Introduction

The AutomationDesk API provides remote access to a subset of the AutomationDesk functionality.

Where to go from here

Information in this section

Information in other sections

Overview of the AutomationDesk API

Introduction

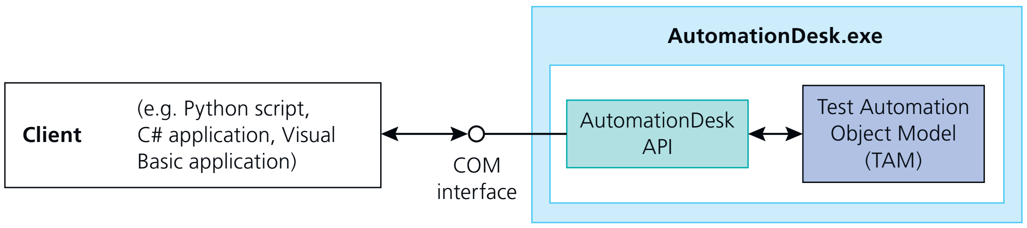
With the AutomationDesk API, you can implement client applications which access AutomationDesk.

Architecture

The AutomationDesk API is implemented as a COM object model. The Microsoft Component Object Model (COM) supports communication between objects from different applications. It can be used by any COM-compatible application, regardless of the programming language in which it was developed. The API allows remote-control access to AutomationDesk's test automation object model (TAM), which all automation features of AutomationDesk are based on.

Depending on your AutomationDesk license, you can use AutomationDesk either interactively via its user interface or via scripts by using it as a COM server.

For further information, refer to Using the API For Accessing AutomationDesk on page 34.


AutomationDesk.exe
Client
(e.g. Python script, C# application, Visual Basic application)
COM interface
AutomationDesk API
Test Automation Object Model (TAM)

Client programming

The COM server can be managed by a client application. The client can be implemented with any COM-compatible programming language.

The code examples in this document and the syntax descriptions in the AutomationDesk API Reference are written in Python, because the Python interpreter comes with your dSPACE installation. You can immediately work with the source code provided. To develop Python scripts you can use PythonWin, which you can find in the Windows Start menu (Start - Programs - Python 3.6 - PythonWin).

Some main differences between Python and other programming languages are described in Translating Code Into Other Programming Languages on page 78.

Features

One intention of the AutomationDesk API is to provide an interface for other software products which are used to develop and manage tests. The test information can be read and “translated” into an AutomationDesk project using the API. AutomationDesk executes the test and returns the test results to the test development tool. Another intention is to provide a subset of AutomationDesk features which is required for executing prepared projects, for example, if the execution of automation tasks is assigned to persons which shall not be able to modify an automation sequence.

With the AutomationDesk API, you can access almost all the commands of AutomationDesk's Project Manager required for these use cases:

- Project handling
  - Creating projects
  - Saving projects
  - Loading existing projects
  - Importing and exporting projects
- Folder handling
  - Creating folders
  - Renaming, moving, copying, and deleting folders
- Data object handling
  - Creating project-specific data objects
  - Creating sequence-specific data objects
  - Renaming, moving, and deleting data objects
  - Parameterizing data objects
- Sequence handling
  - Creating sequences
  - Adding sequences from the custom library
- Execution handling
  - Configuring the execution
  - Starting the execution on projects, folders, and sequences
- Result and report handling
  - Configuring the result logging
  - Viewing the result states
  - Configuring the report settings
  - Generating reports

For further information on these features, refer to Managing Projects (AutomationDesk Basic Practices). For the features that are not supported, refer to Limitations When Using the AutomationDesk API on page 499.

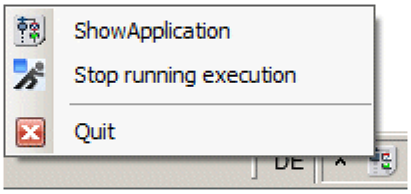
Event mechanism

Some objects of the API provide event handling. The COM server reports an event which the client can react to. Some of the available events are:

- OnProjectOpen, OnProjectClose (Application)
- OnAdd, OnRemove (Blocks, Data Objects, Results, Reports)
- OnModified (Sequence)

Controlling AutomationDesk UI-free

You can start AutomationDesk without displaying its user interface (UI-free) via AutomationDesk API. An AutomationDesk icon is placed in the notification area of the taskbar.



The context menu of the icon offers you to:

- Show the AutomationDesk user interface - ShowApplication
- Stop a running execution - Stop running execution
- Exit AutomationDesk - Quit

If you only have installed AutomationDesk with the AutomationDesk - Automation Server license, you only get the AutomationDesk icon and the ShowApplication dialog is unavailable.

Examples

- Start AutomationDesk:

```
import win32com.client
AutomationDesk = win32com.client.Dispatch("AutomationDesk.TAM")
```

- Make the user interface visible:

```
AutomationDesk.Visible = True   #Depending on your license
```

  If you only have the Automation Server license, the script stops with an error message.

- AutomationDesk is running with its user interface and you want to hide the user interface:

```
import win32com.client
AutomationDesk = win32com.client.Dispatch("AutomationDesk.TAM")
AutomationDesk.Visible = False
```

Related topics

Basics

Basic Concepts (AutomationDesk Introduction And Overview 🕮)
Limitations When Using the AutomationDesk API.......499
Managing Projects (AutomationDesk Basic Practices 🕮)
Translating Code Into Other Programming Languages.......78
Using the API For Accessing AutomationDesk.......34

Using the API For Accessing AutomationDesk

Introduction

If you want to use the AutomationDesk API for accessing AutomationDesk, there are some details to know.

Required installation

The AutomationDesk API is available with the AutomationDesk installation.

Number of instances

You can create and use only one AutomationDesk instance at the same time. If AutomationDesk is already opened, this instance is used for the API client.

Notes on using AutomationDesk

- You can access only those elements with the API which are managed by the Platform Manager. For a feature overview, refer to Overview of the AutomationDesk API on page 32.
- You cannot modify the settings of the user interface using the API, for example, size and position of the Project Manager.
- The scripts or applications you have developed using the API work with the AutomationDesk and the Automation Server license, except for the `Visible` property. For further information, refer to How to Create a COM Server on page 37.

Note

Do not close AutomationDesk while an API script is accessing it.

Related topics

Basics

HowTos

Programming Instructions

Introduction

There are two main use cases for API applications: creating projects and executing projects.

Where to go from here

Information in this section

Project Handling Using the API

Introduction

The AutomationDesk API provides commands to create a COM server and to manage projects.

Where to go from here

Information in this section

How to Create a COM Server

Objective

The AutomationDesk API is based on the COM technology. If you use the AutomationDesk API, you create a COM server interacting with AutomationDesk.

Python code examples

The code examples are implemented in Python. If you want to use other programming languages, refer to Translating Code Into Other Programming Languages on page 78.

Required time delays

If your PC has a low performance or a high CPU workload because of other processes, there are some situations in which time delays are required:

- To guarantee that all objects are correctly cleared, you should call the sleep function before deleting the COM server. A time delay of 5 seconds is mostly sufficient.
- You should also wait for about 10 seconds between the deletion and the recreation of a COM server.

Active AutomationDesk version

You can install different versions of AutomationDesk on the same PC, but only one version can be the active one.

Since AutomationDesk 3.4 you can activate and deactivate AutomationDesk using the Installation Manager, earlier versions have to be activated via the RCP and HIL software entry in the Installation Manager.

If multiple AutomationDesk versions are installed, you can use the ProgID with version information in the Dispatch call to force using a specific version independently of its activation.

Preconditions

- To clear all objects at program end or at program termination, you should always use the `try` statement.
- For using the API commands described, you must import the following modules:
  - `win32com.client`
  - `win32api`

Method

To create a COM server

1 Create a COM server that manages the API commands, type `AudObj = win32com.client.Dispatch("AutomationDesk.TAM")`.

> **Tip**
>
> To force using a specific AutomationDesk version, use the ProgID with version information.
>
> For example, to access AutomationDesk 4.0, type `AutomationDesk.TAM.4.0` in the Dispatch call.
>
> The specified AutomationDesk version must be installed.
>
> Using the ProgID without version information automatically connects the COM server to the currently activated AutomationDesk version.
>
> For AutomationDesk 3.6 and earlier:
>
> - Creating a COM server that accesses the AutomationDesk, type `AutomationDesk.TAM.x.x`
> - Creating a COM server that accesses the Automation Server, type `ADAutomation.TAM.x.x`

2 Delete the COM server after execution by resetting the created objects to `None`.

```
win32api.Sleep(5000)
AudObj = None
```

Result

You have created a COM server accessing AutomationDesk. Before deleting it again, you should wait for some seconds depending on the PC performance to guarantee that all created objects are cleared completely.

> **Note**
>
> Notes when using AutomationDesk:
>
> - If you close AutomationDesk while an API script is running, you will get the error message "The RCP server is unavailable".
> - The termination of the COM server does not cause AutomationDesk to close. To close AutomationDesk, use the **`Quit`** method.

Example

The following code shows an example of creating a COM server that accesses AutomationDesk. If only the AutomationDesk - Automation Server license is available, you are not allowed to make the user interface visible. For more information, refer to Overview of the AutomationDesk API on page 32.

```
import win32com.client
import win32api
import time
try:
    # Create the COM server
    AudObj = win32com.client.Dispatch("AutomationDesk.TAM")
    # Show the user interface of AutomationDesk
    AudObj.Visible = True # depends on your license
    # Close AutomationDesk
     AudObj.Quit()
finally:
    win32api.Sleep(5000)
    AudObj = None
```

Related topics

Basics

References

How to Create a Project Using the API

Objective

The AutomationDesk API can be used to automate the creation of AutomationDesk projects up to sequence level. The entry point for the project management is the project file (*.ADPX) and its project root element. You can create a new project with the API.

Python code examples

The code examples are implemented in Python. If you want to use other programming languages, refer to Translating Code Into Other Programming Languages on page 78.

Restrictions

- The path where you want to create a project must already exist.

Preconditions

- You must know how to create the COM server, refer to How to Create a COM Server on page 37.
- To clear all objects at program end or at program termination, you should always use the `try` statement.

- To use the API commands described, you must import the following modules:
  - `win32com.client`
  - `win32api`

Method

To create a project using the API

1 Start a COM server managing the API commands.

2 Create a new project by calling the create method of a Projects collection object. The method requires three parameters:

- `ProjectName` contains the path and name of the AutomationDesk project file (ADPX).
- `TemplateName` contains the name of the project template you want to use. For AutomationDesk Standard projects, you must specify “Standard Project”.
- `FileOption` decides whether an existing project is overwritten (1) or the project creation is canceled (0).

Note

If you overwrite an existing AutomationDesk project all its information is lost.

```
ProjsObj = AudObj.Projects
ProjObj = ProjsObj.Create("NameOfADPX","Standard Project",1)
```

3 Save the project by calling the save method of the Project object.

```
ProjObj.Save()
```

4 Close the project by calling the close method of the Project object.

```
ProjObj.Close()
```

5 Clear the created objects by setting them to `None` in reverse order of creation.

```
ProjObj = None
ProjsObj = None
AudObj = None
```

Result

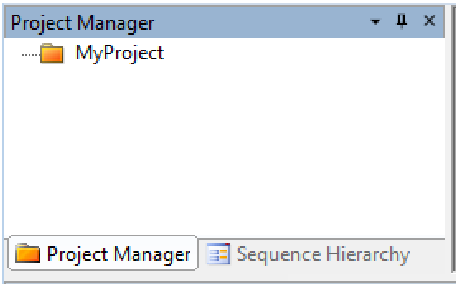
You have created a standard AutomationDesk project using the AutomationDesk API. The project is saved and the COM objects are cleared.

Example

The following code shows an example of creating a project:

```
import win32com.client
import win32api
try:
    # Create the COM Server
    AudObj = win32com.client.Dispatch("AutomationDesk.TAM")
    # Get the Projects collection
    ProjsObj = AudObj.Projects
    # Call the create method of the Projects collection
    # Edit ProjName to specify another AutomationDesk project
    ProjName = "C:\Work\MyProject.adpx"
    ProjObj = ProjsObj.Create(ProjName,"Standard Project",1)
    # Save and close the project
    ProjObj.Save()
    ProjObj.Close()
finally:
    ProjObj = None
    ProjsObj = None
    win32api.Sleep(5000)
    AudObj = None
```

If you open the created project in the AutomationDesk user interface, you will see the following structure in the Project Manager.



Another example of creating a project via AutomationDesk's API is available in `<DocumentsFolder>\API\Scripting_Python\CreateProject.py`.

Related topics

Basics

Managing Projects (AutomationDesk Basic Practices)
Translating Code Into Other Programming Languages........ 78

HowTos

How to Create a COM Server........ 37
How to Import a Project Using the API........ 43
How to Load a Project Using the API........ 42

References

Application........ 87
Project........ 121

How to Load a Project Using the API

Objective

The AutomationDesk API can be used to load an existing AutomationDesk project.

Python code examples

The code examples are implemented in Python. If you want to use other programming languages, refer to Translating Code Into Other Programming Languages on page 78.

Restrictions

- The project you want to load should not be already opened.

Preconditions

- You must know how to create the COM server, refer to How to Create a COM Server on page 37.
- To clear all objects at program end or at program termination, you should always use the `try` statement.
- To use the API commands described, you must import the following modules:
  - `win32com.client`
  - `win32api`

Method

To load a project using the API

1 Start a COM server that manages the API commands.

2 Load a project by calling the load method of a Projects collection object. The only parameter required is the project name containing the path and name of the AutomationDesk project file (*.ADPX).

```
# Get the Projects collection
ProjsObj = AudObj.Projects
# Load the existing project
ProjObj = ProjsObj.Load("NameOfADPX")
```

3 Implement further instructions before saving and closing the project.

Result

You have loaded an existing AutomationDesk project for further processing.

Example

The following code shows an example of loading the project you created in How to Create a Project Using the API on page 39.

```
import win32com.client
import win32api
try:
    # Create the COM Server
    AudObj = win32com.client.Dispatch("AutomationDesk.TAM")
    # Get the Projects collection
    ProjsObj = AudObj.Projects
    ProjName = "C:\Work\MyProject.adpx"
    ProjObj = ProjsObj.Load(ProjName)
    # Do something ...
    # Save and close the project
    ProjObj.Save()
    ProjObj.Close()
finally:
    ProjObj = None
    ProjsObj = None
    win32api.Sleep(5000)
    AudObj = None
```

Related topics

Basics

HowTos

References

How to Import a Project Using the API

Objective

The AutomationDesk API can be used to load an AutomationDesk project from a ZIP file or an XML file.

Project import and export

You can import a project from a ZIP file, if it was exported before. The ZIP file contains all the information of the project and its project elements. If you import it, the ZIP archive is extracted to the folder in the file system where the archive is stored.

You can import a project from an XML file that is created before by using AutomationDesk's export command, or by generating or writing with external tools. The generated XML files must fit the AutomationDesk XML schema definitions.

Note

With AutomationDesk 6.1, a new XML format is introduced for exporting and importing AutomationDesk elements. The XML format used for exporting and importing elements with AutomationDesk 6.0 and earlier is now called *legacy XML*. It is available only for importing existing XML export files. The legacy XML format is not available for exporting elements and will be discontinued in future versions of AutomationDesk.
Both XML file formats are specified by the `adXML` enumeration. The XML format to be used is automatically identified by the specified file suffix. If you want to export to a legacy XML file, an exception occurs. If you import a file in the legacy XML format, a warning is written to the log file, which informs you about the planned discontinuation.

For further information, refer to Exporting and Importing Projects and Project Elements (AutomationDesk Basic Practices 📖).

Note

The methods for XML import and export are supported only by the following objects:

- Projects2
- Projects1
- Project1
- Folder1
- Sequence1

Usually, you do not have to adapt your script because the origin objects inherit the new methods. However, if you are using a Python wrapper in your script, you must specify the above objects as object types. For further information, refer to Using Constants in a Python Script on page 68.

Python code examples

The code examples are implemented in Python. If you want to use other programming languages, refer to Translating Code Into Other Programming Languages on page 78.

Restrictions

- The project you want to import should not be already opened.

Preconditions

- You must know how to create the COM server, refer to How to Create a COM Server on page 37.
- To clear all objects at program end or at program termination, you should always use the `try` statement.
- To use the API commands described, you must import the following modules:
  - `win32com.client`
  - `win32api`

Method

To import a project using the API

1 Start a COM server that manages the API commands.

2 Import a project by calling the import method of a Projects collection object. The method requires three parameters:

- `FileName` contains the path and name of the AutomationDesk project file to be imported.
- `FileFormat` decides whether to import a project from a ZIP file (0) or from an XML file (1).
- `FileOption` decides whether an existing project is overwritten (1) or the project import is canceled (0).

```
# Get the Projects collection
ProjsObj = AudObj.Projects
# Import the exported project
ProjObj = \
    ProjsObj.ImportProject("C:\Work\MyProject.zip",0,1)
```

3 Implement further instructions before saving and closing the project.

Result

You have imported an exported AutomationDesk project for further processing.

Example

The following code shows an example of importing one of the AutomationDesk demo projects stored in <DocumentsFolder>. You must specify the absolute path and the file name of the zipped project in `ZipFile`.

```
import win32com.client
import win32api
import os
import sys
try:
    # Specify the zipped demo project with its absolute path in AbsoluteZipFile
    AbsoluteZipFile = \
        r"C:\Users\ADUser\Documents\dSPACE\AutomationDesk\6.0\Main Library\MainLibraryExamples.zip"
    # Create the COM Server
    AudObj = win32com.client.Dispatch("AutomationDesk.TAM")
    # Get the Projects collection
    ProjsObj = AudObj.Projects
    ProjObj = ProjsObj.Import(AbsoluteZipFile, 1)
    # Do something ...
    # Save and close the project
    ProjObj.Save()
    ProjObj.Close()
```

```
finally:
    ProjObj = None
    ProjsObj = None
    win32api.Sleep(5000)
    AudObj = None
```

Related topics

Basics

HowTos

References

How to Structure a Project Using the API

Objective

An AutomationDesk project can be structured by adding folders to the project tree.

Python code examples

The code examples are implemented in Python. If you want to use other programming languages, refer to Translating Code Into Other Programming Languages on page 78.

Restrictions

If you try to add a folder with the same name in the same project hierarchy, the creation of this object is canceled.

Preconditions

- You must know how to create the COM server, refer to How to Create a COM Server on page 37.
- You must know how to create a project, refer to How to Create a Project Using the API on page 39. If you want to add a folder to an existing project structure, you must ascertain the project tree beforehand. For the required instructions, refer to How to Add Sequences to Your Project on page 50.

- The project you want to modify should not be opened in AutomationDesk.
- To clear all objects at program end or at program termination, you should always use the `try` statement.
- To use the API commands described, you must import the following modules:
  - `win32com.client`
  - `win32api`

Method

To structure a project using the API

1 Create a new project, or open or import an existing project.

2 Instantiate a folder template from the Standard library, which is an element of the Libraries collection.

> **Note**
>
> The Standard library is not accessible in AutomationDesk's Library Browser. It contains the project, folder, and sequence elements you can create via the context menu of the Project Manager.

```
# Get the Libraries collection
LibsObj = AudObj.Libraries
# Get the Standard library
StdLibObj = LibsObj.Item("Standard")
# Get a folder template
FolderTemplObj = StdLibObj.SubBlocks.Item("Folder")
```

3 Add the folder to the project root element by using the create method of the project's SubBlocks collection . The create method contains five parameters:

```
Create(SourceBlock, Position, ConfirmBreakLink, \
        InsertBeforeBlock, InsertAfterBlock)
```

This method can be used in different ways:

- `NewObj = ParentObj.SubBlocks.Create(TemplateObj)`

  To create a new object on the default location.
- `NewObj = ParentObj.SubBlocks.Create(TemplateObj, Pos)`

  To create a new object on the specified position in the same hierarchy level. The position of the first element is 0.

-
  ```
  NewObj = ParentObj.SubBlocks.Create(TemplateObj,\
            -1, 0, InsertBeforeBlock)
  ```

  To create a new object and insert it *before* the specified block. The position must be set to -1. With the third parameter you decide whether an existing library link will be broken.

-
  ```
  NewObj = ParentObj.SubBlocks.Create(TemplateObj,\
             -1, 0, None, InsertAfterBlock)
  ```

To create a new object and insert it *after* the specified block. The position must be set to -1. With the third parameter you decide whether an existing library link will be broken. The parameter for the `InsertBeforeBlock` must be set to `None`.

Here, we create the folder object at the default location:

```
FolderObj = ProjObj.SubBlocks.Create(FolderTemplObj)
```

A folder with its default name “Folder” is added to the project’s root element.

4 Create a subfolder at its default location by adding the folder template to the folder object already created.

```
SubfolderObj = FolderObj.SubBlocks.Create(FolderTemplObj)
```

5 Change the default names of the created folders using the name property of a Block object.

```
FolderObj.Name = "NameForFolder"
SubfolderObj.Name = "NameForSubfolder"
```

6 Save and close the project.

7 Clear the created objects by setting them to `None`.

Result

You have added two folders to the project’s root element in different project hierarchy levels.

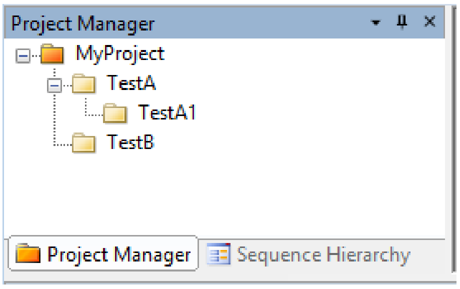
Example

The following code shows an example how to build a project structure.

```
import win32com.client
import win32api
try:
    AudObj = win32com.client.Dispatch("AutomationDesk.TAM")
    # Get the Projects collection
    ProjsObj = AudObj.Projects
    # Load the existing project
    ProjObj = ProjsObj.Load(r"C:\Work\MyProject.adpx")
    # Get the Libraries collection
    LibsObj = AudObj.Libraries
    # Get the Standard library
    StdLibObj = LibsObj.Item("Standard")
    # Get a folder template
    FolderTemplObj = StdLibObj.SubBlocks.Item("Folder")
    # Create two folders as child elements of the project element
    FolderAObj = ProjObj.SubBlocks.Create(FolderTemplObj)
    FolderBObj = ProjObj.SubBlocks.Create(FolderTemplObj)
    # Create a subfolder for the first folder
    SubfolderAObj = FolderAObj.SubBlocks.Create(FolderTemplObj)
    # Rename the folders
    FolderAObj.Name = "TestA"
    FolderBObj.Name = "TestB"
    SubfolderAObj.Name = "TestA1"
    # Save and close the project
    ProjObj.Save()
    ProjObj.Close()
```

```
finally:
    SubfolderAObj = None
    FolderBObj = None
    FolderAObj = None
    FoldTemplObj = None
    StdLibObj = None
    LibsObj = None
    ProjObj = None
    ProjsObj = None
    win32api.Sleep(5000)
    AudObj = None
```

If you open the created project in AutomationDesk, you will see the following structure in the Project Manager:



Another example of structuring a project via AutomationDesk's API is available in `<DocumentsFolder>\API\Scripting_Python\CreateProject.py`.

Related topics

Basics

Managing Projects (AutomationDesk Basic Practices)

HowTos

How to Add Data Objects to Your Project 56
How to Add Sequences to Your Project 50
How to Create a COM Server 37
How to Create a Project Using the API 39

References

Blocks 92
Folder 105
Libraries (Object) 110

How to Add Sequences to Your Project

Objective

With the AutomationDesk API, you can add an empty sequence to your project.

Python code examples

The code examples are implemented in Python. If you want to use other programming languages, refer to Translating Code Into Other Programming Languages on page 78.

Restrictions

- With the API, you cannot build the content of a sequence, for example, the control flow of an automation task.
- If you try to add a sequence with the same name in the same project hierarchy level, the execution is canceled.
- You can add a sequence only to a project element or to a folder. You cannot add a sequence to a data object or another sequence.

Preconditions

- You must know how to create the COM server, refer to How to Create a COM Server on page 37.
- You must know how to create a project using AutomationDesk or the Automation Server, refer to How to Create a Project Using the API on page 39.
- You must know how to create folders in a project, refer to How to Structure a Project Using the API on page 46.
- For the following instructions you can use a project with at least one folder, for example, the project that you created in the previous topic.
- The project you want to modify should not be opened in AutomationDesk.
- To clear all objects at program end or at program termination, you should always use the `try` statement.
- To use the API commands described, you must import the following modules:
  - `win32com.client`
  - `win32api`

Method

To add a sequence to your project

1 Create a new project or open an existing project.

2 Determine the existing child elements of the project element.

You can use the count property to get the number of the child elements. These can be folders and sequences.

```
NoOfChildren = ProjObj.SubBlocks.Count
```

You can use the names property to get the names of the child elements. The returned value is of list type.

```
NameOfChildren = ProjObj.SubBlocks.Names
```

3 Create an object for the folder you want to add the sequence. You can get a specific folder by specifying the name of an existing folder, or by specifying its list index, starting with 0.

```
FolderObj = ProjObj.SubBlocks.Item(Index)
```

4 Instantiate a sequence template from the Standard library, which is an element of the Libraries collection.

Note

The Standard library is not accessible in AutomationDesk's Library Browser. It contains the project, folder, and sequence elements you can create via the context menu of the Project Manager.

```
# Get the Libraries collection
LibsObj = AudObj.Libraries
# Get the Standard library
StdLibObj = LibsObj.Item("Standard")
# Get a sequence template
SequenceTemplObj = StdLibObj.SubBlocks.Item("Sequence")
```

5 Add the sequence to the folder element using the create method of the Folder object.

```
SequenceObj = FolderObj.SubBlocks.Create(SequenceTemplObj)
```

A sequence with its default name 'Sequence' is added to the folder's SubBlocks collection.

6 Change the default name of the sequence using the name property of a Block object.

```
SequenceObj.Name = "NameForSequence"
```

7 Save and close the project.

8 Clear the created objects by setting them to `None`.

Result

You have added a sequence to an existing folder.

Tip

By instantiating the Test Builder library, you can also add a TestCase object to your project.

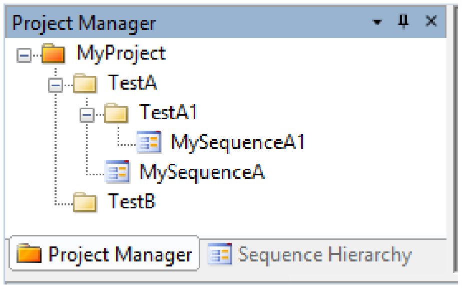
Example

The following code shows an example of adding one sequence to a folder and one to a subfolder using the AutomationDesk project, generated in How to Structure a Project Using the API on page 46.

```
import win32com.client
import win32api
```

```
try:
    AudObj = win32com.client.Dispatch("AutomationDesk.TAM")
    # Get the Projects collection
    ProjsObj = AudObj.Projects
    # Load the existing project
    ProjObj = ProjsObj.Load(r"C:\Work\MyProject.adpx")
    # Get the number of the project’s child elements
    # (existing folders and sequences)
    NoOfChildren = ProjObj.SubBlocks.Count
    print("Project contains %i element(s)" %NoOfChildren)
    # Get the names of the project’s elements
    NameOfChildren = ProjObj.SubBlocks.Names
    for i in NameOfChildren:
        print(i)
    # Get the folder object you want to add the first sequence
    # Select the first element by index - 'Item(0)'
    FolderObj = ProjObj.SubBlocks.Item(0)
    # Get the subfolder object you want to add the second sequence
    # Select the element by name - 'Item("TestA1")'
    SubfolderObj = FolderObj.SubBlocks.Item("TestA1")
    # Get the Libraries collection
    LibsObj = AudObj.Libraries
    # Get the Standard library
    StdLibObj = LibsObj.Item("Standard")
    # Get a sequence template
    SequenceTemplObj = StdLibObj.SubBlocks.Item("Sequence")
    # Add a sequence to the folder
    SequenceObj = FolderObj.SubBlocks.Create(SequenceTemplObj)
    SequenceObj.Name = "MySequenceA"
    # Add a sequence to the subfolder
    Sequence2Obj = SubfolderObj.SubBlocks.Create(SequenceTemplObj)
    Sequence2Obj.Name = "MySequenceA1"
    # Save and close the project
    ProjObj.Save()
    ProjObj.Close()
finally:
    Sequence2Obj = None
    SequenceObj = None
    SequenceTemplObj = None
    StdLibObj = None
    LibsObj = None
    SubfolderObj = None
    FolderObj = None
    ProjObj = None
    ProjsObj = None
    win32api.Sleep(5000)
    AudObj = None
```

If you open the created project in AutomationDesk, you will see the following structure in the Platform Manager.



Another example of adding sequences to a project via AutomationDesk's API is available in `<DocumentsFolder>\API\Scripting_Python\CreateProject.py`.

Related topics

Basics

Building Automation Sequences (AutomationDesk Basic Practices 🕮)
Managing Projects (AutomationDesk Basic Practices 🕮)

HowTos

References

How to Add Custom Sequences to Your Project

Objective

If you want to create a new project with executable sequences, you can add prepared sequences from a custom library to your project.

Python code examples

The code examples are implemented in Python. If you want to use other programming languages, refer to Translating Code Into Other Programming Languages on page 78.

Restriction

You can only access custom sequences from a custom library with the API. Customized automation blocks which are stored in a custom library are not accessible.

Preconditions

- You must know how to create the COM server, refer to How to Create a COM Server on page 37.
- You must know how to create or load a project, refer to How to Create a Project Using the API on page 39.
- The project you want to modify should not be opened in AutomationDesk.
- To clear all objects at program end or at program termination, you should always use the `try` statement.

- To use the API commands described, you must import the following modules:
  - `win32com.client`
  - `win32api`
- The custom sequence you want to add to the project must be available on your computer. For further information, refer to Lesson 4: Creating Custom Libraries (AutomationDesk Tutorial 📖) and Lesson 5: Working With the Custom Library (AutomationDesk Tutorial 📖).

Method

To add a prepared sequence to your project

1. Create a COM server.
2. Create a project or open an existing project.
3. Instantiate an object for the custom library, which is an element of the Libraries collection.

```
# Get the Libraries collection
LibsObj = AudObj.Libraries
# Get the custom library
CustomLibObj = LibsObj.Item("Custom Library")
```

4. Use the properties of the library object to get the available custom sequences, for example:

```
NoOfElements = CustomLibObj.SubBlocks.Count
print("Custom library contains %i element(s)"\
                  %NoOfElements)
# Get the names of the custom library’s elements
NameOfElements = CustomLibObj.SubBlocks.Names
for i in NameOfElements:
    print(i)
```

5. Instantiate a custom sequence template by name or index.

```
# Get a custom sequence template
CustomSeqTemplObj = CustomLibObj.SubBlocks.Item("SeqName")
# or
# CustomSeqTemplObj = CustomLibObj.SubBlocks.Item(Index)
```

6. Add the sequence to the project or folder element using the create method. For example:

```
SequenceObj = FolderObj.SubBlocks.Create(CustomSeqTemplObj)
```

   A sequence with the template’s name is added to the folder element. If a sequence with the same name already exists, the added sequence is renamed according to AutomationDesk’s default naming concept.
7. Change the name of the sequence using the name property of a Block object.

```
SequenceObj.Name = "NameForCustomSequence"
```

8. Save and close the project.
9. Clear the created objects by setting them to `None`.

Result

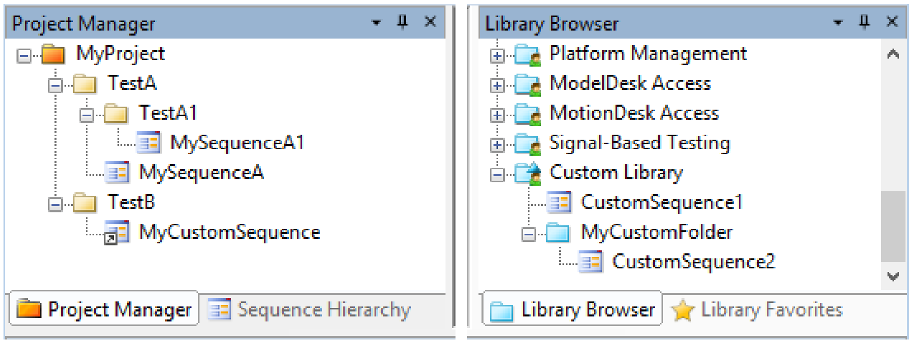
You have added an executable sequence from the custom library to a project.

Example

The following code shows an example of adding a prepared sequence from the custom library to a folder using the AutomationDesk project, generated in How to Add Sequences to Your Project on page 50. The custom library must contain a sequence with the name “CustomSequence2”.

```
import win32com.client
import win32api
try:
    AudObj = win32com.client.Dispatch("AutomationDesk.TAM")
    # Get the Projects collection
    ProjsObj = AudObj.Projects
    # Load the existing project
    ProjObj = ProjsObj.Load(r"C:\Work\MyProject.adpx")
    # Get the folder 'TestB' as object
    FolderObj = ProjObj.SubBlocks.Item("TestB")
    # Get the Libraries collection
    LibsObj = AudObj.Libraries
    # Get the Custom library
    CustomLibObj = LibsObj.Item("Custom Library")
    # Get a custom sequence template
    CustomLibSubFolder = CustomLibObj.SubBlocks.Item("MyCustomFolder")
    CustomSequenceTemplObj = CustomLibSubFolder.SubBlocks.Item("CustomSequence2")
    # Add a sequence to the folder containing the custom sequence
    SequenceObj = FolderObj.SubBlocks.Create(CustomSequenceTemplObj)
    SequenceObj.Name = "MyCustomSequence"
    # Save and close the project
    ProjObj.Save()
    ProjObj.Close()
finally:
    SequenceObj = None
    CustomSequenceTemplObj = None
    CustomLibObj = None
    LibsObj = None
    FolderObj = None
    ProjObj = None
    ProjsObj = None
    win32api.Sleep(5000)
    AudObj = None
```

If you open the created project in AutomationDesk, you will see the following structure in the Platform Manager. You can also see that the custom sequence which you want to add to the project is stored in a subfolder of the custom library. In which folder of the custom library the custom sequence resides is irrelevant concerning the linking mechanism. For further information, refer to Basics on Custom Library Links (AutomationDesk Basic Practices).



Related topics

Basics

Building Automation Sequences (AutomationDesk Basic Practices 📖)
Lesson 4: Creating Custom Libraries (AutomationDesk Tutorial 📖)
Lesson 5: Working With the Custom Library (AutomationDesk Tutorial 📖)
Managing Projects (AutomationDesk Basic Practices 📖)
Translating Code Into Other Programming Languages.......... 78

HowTos

How to Add Data Objects to Your Project.......... 56
How to Add Sequences to Your Project.......... 50
How to Create a COM Server.......... 37
How to Create a Project Using the API.......... 39
How to Structure a Project Using the API.......... 46

References

Blocks.......... 92
Libraries (Object).......... 110
Sequence.......... 145

How to Add Data Objects to Your Project

Objective

With the AutomationDesk API, you can add data objects to your project.

Python code examples

The code examples are implemented in Python. If you want to use other programming languages, refer to Translating Code Into Other Programming Languages on page 78.

Restrictions

- With the API, you cannot create and parameterize internal data objects of the automation blocks. You can only create and specify project-specific data

objects that are added to the project tree. They can be referenced by the data objects used in a sequence.
- If you try to add a data object with the same name in the same project hierarchy level, the creation of this object is canceled.
- Not all libraries provide specific data objects to be accessed.

Preconditions

- You must know how to create the COM server, refer to How to Create a COM Server on page 37.
- You must know how to create a project using AutomationDesk or the Automation Server, refer to How to Create a Project Using the API on page 39.
- You must know how to create folders in a project, refer to How to Structure a Project Using the API on page 46.
- You must know how to create sequences in a project, refer to How to Add Sequences to Your Project on page 50.
- The project you want to modify should not be opened in AutomationDesk.
- To clear all objects at program end or at program termination, you should always use the `try` statement.
- To use the API commands described, you must import the following modules:
  - `win32com.client`
  - `win32api`

Method

To add data objects to your project

1 Create a project or open an existing project.

2 Instantiate templates for the data objects you want to add to the project. The following data objects are elements of the AutomationDesk Main Library. To add data objects from other libraries to your project, you must instantiate those libraries instead.

```
# Get the Libraries collection
LibsObj = AudObj.Libraries
# Get the Main library
MainLibObj = LibsObj.Item("Main Library")
# Get a template for a string
StringTemplObj = MainLibObj.DataObjects.Item("String")
# Get a template for an integer
IntTemplObj = MainLibObj.DataObjects.Item("Int")
# Get a template for a float
FloatTemplObj = MainLibObj.DataObjects.Item("Float")
# Get a template for a file
FileTemplObj = MainLibObj.DataObjects.Item("File")
# Get a template for a data container
DataContainerTemplObj = \
    MainLibObj.DataObjects.Item("DataContainer")
```

3 Add a string data object to the project, rename it, and parameterize its value.

```
StringObj = ProjObj.DataObjects.Create(StringTemplObj)
StringObj.Name = "MyString"
StringObj.Value = "Hello world!"
```

4 Add an int data object to the project, rename it, and parameterize its value.

```
IntObj = ProjObj.DataObjects.Create(IntTemplObj)
IntObj.Name = "MyInt"
IntObj.Value = 12345
```

5 Add a float data object to the project, rename it, and parameterize its value.

```
FloatObj = ProjObj.DataObjects.Create(FloatTemplObj)
FloatObj.Name = "MyFloat"
FloatObj.Value = 12.345
```

6 Add a file data object to the project, rename it, and parameterize its value.

```
FileObj = ProjObj.DataObjects.Create(FileTemplObj)
FileObj.Name = "MyFile"
FileObj.Value = "C:\Work\Example.adpx"
```

7 Add a data container object to the project, rename it, and add a string to it by using the `ChildDataObjects` property.

```
DataContainerObj = ProjObj.DataObjects.Create(DataContainerTemplObj)
DataContainerObj.Name = "MyDataContainer"
StringObj = DataContainerObj.ChildDataObjects.Create(StringTemplObj)
```

8 Save and close the project.

9 Clear the created objects by setting them to `None`.

Result

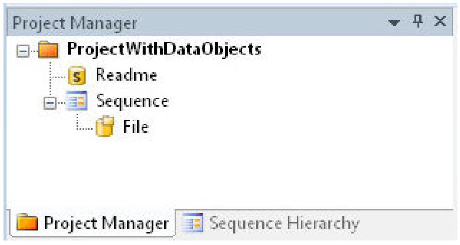
You have added data objects to a project.

Example

The following code shows an example of adding data objects to different hierarchy levels in the project tree. A string data object is added to the project element, and a file and a platform data object are added to the sequence.

```
import win32com.client
import win32api
try:
    # Create the COM server for the Automation Server
    AudObj = win32com.client.Dispatch("AutomationDesk.TAM")
    # Create the project
    # Get the Projects collection
    ProjsObj = AudObj.Projects
    # Call the create method of the Projects collection
    # Edit ProjName to specify another AutomationDesk project
    ProjName = r"C:\Work\ProjectWithDataObjects.adpx"
    ProjObj = ProjsObj.Create(ProjName,"Standard Project",1)
    # Create a sequence
    # Get the Libraries collection
    LibsObj = AudObj.Libraries
    # Get the Standard library
    StdLibObj = LibsObj.Item("Standard")
    # Get a sequence template
    SequenceTemplObj = StdLibObj.SubBlocks.Item("Sequence")
    # Add a sequence to the project
    SequenceObj = ProjObj.SubBlocks.Create(SequenceTemplObj)
    # Add the data objects to the project
    # Get the Main library
    MainLibObj = LibsObj.Item("Main Library")
    # Get a string template
    StringTemplObj = MainLibObj.DataObjects.Item("String")
    # Add the string to the project
    StringObj = ProjObj.DataObjects.Create(StringTemplObj)
    # Rename the data object
    StringObj.Name="Readme"
    # Set the string value
    StringObj.Value = "This project contains data objects on several hierarchies."
    # Get a file data object template
    FileTemplObj = MainLibObj.DataObjects.Item("File")
    # Add the file data object to the sequence
    FileObj = SequenceObj.DataObjects.Create(FileTemplObj)
    # Save and close the project
    ProjObj.Save()
    ProjObj.Close()
finally:
    # Clear all created objects
    FileObj = None
    FileTemplObj = None
    StringObj = None
    StringTemplObj = None
    MainLibObj = None
    SequenceObj = None
    SequenceTemplObj = None
    FolderObj = None
    FolderTemplObj = None
    StdLibObj = None
    LibsObj = None
    ProjObj = None
    ProjsObj = None
    win32api.Sleep(5000)
    AudObj = None
```

If you open the created project in AutomationDesk, you will see the following structure in the Platform Manager.



Another example of adding data objects to a project via AutomationDesk's API is available in `<DocumentsFolder>\API\Scripting_Python\MainLibrary.py`.

Related topics

Basics

Project Execution Using the API

Introduction

The AutomationDesk API provides commands to configure the settings for execution and report generation, and executing projects.

Where to go from here

Information in this section

How to Configure an Execution

Objective

With the AutomationDesk API, you can configure the settings of an execution.

Execution settings

The AutomationDesk API provides the following execution settings:

- Option for creating a result
- Option for generating a report
- Selection of the record depth
- Edit field for the result name
- Edit field for a description

If you use the API, these settings are not combined in one object, but split into the ExecutionConfiguration object and the execute method of the object to be executed. The settings that you specify for an execution are valid for one session only. If you create a new COM server, the execution configuration contains the default values.

Setting	Default Value	Object
Create result	true	ExecutionConfiguration object (CreateResult property)
Generate report	false	ExecutionConfiguration object (CreateReport property)
Record depth	None	ExecutionConfiguration object (RecordDepth property)
Result name	"Result"	Project, Folder, and Sequence object (ExecutionName parameter of the Execute method)
Description	" "	Project, Folder, and Sequence object (Description parameter of the Execute method)

Note

The libraries are used in the operation mode that you specified in AutomationDesk.
For example, if you have set the XIL API Convenience library in AutomationDesk to the offline mode, the Automation Server will execute it also in offline mode.
Use the `OperationMode` property to get or set the library-specific operation mode in AutomationDesk before you start an API script that is accessing external devices.

For detailed information on configuring the execution, refer to Executing Automation Sequences (AutomationDesk Basic Practices).

Python code examples

The code examples are implemented in Python. If you want to use other programming languages, refer to Translating Code Into Other Programming Languages on page 78.

Preconditions

- To use the API constants available for the execution configuration, you must import a Python wrapper beforehand. For further information, refer to Using API Constants on page 67.
- To clear all objects at program end or at program termination, you should always use the `try` statement.
- To use the API commands described, you must import the following modules:
  - `win32com.client`
  - `win32api`

Method

To configure an execution

1 Create a COM server.

2 Get the Options object of the created application.

```
OptionsObj = AudObj.Options
```

3 Get the ExecutionConfiguration object from the Options object.

```
ExecCfgObj = OptionsObj.Execution
```

4 Specify the execution settings:

Property	Possible Values	Description
ExecCfgObj.CreateResult	0 1 (default)	"0" means that the execution is not logged and no result is created. "1" means that the execution is logged and stored in a result.
ExecCfgObj.CreateReport	0 1 (default)	This setting is only considered if a result exists. "0" means that a report is not generated directly after the execution. You can generate a report later on, independently of execution. "1" means that a report is generated directly after the execution.
ExecCfgObj.RecordDepth	0 (high and medium) 1 (high) 2 (none; default)	The record depth specifies the amount of information that is to be logged. It corresponds to the result level (None, Medium, High) which you can specify for blocks and data objects. A block with a "Medium" result level is not logged in a result specified with a "High" record depth. You can also use the constants `adRecordNone`, `adRecordHigh`, and `adRecordHighAndMedium`.

5 Specify the result name and the description as parameters of the execute method.

6 Implement further instructions in your script and save it.

Result

If you start the execution, the result will be created according to the specified configuration settings.

Example

An example of executing a project via AutomationDesk's API is available in `<DocumentsFolder>\API\Scripting_Python\ExecuteProject.py`.

Related topics

Basics

HowTos

References

How to Configure the Report Generation

Objective

With the AutomationDesk API, you can configure the settings of the report generation.

Report settings

The AutomationDesk API provides the following report settings:

- Style sheet used
- Logo used and its placement
- Attributes which are to be included in the report, for example, date and time of execution.

For detailed information on the report settings, refer to Generating Reports (AutomationDesk Basic Practices 🕮).

Python code examples

The code examples are implemented in Python. If you want to use other programming languages, refer to Translating Code Into Other Programming Languages on page 78.

Preconditions

- The content of a report depends on its result. A Result object must exist before you can generate a report.
- To use the API constants available for the report generation, you must import a Python wrapper beforehand. For further information, refer to Using API Constants on page 67.

Method

To configure the report generation

1 Create a COM server.

2 Get the Options object of the created application.

```
OptionsObj = AudObj.Options
```

3 Get the ReportConfiguration object from the Options object

```
ReportCfgObj = OptionsObj.Report
```

4 Specify the report settings:

Property	Possible Values	Description
ReportCfgObj.ReportType	0 (HTML) 1 (PDF)	Indicates the output format of the report. The possible values correspond to the constants `adHTML` and `adPDF`.
ReportCfgObj.IsCustomReport	0 (standard) 1 (customized)	Indicates whether a custom style sheet for the report generation is used.
ReportCfgObj.StylesheetPath	" "	Specifies the path to the style sheet you want to use. This setting is required only if you want to use a custom style sheet.
ReportCfgObj.LogoPath	" "	Specifies the path to the logo you want to add to the report. The default is the dSPACE logo.
ReportCfgObj.LogoAlignment	0 (left) 1 (center) 2 (right)	Specifies the horizontal alignment of the logo. The default setting depends on the registry entry. You can also use the constants `adLeft`, `adCenter`, and `adRight`.
ReportCfgObj.IsAllAttributes	0 (all) 1 (customized)	Specifies whether you want to add all the available attributes to the report, or a customized subset of them.
ReportCfgObj.VisibleAttributes	" "	Specifies the subset of attributes which should be added to the report.
ReportCfgObj.StaticAttribute	StaticAttribute object	Specifies if you want to add the following information to the report: ▪ Folder and project information ▪ Descriptions ▪ Result states (passed, failed, unknown) ▪ Results of Report blocks

5 Implement further instructions in your script and save it.

Result

The specified report settings are saved on your PC. They will be used for any subsequent report generation until you modify them. They are used for automatic report generation started directly at the end of result logging, and explicit generation using the GenerateReport method of the Reports object.

Related topics

Basics

HowTos

References

How to Execute a Project Using the API

Objective

With the AutomationDesk API, you can execute an AutomationDesk project.

Python code examples

The code examples are implemented in Python. If you want to use other programming languages, refer to Translating Code Into Other Programming Languages on page 78.

Preconditions

- You must know how to create the COM server, refer to How to Create a COM Server on page 37.
- You must know how to load a project, refer to How to Structure a Project Using the API on page 46.
- You must know how to specify the result and report configuration, refer to How to Configure an Execution on page 60 and How to Configure the Report Generation on page 63.
- The project you want to execute should not be opened in AutomationDesk.
- To clear all objects at program end or at program termination, you should always use the `try` statement.
- To use the API commands described, you must import the following modules:
  - `win32com.client`
  - `win32api`
- To use the API constants available for the project execution, you must import a Python wrapper beforehand. For further information, refer to Using API Constants on page 67.

Note

The libraries are used in the operation mode that you specified in AutomationDesk.
For example, if you have set the XIL API Convenience library in AutomationDesk to the offline mode, the Automation Server will execute it also in offline mode.
Use the `OperationMode` property to get or set the library-specific operation mode in AutomationDesk before you start an API script that is accessing external devices.

Method

To execute a project using the API

1 Open an existing project by using the load or import method of the Projects object.

2 Configure the execution settings.

3 Configure the report settings if required.

4 Implement the execution of the project.

```
ResObj = ProjObj.Execute("MyResult", "MyDescription")
```

5 Implement further instructions to handle the result, for example, for evaluating the result states.

```
ResStateObj = ResObj.ResultState
Verdict = ResStateObj.Verdict
if Verdict == 4 :
    print ("An unexpected error raised during execution.")
if Verdict == 3 :
    print ("The execution failed.")
if Verdict == 2 :
    print ("The execution state cannot be verified.")
if Verdict == 1 :
    print ("The execution sucessfully passed.")
if Verdict == 0 :
    print ("The execution ended.")
```

6 Save your script.

Result

You have executed a project. The result is logged according to the specified execution settings. If you specified that a report should be generated directly after execution, a report is also generated according to the report settings.

Note

If you use the AutomationDesk API for executing a sequence containing automation blocks of the Dialogs library, it could be possible - depending on the other opened applications - that an input or message dialog is not opened on top of your window. The execution is interrupted until you close the dialog.

Related topics

Basics

HowTos

References

Using API Constants

Introduction

The AutomationDesk API provides constants which you can use for specifying parameters, for example, the output format of reports. If you want to use these constants, you must make them available explicitly in a Python wrapper, a C# reference, or a Visual Basic reference, and your source code must be modified.

Where to go from here

Information in this section

Using Constants in a Python Script

Getting a Python wrapper

If you want to use the constants of the AutomationDesk API, copy a Python wrapper according to the used AutomationDesk version to your working directory.

You can use the `TAMAutomation.py` file as Python wrapper that is located in the `<DocumentsFolder>\API\Scripting_Python` folder.

Note

The wrapper must not be available in the `gen_py` folder, because the `win32com` module imports it automatically also for scripts which do not use the wrapper.

Using constants

To use a constant from the API in your script, you must import the Python wrapper and enter the required constant in dot notation. For example, to specify the output format of a report as PDF, you must use the following command:

```
import TAMAutomation
...
ReportCfgObject.ReportType = TAMAutomation.constants.adPDF
```

Notes on using the Python wrapper

Note

- If you import the Python wrapper, it is stored in the Python namespace and used for each script running in the current session. Before you execute a Python script that does not use the wrapper, you must close and restart PythonWin to clear its namespace.
- If you use the wrapper in your script, an instantiated element of a collection object does not know its type. If you implemented type-specific methods or properties, your application stops with an exception. You must identify the element type beforehand, and then instantiate a new object using the wrapper's interface definition.

Examples

Here are some examples showing how to use collection objects (you must replace "Wrapper" by the name you have specified for the wrapper file):

Object	Example Code
Blocks collection Element types: ▪ Project ▪ Folder ▪ Sequence	Use case: Blocks[0] is a Sequence object. ▪ Without wrapper: `Blocks[0].Execute(Name, Description)`

Object	Example Code

	▪ With wrapper:
	`if (Blocks[0].Type == Wrapper.constants.adSequence)`
	`    SequenceObj = Wrapper.IADSequence(Blocks[0])`
	`    SequenceObj.Execute(Name, Description)`

DataObjects collection
Element types:
▪ String
▪ Int
▪ Float
▪ File
	Use case: DataObjects[0] is an Int object.
	▪ Without wrapper:
	`DataObject[0].Value = 3`
	▪ With wrapper:
	`if (DataObjects[0].Type == Wrapper.constants.adMainLibraryInt)`
	`    IntObj = Wrapper.IADInt(DataObjects[0])`
	`    IntObj.Value = 3`

Libraries collection
Element types:
▪ LibraryFolder
▪ CustomLibraryFolder
	Use case: Libraries[0] is a LibraryFolder object.
	▪ Without wrapper:
	`Name = Libraries[0].Name`
	▪ With wrapper:
	`if (Libraries[0].Type == Wrapper.constants.adLibraryFolder)`
	`    LibFolderObj = Wrapper.IADLibFolder(Libaries[0])`
	`    Name = LibFolderObj.Name`

Type casting

Type casting must also be done if the object is not referenced as an element of a collection. For example:

```
IntTemplateObj = MainLibObj.DataObjects.Item("Int")
IntObj = ProjObj.DataObjects.Create(IntTemplateObj)
# Type casting
IntObj = TAMAutomation.IADInt(IntObj)
IntObj.Value = 0
```

Using Constants in a C# Application

Library reference

If you want to use the constants of the API in a C# application, you must add the type library information as a reference to your C# project. The instructions for adding a reference depend on the software that is used.

For example, if you use Visual Studio®, you must select the **Add Reference** command from the **Project** menu.

Using constants

To use a constant from the API in your application, you must enter the required constant in dot notation. For example, to specify center alignment for the logo in your report, you must use the following command:

```
ReportCfgObject.LogoAlignment = TAMAUTOMATIONLib.adCenter
```

Further use cases

Here are some examples showing the differences when you use a library reference.

Use Case	Example Code
Creating a COM server	▪ Without library reference: `dynamic AdApp;` `System.Type tam = System.Type.GetTypeFromProgID("AutomationDesk.TAM");` `AdApp = System.Activator.CreateInstance(tam);` ▪ With library reference: `IADApplication1 AdApp = null;` `System.Type tam = System.Type.GetTypeFromProgID("AutomationDesk.TAM");` `AdApp = (IADApplication1)Activator.CreateInstance(tam);`
Creating objects	▪ Without library reference: `dynamic AdProject;` `AdProject = AdApp.Projects[0];` ▪ With library reference: `IADProject1 AdProject;` `AdProject = (IADProject1)AdApp.Projects[0];`
Casting objects	▪ Without library reference: `dynamic AdBlock;` `AdBlock = AdProject.SubBlocks[0];` ▪ With library reference: `IADFolder AdFolder;` `Var block = AdProject.SubBlocks[0];` `If (block.Type == TAMAutomationLib.adFolder) {` `AdFolder = (IADFolder)block;` `}`

Using Constants in a Visual Basic Script

Library reference

If you want to use the constants of the API in a Visual Basic script, you must add the type library information as reference to your Visual Basic project. The instructions for adding a reference depend on the software used.

For example:

- Using Visual Studio, you must select the **Add Reference** command from the **Project** menu.
- Using the Visual Basic editor from Excel, you must select the **References** command from the **Tools** menu. You always must select "TAMAutomation <VersionNumber> Type Library" from the list of available references to make the TAMAUTOMATIONLib available in your Visual Basic script.

Using constants

To use a constant from the API in your script, you must enter the required constant in dot notation. For example, to specify the output format of a report as PDF, you must use the following command:

```
Set ReportCfgObject.ReportType = TAMAUTOMATIONLib.adPDF
```

Further use cases

Here are some examples showing the differences when you use a library reference.

Use Case	Example Code

Creating a COM server	▪ Without library reference:
	`Dim ADApp As Object`
	`Set ADApp = CreateObject("AutomationDesk.TAM")`
	▪ With library reference:
	`Dim ADApp As TAMAUTOMATIONLib.Application`
	`Set ADApp = New Application`

Creating objects	▪ Without library reference:
	`Dim ADProject As Object`
	`Set ADProject = ADApp.Project`
	▪ With library reference:
	`Dim ADProject As TAMAUTOMATIONLib.Project`
	`Set ADProject = ADApp.Project`

Casting objects	▪ Without library reference:
	Not required
	▪ With library reference:
	`' Define the folder's variables`
	`Dim ADFolder As TAMAUTOMATIONLib.Folder`
	`Dim ADLibraryFolder As TAMAUTOMATIONLib.LibraryFolder`
	`' Do something and get a folder`
	`'...`
	`' Determine the type of the folder and set it`
	`If ADFolder.Type = TAMAUTOMATIONLib.adLibraryFolder Then`
	`    Set ADLibraryFolder = ADFolder`
	`EndIf`

Application Examples

Introduction

The AutomationDesk installation includes several demo scripts, for example, for integrating the API commands in a custom user interface application.

You can find the latest demo scripts in `<DocumentsFolder>\API`.

Where to go from here

Information in this section

Information in other sections

How to Work with the AutomationDesk API Demos

Objective

AutomationDesk provides source code examples that demonstrate use cases for the AutomationDesk API in Python, C# and Visual Basic.

Possible methods

The integrated development environment (IDE) to be used depends on the demo's programming language:

- Python

  Each use case is implemented as a Python module in a separate PY file. You can edit and execute the Python sources via the Python interpreter for Windows that is installed together with AutomationDesk. Refer to Method 1.
- C#

  Each use case is implemented as a source in a CS file. You can edit the sources in any editor, but before you can execute the changed code, you must compile the source files and link them to an executable, for example, by using Microsoft Visual Studio®. Refer to Method 2.

  Tip

  In the demo folder, you find the **ComApiDemo** executable which is built from the demo sources.
- Visual Basic

  A demonstration for parameterizing and executing an AutomationDesk sequence from a Microsoft Excel workbook via Visual Basic is provided in

`<DocumentsFolder>\API\ExcelVBA\AUDCOM.xls`. For more information, refer to the `readme.txt` file in the same folder.

You can work with the AutomationDesk API demo in Visual Basic via Microsoft Visual Basic for Applications, which is available in Excel. For more information, refer to the Microsoft Excel documentation.

Method 1

To work with the AutomationDesk API demos in Python

1. From the Windows Start menu, choose All Programs – Python 3.6 – PythonWin to open the interpreter.
2. From the menu bar, choose File – Open and specify the demo script to be opened.

   For information on where to find a demo script that relates to the task you want to automate, refer to Overview of the AutomationDesk API Demos on page 75.
3. From the menu bar, choose File – Run to execute a demo script.

   Alternatively, you can double-click the PY file in the File Explorer.

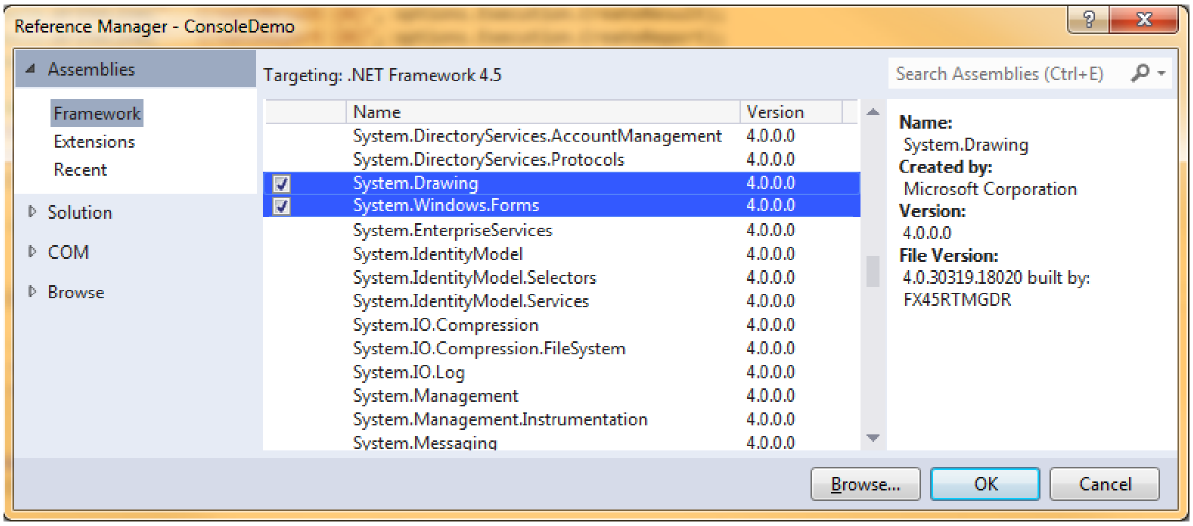
Method 2

To work with the AutomationDesk API demos in C#

1. Create a new Visual Studio project by selecting Visual C# – Windows – Console Application

   Note

   At the top of the New Project dialog, select .NET Framework 4.5.
2. In the Solution Explorer, click Add Reference from the References entry's context menu. The Reference Manager dialog opens.
3. Go to the Assemblies – Framework page and add the following assemblies:
   - System.Drawing
   - System.Windows.Forms



Reference Manager - ConsoleDemo

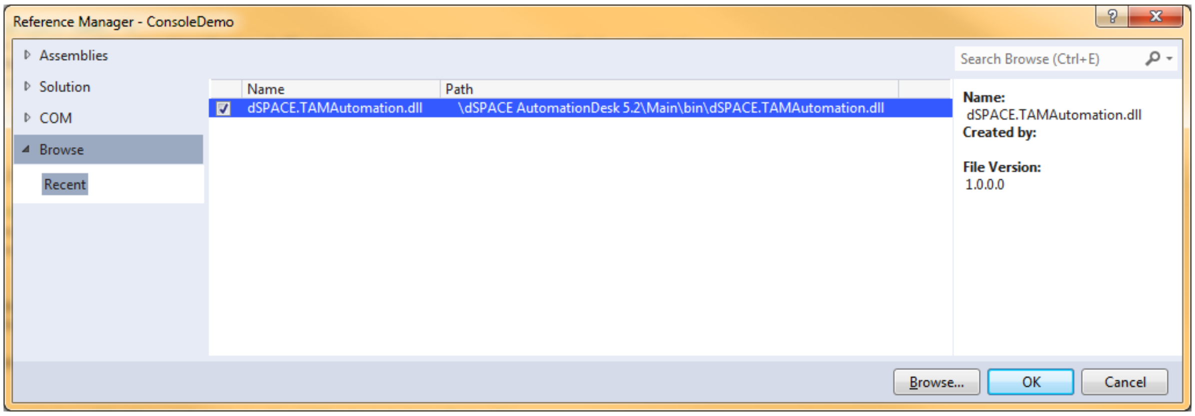
Targeting: .NET Framework 4.5

	Name	Version
	System.DirectoryServices.AccountManagement	4.0.0.0
	System.DirectoryServices.Protocols	4.0.0.0
✓	System.Drawing	4.0.0.0
✓	System.Windows.Forms	4.0.0.0
	System.EnterpriseServices	4.0.0.0
	System.IdentityModel	4.0.0.0
	System.IdentityModel.Selectors	4.0.0.0
	System.IdentityModel.Services	4.0.0.0
	System.IO.Compression	4.0.0.0
	System.IO.Compression.FileSystem	4.0.0.0
	System.IO.Log	4.0.0.0
	System.Management	4.0.0.0
	System.Management.Instrumentation	4.0.0.0
	System.Messaging	4.0.0.0

Name:
System.Drawing
Created by:
Microsoft Corporation
Version:
4.0.0.0
File Version:
4.0.30319.18020 built by: FX45RTMGDR

4 Go to the Browse – Recent page and add the following DLL file, which is located in `<InstallationPath>\Main\bin`:

- dSPACE.TAMAutomation.dll



Reference Manager - ConsoleDemo

	Name	Path
✓	dSPACE.TAMAutomation.dll	\dSPACE AutomationDesk 5.2\Main\bin\dSPACE.TAMAutomation.dll

Name:
dSPACE.TAMAutomation.dll
Created by:

File Version:
1.0.0.0

5 In the Solution Explorer, right-click the `Program.cs` file and delete it.

6 From the context menu of your project, choose Add — Existing Item and add all CS files from the `<DocumentsFolder>\API\Csharp` folder to your project.

7 From the context of your Visual Studio project, choose Build to compile and link the executable of the C# demos.

8 In the File Explorer, double-click the generated executable to start the demo.

Result

You opened the AutomationDesk demos in an IDE that lets you edit and execute the provided sample sources.

To execute a specific Python example, open the related source file in the IDE and run it. To execute a specific C# example, start the generated executable and choose the related example from the menu.

Related topics

Basics

Overview of the AutomationDesk API Demos

Introduction

You find code examples for automating tasks via the AutomationDesk API in the demo sources.

Code examples

AutomationDesk provides source code examples in Python and C# that demonstrate the use of the AutomationDesk API for automating the following tasks.

Creating a project This example contains source code for creating a project with folders, sequences, and data objects, such as Int, Float, and String.

Language	Where to Find
Python	`<DocumentsFolder>\API\Scripting_Python\CreateProject.py`
C#	`<DocumentsFolder>\API\Csharp\CreateProject.cs`

To execute the C# code, start the C# demo executable. In the displayed menu, enter `-cp`. Refer to Executing C# demos on page 77.

Working with Main Library data objects This example contains source code for working with data objects of the Main Library, such as Tuple, List, Dictionary, Variant, and DataContainer.

Language	Where to Find
Python	`<DocumentsFolder>\API\Scripting_Python\MainLibrary.py`
C#	`<DocumentsFolder>\API\Csharp\MainLibrary.cs`

To execute the C# code, start the C# demo executable. In the displayed menu, enter `-ml`. Refer to Executing C# demos on page 77.

Working with custom libraries This example contains source code for creating a custom library with folders and templates for data objects and sequences.

Language	Where to Find
Python	`<DocumentsFolder>\API\Scripting_Python\CreateCustomLibrary.py`
C#	`<DocumentsFolder>\API\Csharp\CustomLib.cs`

To execute the C# code, start the C# demo executable. In the displayed menu, enter `-cl`. Refer to Executing C# demos on page 77.

Configuring a project's reporting This example shows you how to change the text color in the Report Library demo project.

Language	Where to Find
Python	`<DocumentsFolder>\API\Scripting_Python\ReportLibary.py`

Executing a project This example contains source code for importing a project from a ZIP file, executing its sequences, and checking its results.

Language	Where to Find
Python	`<DocumentsFolder>\API\Scripting_Python\ExecuteProject.py`
C#	`<DocumentsFolder>\API\Csharp\ExecuteProject.cs`

To execute the C# code, start the C# demo executable. In the displayed menu, enter `-ep`. Refer to Executing C# demos on page 77.

Terminating the execution of a project via an event This example contains an event handler method that is invoked each time a specific events occurs.

The **`OnShouldExecutionBeStopped`** method is implemented to terminate the execution of a demo project when a condition is fulfilled. The related execution event occurs each time AutomationDesk begins to execute an automation block.

You can use such a mechanism to react to AutomationDesk state changes or value modifications. For details, refer to Events in Alphabetical Order on page 481.

Language	Where to Find
Python	`<DocumentsFolder>\API\Scripting_Python\ExecuteProject_StopEvent.py`
C#	`<DocumentsFolder>\API\Csharp\StopExecution.cs`

If you want to execute the Python code after using an AutomationDesk API Python wrapper, that was generated via the **COM Makepy utility**, delete the contents of `<PythonInstallationPath>Lib\site-packages\win32com\gen_py`.

To execute the C# code, start the C# demo executable. In the displayed menu, enter `-se`. Refer to Executing C# demos on page 77.

Evaluating Signal data objects This example contains source code for working with Signal data objects and for evaluating signals. The demo project for the Evaluation library is used to execute basic evaluation operations.

Language	Where to Find
Python	`<DocumentsFolder>\API\Scripting_Python\Evaluation.py`

Accessing MATLAB This example contains source code for working with MATLAB and MATFile data objects. Data objects of the demo project for the MATLAB Access library are modified and the project is executed.

Language	Where to Find
Python	`<DocumentsFolder>\API\Scripting_Python\MATLABAccess.py`

Using Remote Calibration (COM) This example contains source code for working with Calibration (COM) library-specific data objects, such as System, Project, and LogicalLink. Data objects of the demo project for the Remote Calibration (COM) library are modified and the project is executed.

Language	Where to Find
Python	`<DocumentsFolder>\API\Scripting_Python\RemoteCalibrationCom.py`

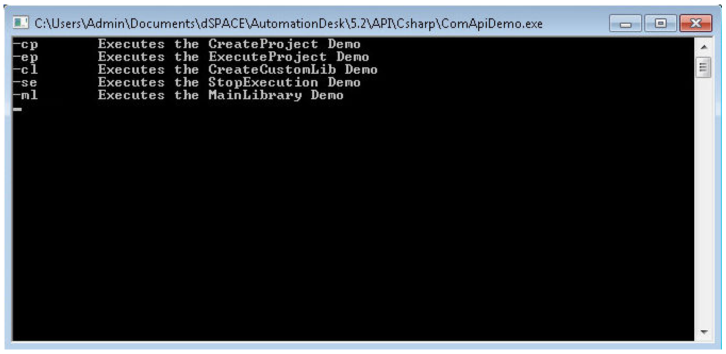
Using Remote Diagnostics (COM) This example contains source code for working with Diagnostics (COM) library-specific data objects, such as System, Project, and VehicleInformation. Data objects of the demo project for the Remote Diagnostics (COM) library are modified and the project is executed.

Language	Where to Find
Python	`<DocumentsFolder>\API\Scripting_Python\ControlDeskRemoteDiagnosticsCom.py`

Executing C# demos

In the demo folder, you find the ComApiDemo executable, which is built from the demo sources.

When you run the executable, a menu opens for you to specify the demo to be performed by entering the related option.


C:\Users\Admin\Documents\dSPACE\AutomationDesk\5.2\API\Csharp\ComApiDemo.exe

```
-cp       Executes the CreateProject Demo
-ep       Executes the ExecuteProject Demo
-cl       Executes the CreateCustomLib Demo
-se       Executes the StopExecution Demo
-ml       Executes the MainLibrary Demo
_
```

You can close the window by entering `quit`.

Working with a custom user interface

For a code example that implements a customized user interface for AutomationDesk via the API, refer to the dSPACE Test Automation Software Support Center at http://www.dspace.com/go/audoperatordemo.

Related topics

Basics

Translating Code Into Other Programming Languages.. 78

Translating Code Into Other Programming Languages

Introduction

All code examples in this documentation are written in Python. You can translate these examples into other programming languages.

Comparison of typical code sequences

The main differences between the languages are shown in the following table. With these typical code sequences, you should be able to translate the Python examples in this documentation into the language of your choice.

Code Sequence	Python	C#	Visual Basic
Comment	`# This is a comment`	`// This is a comment`	`' This is a comment`
Line continuation	`LongFunctionName(\` `Parameter)`	`LongFunctionName(` `Parameter);`	`LongFunctionName( _` `Parameter)`
Control structure	`if A == B and C == D:` `...`	`if (A == B & C == D) { ... }`	`If A = B Then` `If C = D Then` `...` `End If` `End If`
Creation	`AudObj =` `win32com.client.Dispatch\` `("AutomationDesk.TAM")`	`System.Type type =` `GetTypeFromProgID("AutomationDesk.TAM");` `AudObj =` `System.Activator.CreateInstance(type);`	`Set AudObj =` `CreateObject( _` `"AutomationDesk.TAM")`
Destruction	`AudObj = None`	`AudObj = null;`	`Set AudObj = Nothing`
Calling methods without parameters	`ProjObj.Save()`	`ProjObj.Save();`	`ProjObj.Save`
Collections	Indexing: `Blocks[0]` Loop: `for Element in Collection:` `...`	Indexing: `Blocks[0]` Loop: `foreach(element in Collection)` `{ ... }`	Indexing: `Blocks(0)` Loop: `For Each Element In` `Collection` `...`

Code Sequence	**Python**	**C#**	**Visual Basic**

Array handling	`MyArray = (0, 2)`	`object[] array = new object[2];`
		`array[0] = 1;`
		`array[1] = 2;`
			`Dim MyArray(2) As Variant`
			`MyArray(0) = 0`
			`MyArray(1) = 2`

Related topics

Basics

Overview of the AutomationDesk API.. 32

HowTos

How to Work with the AutomationDesk API Demos.. 72

Reference Information

Where to go from here

Information in this section

Objects of the AutomationDesk COM API

Introduction

The AutomationDesk COM API provides all the relevant objects that are required to work with projects, sequences and libraries. The object overview describes the objects of the basic interface and the supported built-in libraries.

Where to go from here

Information in this section

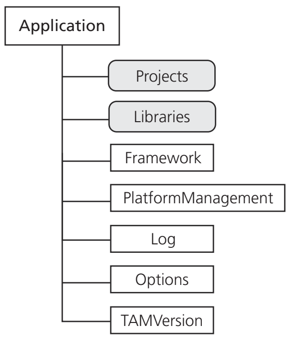
Basic Interface

Where to go from here

Information in this section

Application

Object



Syntax

No direct creation.

Purpose

To create a COM server for automating access to AutomationDesk.

Description

If you start the COM server, for example, by using the dispatch function, you instantiate automatically the Application object. All other objects of the API are properties of the Application object.

Properties

The Application object definition contains the following properties:

Property	Purpose
Libraries (Property) on page 349	To get the Libraries collection of the application.
Options (Property) on page 365	To get the Options object of the application.
Projects (Property) on page 372	To get the Projects collection of the application.
TAMVersion (Property) on page 402	To get the version of the object model used.

Methods

None

Events

The Application object definition contains the following events:

Event	Purpose
OnError on page 483	To react to an error of the application.
OnProjectActivate on page 487	To react to project activation.
OnProjectClose on page 488	To react to a project being closed.

Event	Purpose
OnProjectClosed on page 488	To react to a closed project.
OnProjectCreate on page 489	To react to a project being created.
OnProjectCreated on page 490	To react to a created project.
OnProjectOpen on page 491	To react to a project being opened.
OnProjectOpened on page 491	To react to an opened project.
OnProjectSave on page 492	To react to a project being saved.
OnProjectSaved on page 493	To react to a saved project.
OnWrite on page 496	To react to an output by the application.

Related topics

References

Application1

Syntax

No direct creation.

Purpose

To create a COM server for automating access to AutomationDesk.

Description

The Application1 object is based on the interface definition of the Application object. It additionally provides the property for the display mode of AutomationDesk and a method to close AutomationDesk.

Properties

The Application1 object definition contains the following properties:

Property	Purpose
Libraries (Property) on page 349	To get the Libraries collection of the application.
Options (Property) on page 365	To get the Options object of the application.
Projects (Property) on page 372	To get the Projects collection of the application.
TAMVersion (Property) on page 402	To get the version of the object model used.
Visible on page 409	To set or get the display mode of AutomationDesk.

Methods

The Application1 object definition contains the following methods:

Method	Purpose
Quit on page 457	To close AutomationDesk.

Events

The Application1 object definition contains the following events:

Event	Purpose
OnError on page 483	To react to an error of the application.
OnProjectActivate on page 487	To react to project activation.
OnProjectClose on page 488	To react to a project being closed.
OnProjectClosed on page 488	To react to a closed project.
OnProjectCreate on page 489	To react to a project being created.
OnProjectCreated on page 490	To react to a created project.
OnProjectOpen on page 491	To react to a project being opened.
OnProjectOpened on page 491	To react to an opened project.
OnProjectSave on page 492	To react to a project being saved.
OnProjectSaved on page 493	To react to a saved project.
OnWrite on page 496	To react to an output by the application.

Related topics

References

Application2

Syntax

No direct creation.

Purpose

To create a COM server for automating access to AutomationDesk.

Description

The Application2 object is based on the interface definition of the Application1 object. Additionally, it provides a property to get access to the message logging.

If you start the COM server, for example, by using the dispatch function, you automatically instantiate the Application2 object. All other objects of the API are properties of the Application object.

Properties

The Application2 object definition contains the following properties:

Property	Purpose
Framework (Property) on page 325	To get the Framework object of the application.
Libraries (Property) on page 349	To get the Libraries collection of the application.
Log (Property) on page 350	To get the Log object of the application.
Options (Property) on page 365	To get the Options object of the application.
PlatformManagement on page 370	To get the dispatch object for platform management.
Projects (Property) on page 372	To get the Projects collection of the application.
Selection (Property) on page 374	To get the collection of selected elements.
TAMVersion (Property) on page 402	To get the version of the object model used.
Visible on page 409	To set or get the display mode of AutomationDesk.

Methods

The Application2 object definition contains the following methods:

Method	Purpose
Quit on page 457	To close AutomationDesk.

Events

The Application2 object definition contains the following events:

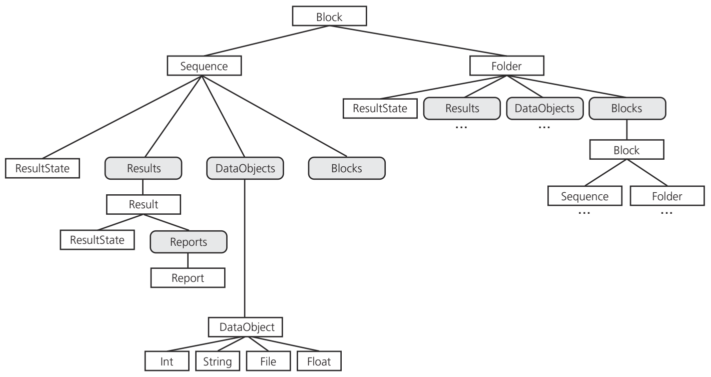
Event	Purpose
OnError on page 483	To react to an error of the application.
OnProjectActivate on page 487	To react to project activation.
OnProjectClose on page 488	To react to a project being closed.
OnProjectClosed on page 488	To react to a closed project.
OnProjectCreate on page 489	To react to a project being created.
OnProjectCreated on page 490	To react to a created project.
OnProjectOpen on page 491	To react to a project being opened.
OnProjectOpened on page 491	To react to an opened project.
OnProjectSave on page 492	To react to a project being saved.
OnProjectSaved on page 493	To react to a saved project.
OnWrite on page 496	To react to an output by the application.

Related topics

References

Block

Object



Syntax

No direct creation.

Purpose

To handle a specific folder or sequence.

Description

A Block object gives you access to folder and sequence elements in projects and custom libraries. You can use such blocks to build a hierarchical structured project tree. All Block objects are managed by the Blocks collection (refer to Blocks on page 92).

Properties

The Block object definition contains the following properties:

Property	Purpose
Author on page 290	To set or get the name of the person who created the object.
CreationDate on page 314	To get the date the object was created.
Description on page 317	To set or get the description of the object.
HierarchyName on page 328	To get the hierarchy path of the object.
Hyperlink on page 329	To get the AutomationDesk hyperlink of the object.
IconPath on page 332	To get the path to the symbol representing the object type.

Property	Purpose
IsCollapsed on page 340	To set or get the option for collapsing the object's structure in the project tree.
IsEnabled on page 342	To set or get the enable state of an element.
IsLibraryElement on page 345	To check whether the object is a library object.
ModificationDate on page 358	To get the date of the last modification of the object.
Name on page 360	To set or get the name of the object.
Parent on page 368	To get the parent object of the specified object.
Protected on page 373	To check whether the object is protected.
ResultLevel on page 382	To set or get the result level of the object.
StateIconPath on page 394	To get the path to the symbol representing the state of the object.
Type on page 402	To get the type of the specified object.

Methods

The Block object definition contains the following method:

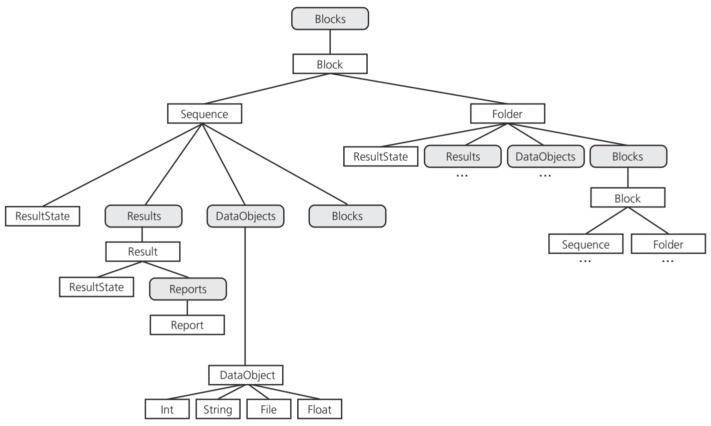
Method	Purpose
Highlight on page 441	To highlight the object's element.

Events

None

Blocks

Object



Syntax

No direct creation.

Purpose

To create and handle folders and sequences.

Description

The Blocks object contains access to the Blocks collection. You can create and manage block objects of folder and sequence element type. You can use such blocks to build a hierarchical project tree.

Properties

The Blocks object definition contains the following properties:

Property	Purpose
Count on page 312	To get the number of the object's instances.
Names on page 361	To get the child element names of a collection.
Parent on page 368	To get the parent of the specified object.

Methods

The Blocks object definition contains the following methods:

Method	Purpose
Copy on page 423	To create a copy of the specified object at the specified position.
Create on page 425	To create a new object based on its collection object.
FindElement on page 437	To get the object of the element that is specified by its hierarchy path.
Item on page 449	To get a specific item of the specified object.
Move on page 454	To move an object to the specified position.
Remove on page 459	To delete an object.
RemoveAll on page 460	To delete all created child elements of a collection.

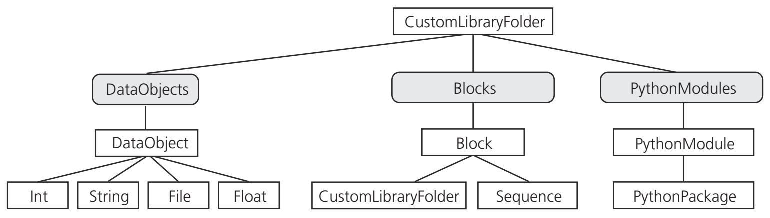
Events

The Blocks object definition contains the following events:

Event	Purpose
OnAdd on page 482	To react to a folder or sequence being created.
OnRemove on page 494	To react to a folder or sequence being deleted.

CustomLibraryFolder

Object



Syntax

No direct creation.

Purpose

To handle a custom library folder.

Description

The CustomLibraryFolder object lets you access the custom library. If you already created custom sequence templates in your custom library, you can use them to add custom sequences to your project. You can also create new templates by adding a sequence to the custom library.

Properties

The CustomLibraryFolder object definition contains the following properties:

Property	Purpose
Author on page 290	To set or get the name of the person who created the object.
CreationDate on page 314	To get the date the object was created.
Description on page 317	To set or get the description of the object.
HierarchyName on page 328	To get the hierarchy path of the object.
Hyperlink on page 329	To get the AutomationDesk hyperlink of the object.
IconPath on page 332	To get the path to the symbol representing the object type.
IsCollapsed on page 340	To set or get the option for collapsing the object's structure in the project tree.
IsLibraryElement on page 345	To check whether the object is a library object.
ModificationDate on page 358	To get the date of the last modification of the object.
Name on page 360	To set or get the name of the object.
Parent on page 368	To get the parent of the specified object.
Protected on page 373	To check whether the object is protected.
ResultLevel on page 382	To set or get the result level of the specified object.
StateIconPath on page 394	To get the path to the symbol representing the state of the object.
SubBlocks on page 399	To get the subblocks of the specified object.
Type on page 402	To get the type of the specified object.

Methods

The CustomLibraryFolder object definition contains the following methods:

Method	Purpose
Highlight on page 441	To highlight the object's element.
Save on page 470	To save the custom library folder.
Synchronize on page 476	To synchronize the sequences with the custom library templates.

Events

None

Related topics

References

CustomLibraryFolder1

Syntax

No direct creation.

Purpose

To handle a custom library folder.

Description

The CustomLibraryFolder1 object is based on the interface definition of the CustomLibraryFolder object. It additionally provides a method for exporting a custom library to a file.

Properties

The CustomLibraryFolder1 object definition contains the following properties:

Property	Purpose
Author on page 290	To set or get the name of the person who created the object.
CreationDate on page 314	To get the date the object was created.
Description on page 317	To set or get the description of the object.
HierarchyName on page 328	To get the hierarchy path of the object.
Hyperlink on page 329	To get the AutomationDesk hyperlink of the object.
IconPath on page 332	To get the path to the symbol representing the object type.
IsCollapsed on page 340	To set or get the option for collapsing the object's structure in the project tree.
IsLibraryElement on page 345	To check whether the object is a library object.

Property	Purpose
ModificationDate on page 358	To get the date of the last modification of the object.
Name on page 360	To set or get the name of the object.
Parent on page 368	To get the parent of the specified object.
Protected on page 373	To check whether the object is protected.
ResultLevel on page 382	To set or get the result level of the specified object.
StateIconPath on page 394	To get the path to the symbol representing the state of the object.
SubBlocks on page 399	To get the subblocks of the specified object.
Type on page 402	To get the type of the specified object.

Methods

The CustomLibraryFolder1 object definition contains the following methods:

Method	Purpose
Close on page 421	To close a custom library.
ExportFile on page 435	To export the custom library.
Highlight on page 441	To highlight the object's element.
Save on page 470	To save the custom library.
SaveAs on page 471	To save the custom library with a new name.
Synchronize on page 476	To synchronize the sequences with the custom library templates.

Events

None

Related topics

References

CustomLibraryFolder..94
CustomLibraryFolder2..96

CustomLibraryFolder2

Syntax

No direct creation.

Purpose

To handle a custom library folder.

Description

The CustomLibraryFolder2 object is based on the interface definition of the CustomLibraryFolder1 object. In addition to the features of the first object, it provides a method for creating a subfolder in a custom library folder. It also provides properties to get the path to the file where the custom library is stored and to access the Python modules and packages that are added to the custom library folder.

Via the Synect property, you can configure the synchronization with SYNECT.

Properties

The CustomLibraryFolder2 object definition contains the following properties:

Property	Purpose
Author on page 290	To set or get the name of the person who created the object.
CreationDate on page 314	To get the date the object was created.
DataObjects (Property) on page 316	To get the collection object for accessing a data object.
Description on page 317	To set or get the description of the object.
HierarchyName on page 328	To get the hierarchy path of the object.
Hyperlink on page 329	To get the AutomationDesk hyperlink of the object.
IconPath on page 332	To get the path to the symbol representing the object type.
IsCollapsed on page 340	To set or get the option for collapsing the object's structure in the project tree.
IsLibraryElement on page 345	To check whether the object is a library object.
ModificationDate on page 358	To get the date of the last modification of the object.
Name on page 360	To set or get the name of the object.
Parent on page 368	To get the parent of the specified object.
Path on page 369	To get the path that contains the specified object.
Protected on page 373	To check whether the object is protected.
PythonModules (Property) on page 374	To get the collection object for accessing a Python module or package.
ResultLevel on page 382	To set or get the result level of the specified object.
StateIconPath on page 394	To get the path to the symbol representing the state of the object.
SubBlocks on page 399	To get the subblocks of the specified object.
Synect (Property) on page 400	To get the Synect object.
Type on page 402	To get the type of the specified object.

Methods

The CustomLibraryFolder2 object definition contains the following methods:

Method	Purpose
ClearValues on page 419	To recursively clear the values of all contained output data objects and/or local data objects.
Close on page 421	To close a custom library.
ExportFile on page 435	To export the custom library.
Highlight on page 441	To highlight the object's element.
Save on page 470	To save the custom library.

Method	Purpose
SaveAs on page 471	To save the custom library with a new name.
Synchronize on page 476	To synchronize the sequences with the custom library templates.
CreateSubFolder on page 428	To create a subfolder in the CustomLibraryFolder.

Events

None

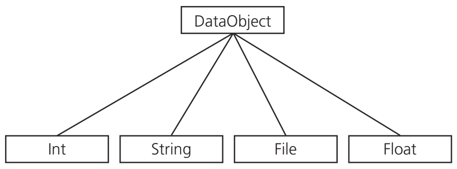
Related topics

References

CustomLibraryFolder.......... 94
CustomLibraryFolder1.......... 95

DataObject

Object



Syntax

No direct creation.

Purpose

To handle a specific data object.

Description

A DataObject object is an element that stores data of different types, for example, file, float, int, and string. The data objects cannot be executed, so they have no behavior with respect to the execution. The DataObject object can be either a library or a project element. It can be created as a subelement of a project, folder, sequence, or custom library folder. The DataObject objects are managed by the DataObjects collection.

Properties

The DataObject object definition contains the following properties:

Property	Purpose
HierarchyName on page 328	To get the hierarchy path of the object.
Hyperlink on page 329	To get the AutomationDesk hyperlink of the object.

Property	Purpose
IconPath on page 332	To get the path to the symbol representing the object type.
InOutState on page 338	To set or get the data direction of a data object.
IsLibraryElement on page 345	To check whether the object is a library object.
Name on page 360	To set or get the name of the object.
Parent on page 368	To get the parent of the specified object.
Protected on page 373	To check whether the object is protected.
ReferenceName on page 378	To set or get the name of the referenced object.
ResultLevel on page 382	To set or get the result level of the object.
StateIconPath on page 394	To get the path to the symbol representing the state of the object.
Type on page 402	To set or get the type of the specified object.

Methods

The DataObject object definition contains the following method:

Method	Purpose
Highlight on page 441	To highlight the object's element.

Events

None

Related topics

References

DataObject2

Syntax

No direct creation.

Purpose

To handle a specific data object.

Description

The DataObject2 object is based on the interface definition of the DataObject object. It additionally provides the properties for linking the data object to a custom library. Furthermore, it provides the properties to hold the creation and modification time and the author of the data object.

Properties

The DataObject2 object definition contains the following properties:

Property	Purpose
Author on page 290	To set or get the name of the person who created the object.
CreationDate on page 314	To get the date the object is created.
Description on page 317	To set or get the description of the object.
HasLibraryLink on page 327	To check whether the data object is linked to the custom library.
HierarchyName on page 328	To get the hierarchy path of the object.
Hyperlink on page 329	To get the AutomationDesk hyperlink of the object.
IconPath on page 332	To get the path to the symbol representing the object type.
InOutState on page 338	To set or get the data direction of a data object.
IsLibraryElement on page 345	To check whether the object is a library object.
LibraryLink on page 349	To get the library link of the data object.
ModificationDate on page 358	To get the date of the last object modification.
Name on page 360	To set or get the name of the object.
Parent on page 368	To get the parent of the specified object.
Protected on page 373	To check whether the object is protected.
ReferenceName on page 378	To set or get the name of the referenced object.
ResultLevel on page 382	To set or get the result level of the object.
StateIconPath on page 394	To get the path to the symbol representing the state of the object.
Type on page 402	To set or get the type of the specified object.

Methods

The DataObject2 object definition contains the following method:

Method	Purpose
Highlight on page 441	To highlight the object's element.

Events

None

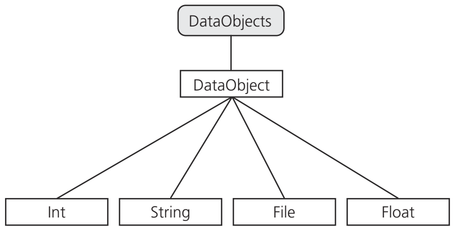
Related topics

References

DataObject.. 98

DataObjects (Object)

Object



Syntax

No direct creation.

Purpose

To create and handle data objects.

Description

The DataObjects collection provides access to the data objects. You can create and manage data objects. A data object cannot be executed, so it has no behavior with respect to the execution.

Properties

The DataObjects object definition contains the following properties:

Property	Purpose
Count on page 312	To get the number of object instances.
Names on page 361	To get the child element names of a collection.
Parent on page 368	To get the parent of the specified object.

Methods

The DataObjects object definition contains the following methods:

Method	Purpose
Copy on page 423	To create a copy of the specified object at the specified position.
Create on page 425	To create a new object based on its collection object.
FindElement on page 437	To get the object of the element that is specified by its hierarchy path.
Item on page 449	To get a specific item of the specified object.
Move on page 454	To move an object to the specified position.
Remove on page 459	To delete an object.
RemoveAll on page 460	To delete all created child elements of a collection.

Events

The DataObjects object definition contains the following events:

Event	Purpose
OnAdd on page 482	To react to a data object being created.
OnRemove on page 494	To react to a data object being deleted.

ExecutionConfiguration

Object

ExecutionConfiguration

Syntax

No direct creation.

Purpose

To configure the execution options.

Description

The ExecutionConfiguration object gives you access to the execution options. You can specify the record depth for the result, and also whether a result should be logged and a report should be generated after execution.

Properties

The ExecutionConfiguration object definition contains the following properties:

Property	Purpose
CreateReport on page 313	To set or get the option for creating a report directly after execution.
CreateResult on page 313	To set or get the option for logging the result of the execution.
Parent on page 368	To get the parent of the specified object.
RecordDepth on page 377	To set or get the record depth for the result.

Methods

None

Events

None

Related topics

References

ExecutionConfiguration1

Syntax

No direct creation.

Purpose

To configure the execution options.

Description

The ExecutionConfiguration1 object is based on the interface definition of the ExecutionConfiguration object and contains all methods/properties available in ExecutionConfiguration. ExecutionConfiguration can be accessed from the Execution property of the Options (Object) object. If a COM wrapper is used, a type casting is needed to access the StopExecution method. That means the ExecutionConfiguration object returned from the Options object has to be typecasted as ExecutionConfiguration1 object.

Properties

The ExecutionConfiguration object definition contains the following properties:

Property	Purpose
CreateReport on page 313	To set or get the option for creating a report directly after execution.
CreateResult on page 313	To set or get the option for logging the result of the execution.
Parent on page 368	To get the parent of the specified object.
RecordDepth on page 377	To set or get the record depth for the result.

Methods

The Folder object definition contains the following methods:

Method	Purpose
StopExecution on page 475	To automatically stop a running execution.

Events

None

Related topics

References

ExecutionConfiguration2

Syntax

No direct creation.

Purpose

To configure the execution options.

Description

You can access ExecutionConfiguration2 via the Execution property of the Options (Object) object. If you use a COM wrapper, the ExecutionConfiguration object returned from the Options object has to be typecast as the ExecutionConfiguration2 object.

The ExecutionConfiguration2 object is based on the interface definition of the ExecutionConfiguration1 object and contains all methods/properties available in ExecutionConfiguration. Additionally, ExecutionConfiguration2 provides properties to specify whether to open the Result Browser after the execution has finished and whether to update data object values during the execution.

Properties

The ExecutionConfiguration2 object definition contains the following properties:

Property	Purpose
CreateReport on page 313	To set or get the option for creating a report directly after execution.
CreateResult on page 313	To set or get the option for logging the result of the execution.
DisplayDataObjectValueUpdates on page 318	To set or get the option for updating data object values in the user interface during the execution.
IsExecutionRunning on page 343	To get the status of the execution.
OpenResultBrowser on page 363	To set or get the option for opening the Result Browser after execution.
Parent on page 368	To get the parent of the specified object.
RecordDepth on page 377	To set or get the record depth for the result.

Methods

The Folder object definition contains the following methods:

Method	Purpose
StopExecution on page 475	To automatically stop a running execution.

Events

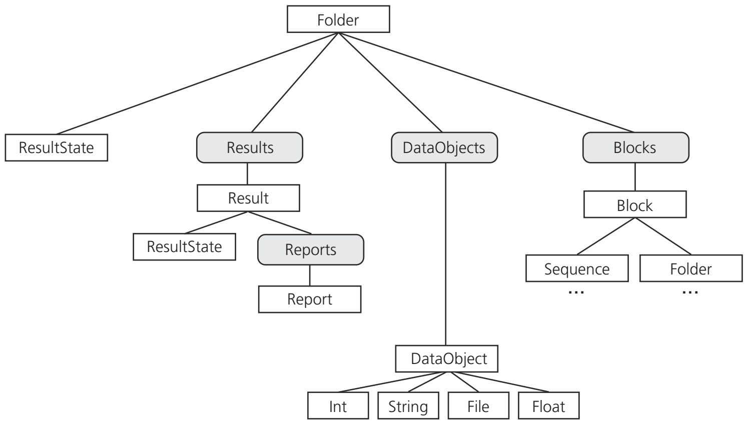
None

Related topics

References

Folder

Object



Syntax

No direct creation.

Purpose

To handle a folder.

Description

The Folder object is part of a project. It can aggregate several other folders, sequences, data objects and results. You can use the Folder objects to build a hierarchical project tree. When the Folder object is executed, all its child elements are executed recursively.

Properties

The Folder object definition contains the following properties:

Property	Purpose
Author on page 290	To set or get the name of the person who created the object.
CreationDate on page 314	To get the date the object was created.
DataObjects (Property) on page 316	To get the collection object for accessing a data object.
Description on page 317	To set or get the description of the object.
HierarchyName on page 328	To get the hierarchy path of the object.
Hyperlink on page 329	To get the AutomationDesk hyperlink of the object.
IconPath on page 332	To get the path to the symbol representing the object type.
IsCollapsed on page 340	To set or get the option for collapsing the object's structure in the project tree.
IsEnabled on page 342	To set or get the enable state of an element.
IsLibraryElement on page 345	To check whether the object is a library object.
ModificationDate on page 358	To get the date of the last modification of the object.
Name on page 360	To set or get the name of the object.
Parent on page 368	To get the parent of the specified object.
Protected on page 373	To check whether the object is protected.
ResultLevel on page 382	To set or get the result level of the specified object.
Results (Property) on page 383	To get the results of the specified object.
ResultState (Property) on page 384	To get the result state of the specified object.
StateIconPath on page 394	To get the path to the symbol representing the state of the object.
SubBlocks on page 399	To get the subblocks of the specified object.
Type on page 402	To get the type of the specified object.

Methods

The Folder object definition contains the following methods:

Method	Purpose
ClearValues on page 419	To recursively clear the values of all contained output data objects and/or local data objects.
Execute on page 433	To execute the sequences of a folder.
Highlight on page 441	To highlight the object's element.
Synchronize on page 476	To synchronize the sequences with the custom library templates.

Events

The Folder object definition contains the following events:

Event	Purpose
OnExecutionFinished on page 484	To react to a finished execution.
OnExecutionProgress on page 484	To react to the progress of an execution.
OnExecutionStarted on page 485	To react to an execution starting.

Event	Purpose
OnModified on page 485	To react to a folder being modified.
OnShouldExecutionBeStopped on page 495	To react to an execution stopping.

Folder1

Syntax

No direct creation.

Purpose

To handle a folder.

Description

The Folder1 object is based on the interface definition of the Folder object. It additionally provides the methods for importing and exporting via XML file. If you use the Import method for a ZIP file, you will import an entire AutomationDesk project.

The Folder object is part of a project. It can aggregate several other folders, sequences, data objects and results. You can use the Folder objects to build a hierarchical project tree. When the Folder object is executed, all its child elements are executed recursively.

Properties

The Folder object definition contains the following properties:

Property	Purpose
Author on page 290	To set or get the name of the person who created the object.
CreationDate on page 314	To get the date the object was created.
DataObjects (Property) on page 316	To get the collection object for accessing a data object.
Description on page 317	To set or get the description of the object.
HierarchyName on page 328	To get the hierarchy path of the object.
Hyperlink on page 329	To get the AutomationDesk hyperlink of the object.
IconPath on page 332	To get the path to the symbol representing the object type.
IsCollapsed on page 340	To set or get the option for collapsing the object's structure in the project tree.
IsEnabled on page 342	To set or get the enable state of an element.
IsLibraryElement on page 345	To check whether the object is a library object.
ModificationDate on page 358	To get the date of the last modification of the object.
Name on page 360	To set or get the name of the object.
Parent on page 368	To get the parent of the specified object.
Protected on page 373	To check whether the object is protected.
ResultLevel on page 382	To set or get the result level of the specified object.

Property	Purpose
Results (Property) on page 383	To get the results of the specified object.
ResultState (Property) on page 384	To get the result state of the specified object.
StateIconPath on page 394	To get the path to the symbol representing the state of the object.
SubBlocks on page 399	To get the subblocks of the specified object.
Type on page 402	To get the type of the specified object.

Methods

The Folder object definition contains the following methods:

Method	Purpose
ClearValues on page 419	To recursively clear the values of all contained output data objects and/or local data objects.
Execute on page 433	To execute the sequences of a folder.
ExportFile on page 435	To export a folder to an XML file.
Highlight on page 441	To highlight the object's element.
ImportFile on page 446	To import a folder or a sequence to the instantiated folder object from an XML file.
Synchronize on page 476	To synchronize the sequences with the custom library templates.

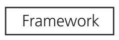
Events

The Folder object definition contains the following events:

Event	Purpose
OnExecutionFinished on page 484	To react to a finished execution.
OnExecutionProgress on page 484	To react to the progress of an execution.
OnExecutionStarted on page 485	To react to an execution starting.
OnModified on page 485	To react to a folder being modified.
OnShouldExecutionBeStopped on page 495	To react to an execution stopping.

Framework (Object)

Object



Syntax

No direct creation.

Purpose

To access the XIL API framework.

Description

The Framework object provides access to the XIL API framework, so you can initialize it, for example.

Properties

The Framework object definition contains the following properties:

Property	Purpose
IsInitialized on page 344	To get whether the XIL API framework is initialized.
Parent on page 368	To get the parent of the specified object.

Methods

The Framework object definition contains the following methods:

Method	Purpose
Init on page 449	To initialize the XIL API framework.
Shutdown on page 474	To shut down the XIL API framework.

Events

None

FrameworkConfiguration

Object

FrameworkConfiguration

Syntax

No direct creation.

Purpose

To configure the XIL API framework.

Description

The FrameworkConfiguration object provides access to the configuration of the XIL API framework.

Properties

The Framework object definition contains the following properties:

Property	Purpose
AvailableImplementations on page 295	To get the list of available XIL API implementations.
ConfigurationFile on page 310	To get or set the path of the XIL API framework configuration file.
Implementation on page 334	To get or set the XIL API implementation to be used.

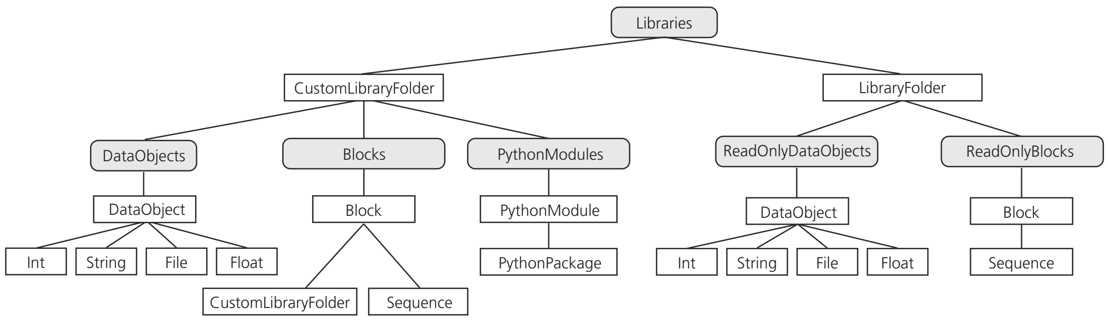
Property	Purpose
InitializeOnStartUp on page 338	To get or set whether to initialize the XIL API framework automatically.
Parent on page 368	To get the parent of the specified object.

Methods None

Events None

Libraries (Object)

Object



Syntax No direct creation.

Purpose To handle the available libraries.

Description The Libraries object contains access to several libraries. .

The Libraries object is the root node of an AutomationDesk library. It provides access to:

- Built-in libraries
- Standard library providing the project elements
- Custom libraries

The library elements are write-protected and cannot be executed.

Properties

The Libraries object definition contains the following properties:

Property	Purpose
Count on page 312	To get the number of the object's instances.
Names on page 361	To get the child element names of a collection.
Parent on page 368	To get the parent of the specified object.

Methods

The Libraries object definition contains the following method:

Method	Purpose
FindElement on page 437	To get the object of the element that is specified by its hierarchy path or its template name.
Item on page 449	To get a specific item of the specified object.

Events

None

Libraries1

Syntax

No direct creation.

Purpose

To handle the available libraries.

Description

The Libraries1 object is based on the interface definition of the Libraries object. The Libraries1 object provides access to the libraries collection, allowing you to create, load, or import a custom library. You can save and close all the custom libraries in the collection.

The Libraries1 object is the root node of an AutomationDesk library. It provides access to:

- Built-in libraries
- Standard library providing the project elements
- Custom libraries

Properties

The Libraries1 object definition contains the following properties:

Property	Purpose
Count on page 312	To get the number of the object's instances.
Names on page 361	To get the child element names of a collection.
Parent on page 368	To get the parent of the specified object.

Methods

The Libraries1 object definition contains the following methods:

Method	Purpose
CloseAll on page 422	To close all custom libraries.
Create on page 425	To create a new custom library.
FindElement on page 437	To get the object of the element that is specified by its hierarchy path or its template name.
Import on page 442	To import a custom library from a ZIP or XML file.
Item on page 449	To get a specific item of the object.
Load on page 451	To load an AutomationDesk custom library.
SaveAll on page 471	To save all opened custom libraries.

Events

None

Libraries2

Syntax

No direct creation.

Purpose

To handle the available libraries.

Description

The Libraries2 object is based on the interface definition of the Libraries1 object. It also provides a property to get the library favorites.

The Libraries2 object is the root node of an AutomationDesk library. It provides access to:

- Built-in libraries
- Standard library providing the project elements
- Custom libraries

Properties

The Libraries2 object definition contains the following properties:

Property	Purpose
Count on page 312	To get the number of the object's instances.
Names on page 361	To get the child element names of a collection.
Parent on page 368	To get the parent of the specified object.
Favorites on page 323	To get the available library favorites.

Methods

The Libraries2 object definition contains the following methods:

Method	Purpose
CloseAll on page 422	To close all custom libraries.
Create on page 425	To create a new custom library.
FindElement on page 437	To get the object of the element that is specified by its hierarchy path or its template name.
Import on page 442	To import a custom library from a ZIP or XML file.
Item on page 449	To get a specific item of the object.
Load on page 451	To load an AutomationDesk custom library.
SaveAll on page 471	To save all opened custom libraries.

Events

None

Libraries3

Syntax

No direct creation.

Purpose

To handle the available libraries.

Description

The Libraries3 object is based on the interface definition of the Libraries2 object. Additionally, it provides a method for opening a custom library from a file.

Files of the following formats are supported:

- ADLX files that contain a custom library that is saved in XML format using AutomationDesk 6.2 or later.
- ADL files that contain a custom library that is saved in a binary legacy format using AutomationDesk 6.1 or earlier.
- ZIP files that contain a custom library that is exported as a compressed archive.

- ALX files that contain a custom library that is exported in legacy XML format using AutomationDesk 6.0 or earlier.
- ADLX files that contain a custom library that is exported in XML format using AutomationDesk 6.1 or later.

Properties

The Libraries3 object definition contains the following properties:

Property	Purpose
Count on page 312	To get the number of the object's instances.
Names on page 361	To get the child element names of a collection.
Parent on page 368	To get the parent of the specified object.
Favorites on page 323	To get the available library favorites.

Methods

The Libraries3 object definition contains the following methods:

Method	Purpose
CloseAll on page 422	To close all custom libraries.
Create on page 425	To create a new custom library.
FindElement on page 437	To get the object of the element that is specified by its hierarchy path or its template name.
Import on page 442	To import a custom library from a ZIP or XML file.
Item on page 449	To get a specific item of the object.
Load on page 451	To load an AutomationDesk custom library.
Open on page 455	To open a library from a file.
SaveAll on page 471	To save all open custom libraries.

Events

None

LibraryFavorites

Syntax

No direct creation.

Purpose

To handle the available library favorites.

Description

The LibraryFavorites object provides methods to export and import the available library favorites to and from an XML file.

Properties

None

Methods

The LibraryFavorites object definition contains the following methods:

Method	Purpose
Export on page 434	To export library favorites to an XML file.
Import on page 442	To import library favorites from an XML file.

Events

None

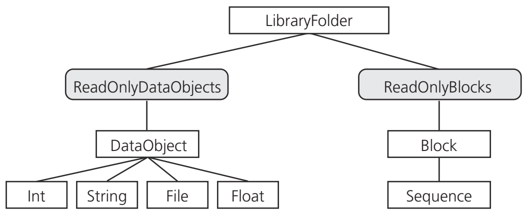
Related topics

References

Library Favorites (AutomationDesk Basic Practices 📖)

LibFolder

Object



Syntax

No direct creation.

Purpose

To handle a specific library.

Description

The LibFolder (LibraryFolder) object is the root node of an AutomationDesk library. It provides access to the built-in libraries and to the Standard library providing the project elements. These libraries are read-only, for example, you cannot modify the library elements.

Properties

The LibFolder object definition contains the following properties:

Property	Purpose
Author on page 290	To set or get the name of the person who created the object.
CreationDate on page 314	To get the date the object was created.
DataObjects (Property) on page 316	To get the collection object for accessing a data object.
Description on page 317	To get the description of the object.
HierarchyName on page 328	To get the hierarchy path of the object.
Hyperlink on page 329	To get the AutomationDesk hyperlink of the object.
IconPath on page 332	To get the path to the symbol representing the object type.
IsCollapsed on page 340	To set or get the option for collapsing the object's structure in the project tree.
IsLibraryElement on page 345	To get the information whether the object is a library element.
ModificationDate on page 358	To get the date of the last object modification.
Name on page 360	To get the name of the object.
Parent on page 368	To get the parent of the specified object.
Protected on page 373	To get the information whether the object is protected.
ResultLevel on page 382	To get the result level of the specified object.
StateIconPath on page 394	To get the path to the symbol representing the state of the object.
SubBlocks on page 399	To get the subblocks of the specified object.
Type on page 402	To get the type of the specified object.

Methods

The LibraryFolder object definition contains the following methods:

Method	Purpose
Highlight on page 441	To highlight the object's element.

Events

None

LibFolder1

Syntax

No direct creation.

Purpose

To handle a specific library.

Description

The LibFolder1 object is based on the interface definition of the LibFolder object. It additionally provides the property for reading and setting the operation mode of a built-in library. By setting the operation mode, you can switch between the online, online recording and offline operation mode.

Properties

The LibFolder1 object definition contains the following properties:

Property	Purpose
Author on page 290	To set or get the name of the person who created the object.
CreationDate on page 314	To get the date the object was created.
DataObjects (Property) on page 316	To get the collection object for accessing a data object.
Description on page 317	To get the description of the object.
HierarchyName on page 328	To get the hierarchy path of the object.
Hyperlink on page 329	To get the AutomationDesk hyperlink of the object.
IconPath on page 332	To get the path to the symbol representing the object type.
IsCollapsed on page 340	To set or get the option for collapsing the object's structure in the project tree.
IsLibraryElement on page 345	To get the information whether the object is a library element.
ModificationDate on page 358	To get the date of the last object modification.
Name on page 360	To get the name of the object.
OperationMode on page 364	To set or get the operation mode of a built-in library.
Parent on page 368	To get the parent of the specified object.
Protected on page 373	To get the information whether the object is protected.
ResultLevel on page 382	To get the result level of the specified object.
StateIconPath on page 394	To get the path to the symbol representing the state of the object.
SubBlocks on page 399	To get the subblocks of the specified object.
Type on page 402	To get the type of the specified object.

Methods

The LibraryFolder1 object definition contains the following methods:

Method	Purpose
Highlight on page 441	To highlight the object's element.

Events

None

Log (Object)

Object

Log

Syntax

No direct creation.

Purpose

To provide methods for accessing the message logs.

Description

The Log object provides methods for simultaneously writing messages of a specified severity to the Message Viewer and to the dSPACE log file:

- The log in the *Message Viewer* contains only messages that are written by AutomationDesk. This log is cleared automatically when AutomationDesk is started. It can be cleared explicitly by using a method that is provided by the Log object.
- The *dSPACE log file* is filled by various installed dSPACE products. Its contents persist the AutomationDesk sessions.

Properties

None

Methods

The Log object definition contains the following methods:

Method	Purpose
ClearMessages on page 418	To clear the messages shown in the Message Viewer.
WriteError on page 477	To write an error message to the log.
WriteInformation on page 478	To write an informational message to the log.
WriteMessage on page 478	To write a message of a specified severity to the log.
WriteWarning on page 479	To write a warning message to the log.

Events

None

LogicalLinkChildBase

Syntax

No direct creation.

Purpose

To handle the child objects of a LogicalLink object.

Description

A LogicalLinkChildBase object is an element that provides the methods and properties to access the child elements of a LogicalLink data object.

For example, it provides access to the ControlPrimitives and Services of a D3LogicalLink.

Properties

The LogicalLinkChildBase object definition contains the following properties:

Property	Purpose
ChildDataObjects on page 308	To get the data objects contained in the LogicalLinkChildBase object.
HierarchyName on page 328	To get the hierarchy path of the object.
IconPath on page 332	To get the path to the symbol representing the object type.
IsLibraryElement on page 345	To check whether the object is a library object.
Name on page 360	To set or get the name of the object.
Parent on page 368	To get the parent of the specified object.
Protected on page 373	To check whether the object is protected.
ReferenceName on page 378	To set or get the name of the referenced object.
ResultLevel on page 382	To set or get the result level of the object.
StateIconPath on page 394	To get the path to the symbol representing the state of the object.
Type on page 402	To set or get the type of the specified object.

Methods

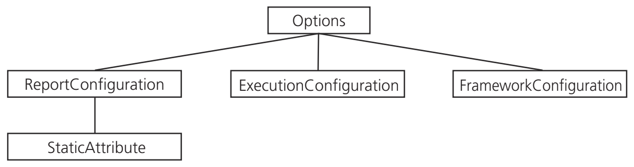
None

Events

None

Options (Object)

Object



Syntax

No direct creation.

Purpose

To configure the execution and report options.

Description

The Options object contains access to the configuration of the execution and the report, for configuring the execution and report generation.

Properties

The Options object definition contains the following properties:

Property	**Purpose**
Execution on page 320	To get the execution configuration object.
Framework (Property) on page 325	To get the FrameworkConfiguration object for configuring the XIL API framework.
Parent on page 368	To get the parent of the specified object.
Report on page 378	To get the report options.

Methods

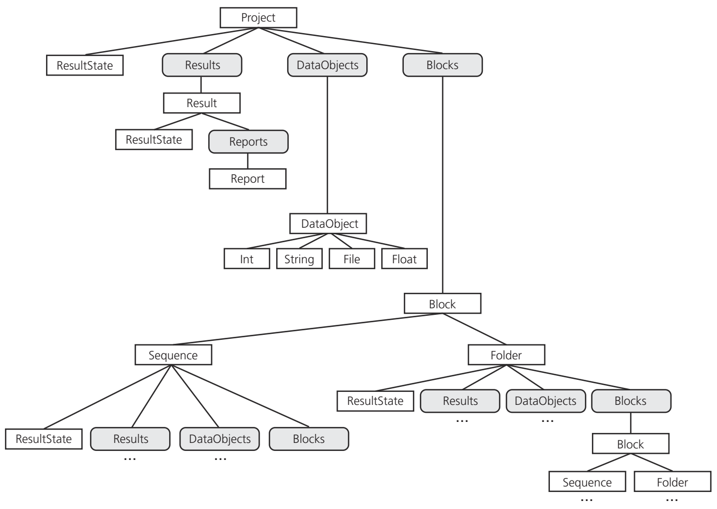
None

Events

None

Project

Object



Syntax

No direct creation.

Purpose

To handle an AutomationDesk project.

Description

The Project object is the root node of an entire AutomationDesk project. It can contain several folders, sequences, and data objects, organized in a hierarchical structure.

Properties

The Project object definition contains the following properties:

Property	Purpose
Author on page 290	To set or get the name of the person who created the object.
CreationDate on page 314	To get the date the object was created.
DataObjects (Property) on page 316	To get the collection object for accessing a data object.
Description on page 317	To set or get the description of the object.
HierarchyName on page 328	To get the hierarchy path of the object.
Hyperlink on page 329	To get the AutomationDesk hyperlink of the object.
IconPath on page 332	To get the path to the symbol representing the object type.
IsCollapsed on page 340	To set or get the option for collapsing the object's structure in the project tree.
IsLibraryElement on page 345	To check whether the object is a library object.
ModificationDate on page 358	To get the date of the last modification of the object.
Modified on page 359	To look up whether the project object was modified.

Property	Purpose
Name on page 360	To get the name of the object.
Parent on page 368	To get the parent of the specified object.
Path on page 369	To get the path of the specified object.
Protected on page 373	To check whether the object is protected.
ReadOnly on page 376	To look up whether the project is read-only.
ResultLevel on page 382	To set or get the result level of the specified object.
Results (Property) on page 383	To get the results of the specified object.
ResultState (Property) on page 384	To get the result state of the specified object.
StateIconPath on page 394	To get the path to the symbol representing the state of the object.
SubBlocks on page 399	To get the subblocks of the specified object.
Type on page 402	To get the type of the specified object.

Methods

The Project object definition contains the following methods:

Method	Purpose
ClearValues on page 419	To recursively clear the values of all contained output data objects and/or local data objects.
Close on page 421	To close a project.
Execute on page 433	To execute the sequences of a folder.
Export on page 434	To export a project as a ZIP file.
Highlight on page 441	To highlight the object's element.
Save on page 470	To save the project.
SaveAs on page 471	To save the project with a new file name.
Synchronize on page 476	To synchronize the sequences with the custom library templates.

Events

The Project object definition contains the following events:

Event	Purpose
OnExecutionFinished on page 484	To react to an execution finishing.
OnExecutionProgress on page 484	To react to the progress of an execution.
OnExecutionStarted on page 485	To react to an execution starting.
OnModified on page 485	To react to a project being modified.
OnShouldExecutionBeStopped on page 495	To react to an execution stopping.

Project1

Syntax	No direct creation.
Purpose	To handle an AutomationDesk project.

Description

The Project1 object is based on the interface definition of the Project object. It additionally provides the methods for importing and exporting a project via XML file.

The Project object is the root node of an entire AutomationDesk project. It can contain several folders, sequences, and data objects, organized in a hierarchical structure.

Properties

The Project1 object definition contains the following properties:

Property	Purpose
Author on page 290	To set or get the name of the person who created the object.
CreationDate on page 314	To get the date the object was created.
DataObjects (Property) on page 316	To get the collection object for accessing a data object.
Description on page 317	To set or get the description of the object.
HierarchyName on page 328	To get the hierarchy path of the object.
Hyperlink on page 329	To get the AutomationDesk hyperlink of the object.
IconPath on page 332	To get the path to the symbol representing the object type.
IsCollapsed on page 340	To set or get the option for collapsing the object's structure in the project tree.
IsLibraryElement on page 345	To check whether the object is a library object.
ModificationDate on page 358	To get the date of the last modification of the object.
Modified on page 359	To look up whether the project object was modified.
Name on page 360	To get the name of the object.
Parent on page 368	To get the parent of the specified object.
Path on page 369	To get the path of the specified object.
Protected on page 373	To check whether the object is protected.
ReadOnly on page 376	To look up whether the project is read-only.
ResultLevel on page 382	To set or get the result level of the specified object.
Results (Property) on page 383	To get the results of the specified object.
ResultState (Property) on page 384	To get the result state of the specified object.
StateIconPath on page 394	To get the path to the symbol representing the state of the object.
SubBlocks on page 399	To get the subblocks of the specified object.
Type on page 402	To get the type of the specified object.

Methods

The Project1 object definition contains the following methods:

Method	Purpose
ClearValues on page 419	To recursively clear the values of all contained output data objects and/or local data objects.
Close on page 421	To close a project.
Execute on page 433	To execute the sequences of a folder.
Export on page 434	To export a project as a ZIP file.
ExportFile on page 435	To export a project to an XML file.
Highlight on page 441	To highlight the object's element.
ImportFile on page 446	To import a folder or a sequence to the instantiated project object from an XML file.
Save on page 470	To save the project.
SaveAs on page 471	To save the project with a new file name.
Synchronize on page 476	To synchronize the sequences with the custom library templates.

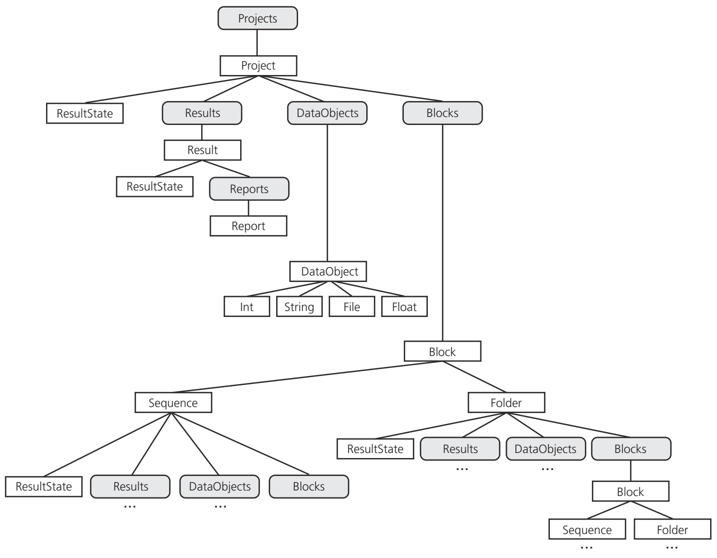
Events

The Project1 object definition contains the following events:

Event	Purpose
OnExecutionFinished on page 484	To react to an execution finishing.
OnExecutionProgress on page 484	To react to the progress of an execution.
OnExecutionStarted on page 485	To react to an execution starting.
OnModified on page 485	To react to a project being modified.
OnShouldExecutionBeStopped on page 495	To react to an execution stopping.

Projects (Object)

Object



Syntax

No direct creation.

Purpose

To handle AutomationDesk projects.

Description

The Projects object provides access to the project collection, allowing you to create, load, or import projects and handle them.

If you create an AutomationDesk project, you must select a project template that defines the general structure of the project. With the Standard Project template, you can build project structures for standard AutomationDesk projects.

Properties

The Projects object definition contains the following properties:

Property	**Purpose**
ActiveProject on page 288	To set or get the active project.
Count on page 312	To get the number of the object's instances.
Names on page 361	To get the child element names of a collection.
Parent on page 368	To get the parent of the specified object.
ProjectTemplates on page 372	To get the available project templates.

Methods

The Projects object definition contains the following methods:

Method	Purpose
CloseAll on page 422	To close all projects.
Create on page 425	To create a new object based on its collection object.
FindElement on page 437	To get the object of the element that is specified by its hierarchy path.
Import on page 442	To import a project from a ZIP file.
Item on page 449	To get a specific item of the specified object.
Load on page 451	To load a project.
SaveAll on page 471	To save all projects.

Events

None

Related topics

References

Projects1

Syntax

No direct creation.

Purpose

To handle AutomationDesk projects.

Description

The Projects1 object is based on the interface definition of the Projects object. It additionally provides the method for importing a project from an XML file. The Projects1 object provides access to the project collection, allowing you to create, load, or import projects and handle them.

If you create an AutomationDesk project, you must select a project template that defines the general structure of the project. With the Standard Project template, you can build project structures for standard AutomationDesk projects.

Properties

The Projects object definition contains the following properties:

Property	Purpose
ActiveProject on page 288	To set or get the active project.
Count on page 312	To get the number of the object's instances.
Names on page 361	To get the child element names of a collection.
Parent on page 368	To get the parent of the specified object.
ProjectTemplates on page 372	To get the available project templates.

Methods

The Projects object definition contains the following methods:

Method	Purpose
CloseAll on page 422	To close all projects.
Create on page 425	To create a new object based on its collection object.
FindElement on page 437	To get the object of the element that is specified by its hierarchy path.
Import on page 442	To import a project from a ZIP file.
ImportProject on page 447	To import a project from an XML or ZIP file.
Item on page 449	To get a specific item of the specified object.
Load on page 451	To load a project.
SaveAll on page 471	To save all projects.

Events

None

Related topics

References

Projects2

Syntax

No direct creation.

Purpose

To handle AutomationDesk projects.

Description

The Projects2 object is based on the interface definition of the Projects1 object. It additionally provides methods for importing and loading a project without displaying the update confirmation dialog. The update confirmation dialog is displayed if you open an AutomationDesk project with a newer AutomationDesk version before the automatic migration is started.

Properties

The Projects2 object definition contains the following properties:

Property	Purpose
ActiveProject on page 288	To set or get the active project.
Count on page 312	To get the number of the object's instances.
Names on page 361	To get the child element names of a collection.
Parent on page 368	To get the parent of the specified object.
ProjectTemplates on page 372	To get the available project templates.

Methods

The Projects2 object definition contains the following methods:

Method	Purpose
CloseAll on page 422	To close all projects.
Create on page 425	To create a new object based on its collection object.
FindElement on page 437	To get the object of the element that is specified by its hierarchy path.
Import on page 442	To import a project from a ZIP file.
ImportEx on page 445	To import a project from a ZIP file with the option to suppress the update confirmation dialog when using a newer AutomationDesk version.
ImportProject on page 447	To import a project from an XML or ZIP file.
Item on page 449	To get a specific item of the specified object.
Load on page 451	To load a project.
LoadEx on page 453	To load a project with the option to suppress the update confirmation dialog when using a newer AutomationDesk version.
SaveAll on page 471	To save all projects.

Events

None

Related topics

References

Projects (Object)........125
Projects1........126

Projects3

Syntax

No direct creation.

Purpose

To handle AutomationDesk projects.

Description

The Projects3 object is based on the interface definition of the Projects2 object. Additionally, it provides a method for opening an AutomationDesk project from a file.

Files of the following formats are supported:

- ADPX files that contain an AutomationDesk project that is saved in XML format using AutomationDesk 6.2 or later.
- ADP files that contain an AutomationDesk project that is saved in a binary legacy format using AutomationDesk 6.1 or earlier.
- ZIP files that contain an AutomationDesk project that is exported as a compressed archive.
- APX files that contain an AutomationDesk project that is exported in legacy XML format using AutomationDesk 6.0 or earlier.
- ADPX files that contain an AutomationDesk project that is exported in XML format using AutomationDesk 6.1 or later.

Properties

The Projects3 object definition contains the following properties:

Property	Purpose
ActiveProject on page 288	To set or get the active project.
Count on page 312	To get the number of the object's instances.
Names on page 361	To get the child element names of a collection.
Parent on page 368	To get the parent of the specified object.
ProjectTemplates on page 372	To get the available project templates.

Methods

The Projects3 object definition contains the following methods:

Method	Purpose
CloseAll on page 422	To close all projects.
Create on page 425	To create a new object based on its collection object.
FindElement on page 437	To get the object of the element that is specified by its hierarchy path.
Import on page 442	To import a project from a ZIP file.
ImportEx on page 445	To import a project from a ZIP file with the option to suppress the update confirmation dialog when using a newer AutomationDesk version.

Method	Purpose
ImportProject on page 447	To import a project from an XML or ZIP file.
Item on page 449	To get a specific item of the specified object.
Load on page 451	To load a project.
LoadEx on page 453	To load a project with the option to suppress the update confirmation dialog when using a newer AutomationDesk version.
Open on page 455	To open a project from a file.
SaveAll on page 471	To save all projects.

Events

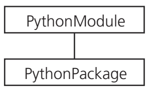
None

Related topics

References

PythonModule

Object



Syntax

No direct creation.

Purpose

To handle a Python module.

Description

A PythonModule object gives you access to a Python module that was added to a custom library.

All PythonModule objects and PythonPackage objects that are contained in a custom library folder are managed by its PythonModules collection.

Properties

The PythonModule object definition contains the following properties:

Property	Purpose
HierarchyName on page 328	To get the hierarchy path of the object.
Hyperlink on page 329	To get the AutomationDesk hyperlink of the object.
IconPath on page 332	To get the path to the symbol representing the object type.
IsLibraryElement on page 345	To check whether the object is a library object.
Name on page 360	To set or get the name of the object.
Parent on page 368	To get the parent object of the specified object.
Path on page 369	To get the path that contains the specified object.
Protected on page 373	To check whether the object is protected.
ResultLevel on page 382	To set or get the result level of the object.
StateIconPath on page 394	To get the path to the symbol representing the state of the object.
Type on page 402	To get the type of the specified object.

Methods

The PythonModule object definition contains the following method:

Method	Purpose
Highlight on page 441	To highlight the object's element.

Events

The PythonModule object definition contains the following event:

Event	Purpose
OnModified on page 485	To react to a Python module being modified.

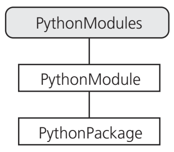
Related topics

References

Python Module (AutomationDesk Basic Practices 📖)
PythonModules... 131

PythonModules

Object



Syntax

No direct creation.

Purpose

To handle Python modules and packages.

Description

The PythonModules collection provides access to the Python modules and packages that were added to a custom library.

Properties

The PythonModules object definition contains the following properties:

Property	Purpose
Count on page 312	To get the number of object instances.
Names on page 361	To get the child element names of a collection.
Parent on page 368	To get the parent of the specified object.

Methods

The PythonModules object definition contains the following methods:

Method	Purpose
FindElement on page 437	To get the object of the element that is specified by its hierarchy path.
Item on page 449	To get a specific item of the specified object.

Events

None

Related topics

References

PythonPackage

Object

PythonPackage

Syntax

No direct creation.

Purpose

To handle a Python package.

Description

A PythonPackage object gives you access to a Python package that was added to a custom library.

All PythonModule objects and PythonPackage objects that are contained in a custom library folder are managed by its PythonModules collection.

A PythonPackage object provides a property with a collection that lets you access the Python modules and packages that are contained in the package.

Properties

The PythonPackage object definition contains the following properties:

Property	Purpose
HierarchyName on page 328	To get the hierarchy path of the object.
Hyperlink on page 329	To get the AutomationDesk hyperlink of the object.
IconPath on page 332	To get the path to the symbol representing the object type.
IsLibraryElement on page 345	To check whether the object is a library object.
Name on page 360	To set or get the name of the object.
Parent on page 368	To get the parent object of the specified object.
Path on page 369	To get the path that contains the specified object.
Protected on page 373	To check whether the object is protected.
PythonModules (Property) on page 374	To get the collection object for accessing the contained Python modules and packages.
ResultLevel on page 382	To set or get the result level of the object.
StateIconPath on page 394	To get the path to the symbol representing the state of the object.
Type on page 402	To get the type of the specified object.

Methods

The PythonPackage object definition contains the following method:

Method	Purpose
Highlight on page 441	To highlight the object's element.

Events

The PythonPackage object definition contains the following event:

Event	Purpose
OnModified on page 485	To react to a Python package being modified.

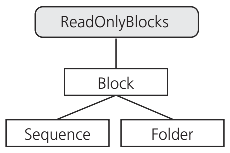
Related topics

References

Python Package (AutomationDesk Basic Practices 📖)
PythonModules... 131

ReadOnlyBlocks

Object



Syntax

No direct creation.

Purpose

To access automation blocks from the AutomationDesk libraries.

Description

The ReadOnlyBlocks object contains access to the blocks collection of the library. You can use the block templates to create blocks in a project. A block can be a folder template or a sequence template. You can build a hierarchical project tree with these templates.

Properties

The ReadOnlyBlocks object definition contains the following properties:

Property	Purpose
Count on page 312	To get the number object instances.
Names on page 361	To get the child element names of a collection.
Parent on page 368	To get the parent of the specified object.

Methods

The ReadOnlyBlocks object definition contains the following method:

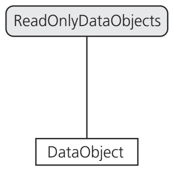
Method	Purpose
FindElement on page 437	To get the object of the element that is specified by its hierarchy path.
Item on page 449	To get a specific item of the specified object.

Events

None

ReadOnlyDataObjects

Object



Syntax

No direct creation.

Purpose

To access data objects from the AutomationDesk libraries.

Description

The ReadOnlyDataObjects object contains access to the data object collection of the library. You can use the data object templates to create and manage data objects in your project.

Properties

The ReadOnlyDataObjects object definition contains the following properties:

Property	Purpose
Count on page 312	To get the number of instances of the object.
Names on page 361	To get the child element names of a collection.
Parent on page 368	To get the parent of the specified object.

Methods

The ReadOnlyDataObjects object definition contains the following method:

Method	Purpose
FindElement on page 437	To get the object of the element that is specified by its hierarchy path.
Item on page 449	To get a specific item of the specified object.

Events

None

Report

Object

Syntax No direct creation.

Purpose To handle a specific report.

Description The Report object contains access to a specific report. A report displays the result information according to the type and the depth that are specified in the configuration. Whereas the settings for the result depth are saved within the project, the settings for the report generation are stored in the registry. A report can be generated for each result and appears as a child element of the result.

Properties The Report object definition contains the following properties:

Property	Purpose
Name on page 360	To get the name of the object.
Parent on page 368	To get the parent of the specified object.
Path on page 369	To get the path of the specified object.
ReportType on page 380	To get the output format of a report.
Type on page 402	To get the type of the specified object.

Methods None

Events The Report object definition contains the following event:

Event	Purpose
OnModified on page 485	To react to a report being modified.

Report1

Syntax No direct creation.

Purpose To handle a specific report.

Description

The Report1 object contains access to a specific report. The Report1 object is based on the interface definition of the Report object. It additionally provides a method for exporting a report.

A report displays the result information according to the type and the depth that are specified in the configuration. Whereas the settings for the result depth are saved within the project, the settings for the report generation are external stored. A report can be generated for each result and appears as a child element of the result.

Properties

The Report1 object definition contains the following properties:

Property	Purpose
Name on page 360	To get the name of the object.
Parent on page 368	To get the parent of the specified object.
Path on page 369	To get the path of the specified object.
ReportType on page 380	To get the output format of a report.
Type on page 402	To get the type of the specified object.

Methods

The Report1 object definition contains the following methods:

Method	Purpose
Export on page 434	To save a report to a specified folder in the same report format (HTML or PDF).

Events

The Report1 object definition contains the following event:

Event	Purpose
OnModified on page 485	To react to a report being modified.

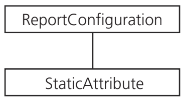
Related topics

References

Report.. 135

ReportConfiguration

Object



Syntax

No direct creation.

Purpose

To configure the report options.

Description

The ReportConfiguration object contains access to the report options, where you can specify the report attributes to be added to the report, or the style sheet to be used. You can also specify the logo path and logo alignment of the specified logo, the report type, or the static attributes.

Properties

The ReportConfiguration object definition contains the following properties:

Property	Purpose
AvailableAttributes on page 291	To get the list of available attributes which you can add to the report.
IsAllAttributes on page 340	To set or get the option for adding all attributes or a customized set of attributes to the report.
IsCustomReport on page 342	To set or get the option for using a custom style sheet for report generation.
LogoAlignment on page 352	To set or get the alignment of the logo used in the report.
LogoPath on page 353	To set or get the path to the logo used in the report.
Parent on page 368	To get the parent of the specified object.
ReportType on page 380	To set or get the output format of a report.
StaticAttribute on page 395	To set or get the static attributes of a report.
StyleSheetPath on page 398	To set or get the path of the report's style sheet.
VisibleAttributes on page 410	To set or get the specified subset of attributes to be added to the report.

AutomationDesk and the Automation Server access the same settings. If you modify the settings via the Automation Server, the new settings are also valid for AutomationDesk sessions, and vice versa. Take care to configure both the **ReportType** property and the **StyleSheetPath** property.

Methods

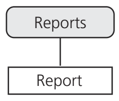
None

Events

None

Reports (Object)

Object



Syntax

No direct creation.

Purpose

To create and handle reports.

Description

The Reports object contains access to the report collection. You can create and manage reports.

A report can be generated directly after an execution, or later. Its content depends on the execution's result and on the report configuration. After generation, the report is stored as a child element of the result.

Properties

The Reports object definition contains the following properties:

Property	Purpose
Count on page 312	To get the number of instances of the object.
Names on page 361	To get the child element names of a collection.
Parent on page 368	To get the parent of the specified object.

Methods

The Reports object definition contains the following methods:

Method	Purpose
GenerateReport on page 438	To generate a report.
FindElement on page 437	To get the object of the element that is specified by its hierarchy path.
Item on page 449	To get a specific item of the specified object.
Remove on page 459	To delete a report.
RemoveAll on page 460	To delete all the created child elements of a collection.

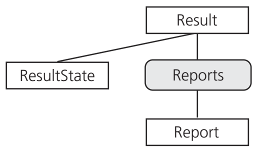
Events

The Reports object definition contains the following events:

Event	Purpose
OnAdd on page 482	To react to a report being created.
OnRemove on page 494	To react to a report being deleted.

Result

Object



Syntax

No direct creation.

Purpose

To handle a specific result.

Description

The Result object contains access to a specific result. A result is logged during execution and added as child element of the executed project, folder, or sequence in the project tree. All folders and sequences below the execution's starting point are executed in the order in which they appear in the project tree. Their information is added to the result according to the settings of the execution configuration.

Properties

The Result object definition contains the following properties:

Property	Purpose
Name on page 360	To get the name of the object.
Parent on page 368	To get the parent of the specified object.
Reports (Property) on page 379	To get the created reports of a result.
ResultState (Property) on page 384	To get the result state of the specified object.
Type on page 402	To get the type of the specified object.

Methods

None

Events

The Result object definition contains the following event:

Event	Purpose
OnModified on page 485	To react to a result being modified.

Result1

Syntax

No direct creation.

Purpose

To handle a specific result.

Description

The Result1 object contains access to a specific result. A result is logged during execution and added as child element of the executed project, folder, or sequence in the project tree. All folders and sequences below the execution's starting point are executed in the order in which they appear in the project tree. Their information is added to the result according to the settings of the execution configuration.

Results are stored on the host PC in folders that are located below the project folder and named with a Globally Unique Identifier (GUID). To this folders, an XML file named **ReportPool.xml** is written, that contains the configuration of the result.

Properties

The Result1 object definition contains the following properties:

Property	**Purpose**
Name on page 360	To get the name of the object.
Parent on page 368	To get the parent of the specified object.
Path on page 369	To get the path of the result's XML file.
Reports (Property) on page 379	To get the created reports of a result.
ResultState (Property) on page 384	To get the result state of the specified object.
Type on page 402	To get the type of the specified object.

Methods

None

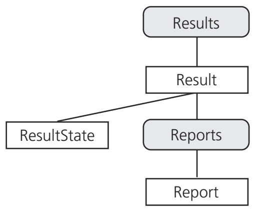
Events

The Result1 object definition contains the following event:

Event	**Purpose**
OnModified on page 485	To react to a result being modified.

Results (Object)

Object



Syntax

No direct creation.

Purpose

To handle results.

Description

The Results object gives you access to the result collection. A result is logged during execution and added as a child element of the executed project, folder, or sequence. All folders and sequences below the execution's starting point are executed recursively. Their information is added to the result according to the settings of the execution configuration.

Properties

The Results object definition contains the following properties:

Property	Purpose
Count on page 312	To get the number of instances of the object instances.
Names on page 361	To get the child element names of a collection.
Parent on page 368	To get the parent of the specified object.

Methods

The Results object definition contains the following methods:

Method	Purpose
FindElement on page 437	To get the object of the element that is specified by its hierarchy path.
Item on page 449	To get a specific item of the specified object.
Remove on page 459	To delete an object.
RemoveAll on page 460	To delete all the created child elements of a collection.

Events

The Results object definition contains the following events:

Event	Purpose
OnAdd on page 482	To react to a result being created.
OnRemove on page 494	To react to a result being deleted.

ResultState (Object)

Object

ResultState

Syntax

No direct creation.

Purpose

To handle state information of a result.

Description

The ResultState object handles information of the result. It contains the state of failed, passed, undefined, and error result states, and also the start time, stop time, and duration time of the execution.

Properties

The ResultState object definition contains the following properties:

Property	Purpose
ErrorCount on page 319	To get the state of subblocks of the executed element that returned with an error.
ExecutionDuration on page 321	To get the duration time of the execution.
FailedCount on page 322	To get the state of failed subblocks of the executed element.
Operator on page 363	To get the name of the person who started the execution.
Parent on page 368	To get the parent of the specified object.
PassedCount on page 369	To get the state of passed subblocks of the executed element.
StartTime on page 393	To get the start time of an execution.
StopTime on page 396	To get the stop time of an execution.
UndefinedCount on page 403	To get the state of undefined subblocks of the executed elements.

Methods

None

Events

None

ResultState1

Syntax	No direct creation.
Purpose	To handle state information of a result.
Description	The ResultState1 object handles information of the result. The ResultState1 object is based on the interface definition of the ResultState object. It additionally provides the verdict property to qualify the result.

Properties

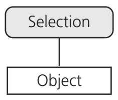
The ResultState1 object definition contains the following properties:

Property	Purpose
ErrorCount on page 319	To get the state of subblocks of the executed element that returned with an error.
ExecutionDuration on page 321	To get the duration time of the execution.
FailedCount on page 322	To get the state of failed subblocks of the executed element.
Operator on page 363	To get the name of the person who started the execution.
Parent on page 368	To get the parent of the specified object.
PassedCount on page 369	To get the state of passed subblocks of the executed element.
StartTime on page 393	To get the start time of an execution.
StopTime on page 396	To get the stop time of an execution.
UndefinedCount on page 403	To get the state of undefined subblocks of the executed elements.
Verdict (Property) on page 409	To get an expression to qualify the ResultState.

Methods	None
Events	None

Selection (Object)

Object



Syntax

No direct creation.

Purpose

To access the collection of selected elements.

Description

The Selection object provides access to the collection of elements that are selected in the AutomationDesk COM interface.

Properties

The Selection object definition contains the following property:

Property	Purpose
Count on page 312	To get the number of object instances.

Methods

The Selection object definition contains the following method:

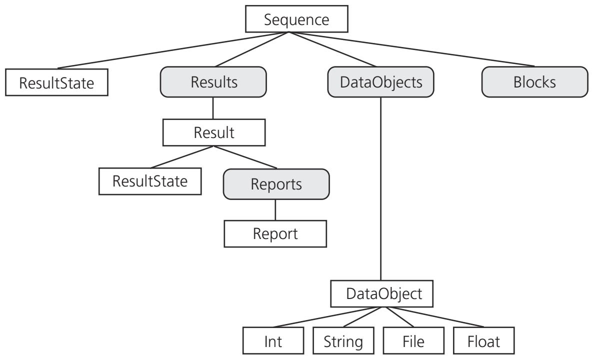
Method	Purpose
Item on page 449	To get the object at the specified position.

Events

None

Sequence

Object



Syntax

No direct creation.

Purpose

To handle a specific sequence.

Description

The Sequence object contains access to a specific sequence. A sequence contains an executable program flow. You can execute this sequence, but you cannot modify it, for example, you cannot change its program flow.

A sequence can be used for other projects if it is added to the custom library to create a new customized template.

Properties

The Sequence object definition contains the following properties:

Property	Purpose
Author on page 290	To set or get the name of the person who created the object.
CreationDate on page 314	To get the date the object was created.
DataObjects (Property) on page 316	To get the collection object for accessing a data object.
Description on page 317	To set or get the description of the object.
HasLibraryLink on page 327	To look up whether the sequence is linked to the custom library.
HierarchyName on page 328	To get the hierarchy path of the object.
Hyperlink on page 329	To get the AutomationDesk hyperlink of the object.
IconPath on page 332	To get the path to the symbol representing the object type.
IsCollapsed on page 340	To set or get the option for collapsing the object's structure in the project tree.
IsEnabled on page 342	To set or get the enable state of an element.
IsLibraryElement on page 345	To check whether the object is a library element.
LibraryLink on page 349	To get the library link of the sequence.
ModificationDate on page 358	To get the date of the last object modification.
Name on page 360	To set or get the name of the object.
Parent on page 368	To get the parent of the specified object.
Protected on page 373	To check whether the object is protected.
ResultLevel on page 382	To set or get the result level of the specified object.
Results (Property) on page 383	To get the results of the specified object.
ResultState (Property) on page 384	To get the result state of the specified object.
StateIconPath on page 394	To get the path to the symbol representing the state of the object.
SubBlocks on page 399	To get the subblocks of the specified object.
Type on page 402	To get the type of the specified object.

Methods

The Sequence object definition contains the following methods:

Method	Purpose
BreakLink on page 416	To break the link between the custom library and the instantiated sequence.
ClearValues on page 419	To recursively clear the values of all contained output data objects and/or local data objects.
Execute on page 433	To execute the specified sequence.
Highlight on page 441	To highlight the object's element.
Synchronize on page 476	To synchronize the sequence(s) with the custom library templates.

Events

The Sequence object definition contains the following events:

Event	Purpose
OnExecutionFinished on page 484	To react to an execution finishing.
OnExecutionProgress on page 484	To react to the progress of an execution.
OnExecutionStarted on page 485	To react to an execution starting.
OnModified on page 485	To react to a sequence being modified.
OnShouldExecutionBeStopped on page 495	To react to an execution stopping.

Sequence1

Syntax

No direct creation.

Purpose

To handle a specific sequence.

Description

The Sequence1 object is based on the interface definition of the Sequence object. It additionally provides the methods for exporting a sequence via XML file.

Via the Synect property, you can configure the synchronization with SYNECT.

The Sequence object contains access to a specific sequence. A sequence contains an executable program flow. You can execute this sequence, but you cannot modify it, for example, you cannot change its program flow.

A sequence can be used for other projects if it is added to the custom library to create a new customized template.

You must explicitly specify a Sequence1 object only, if you are using the constants of the AutomationDesk API. For further information, refer to Using API Constants on page 67.

Properties

The Sequence1 object definition contains the following properties:

Property	Purpose
Author on page 290	To set or get the name of the person who created the object.
CreationDate on page 314	To get the date the object was created.
DataObjects (Property) on page 316	To get the collection object for accessing a data object.
Description on page 317	To set or get the description of the object.
HasLibraryLink on page 327	To look up whether the sequence is linked to the custom library.
HierarchyName on page 328	To get the hierarchy path of the object.
Hyperlink on page 329	To get the AutomationDesk hyperlink of the object.
IconPath on page 332	To get the path to the symbol representing the object type.
IsCollapsed on page 340	To set or get the option for collapsing the object's structure in the project tree.
IsEnabled on page 342	To set or get the enable state of an element.
IsLibraryElement on page 345	To check whether the object is a library object.
LibraryLink on page 349	To get the library link of the sequence.
ModificationDate on page 358	To get the date of the last object modification.
Name on page 360	To set or get the name of the object.
Parent on page 368	To get the parent of the specified object.
Protected on page 373	To check whether the object is protected.
ResultLevel on page 382	To set or get the result level of the specified object.
Results (Property) on page 383	To get the results of the specified object.
ResultState (Property) on page 384	To get the result state of the specified object.
StateIconPath on page 394	To get the path to the symbol representing the state of the object.
SubBlocks on page 399	To get the subblocks of the specified object.
Synect (Property) on page 400	To get the Synect object.
Type on page 402	To get the type of the specified object.

Methods

The Sequence1 object definition contains the following methods:

Method	Purpose
BreakLink on page 416	To break the link between the custom library and the instantiated sequence.
ClearValues on page 419	To recursively clear the values of all contained output data objects and/or local data objects.
Execute on page 433	To execute the specified sequence.
ExportFile on page 435	To export a sequence to an XML file.

Method	Purpose
Highlight on page 441	To highlight the object's element.
Synchronize on page 476	To synchronize the sequence(s) with the custom library templates.

Events The Sequence1 object definition contains the following events:

Event	Purpose
OnExecutionFinished on page 484	To react to an execution finishing.
OnExecutionProgress on page 484	To react to the progress of an execution.
OnExecutionStarted on page 485	To react to an execution starting.
OnModified on page 485	To react to a sequence being modified.
OnShouldExecutionBeStopped on page 495	To react to an execution stopping.

StaticAttribute

Object

Syntax No direct creation.

Purpose To configure the attributes of a report.

Description You can use the StaticAttribute object to define whether to add any additional attributes to the reports beside the available attributes. You can specify to add the project and folders information, and the descriptions and result states of all the blocks, in addition to the sequence information. The outputs of Report blocks can also be added to the report.

Properties The StaticAttribute object definition contains the following properties:

Property	Purpose
IncludeDescription on page 335	To set or get the option for adding all descriptions to the report.
IncludeFolderAndProject on page 335	To set or get the option for adding folder and project information to the report.
IncludeReportBlocks on page 336	To set or get the option for adding the output of report blocks to the report.

Property	Purpose
IncludeResultState on page 337	To set or get the option for adding the result states to the report.
Parent on page 368	To get the parent of the specified object.

Methods None

Events None

Synect

Syntax No direct creation.

Purpose To handle the sychronization with SYNECT.

Description The Synect object provides properties to configure the synchronization with SYNECT in both directions, i.e., for push and for pull operations.

Properties The Synect object definition contains the following properties:

Property	Purpose
Ignore on page 333	To get or set whether the object and its child objects will be ignored for synchronization with SYNECT.
IsIgnored on page 343	To get whether the object is ignored for synchronization with SYNECT.

Methods None

Events None

Related topics

Basics

Basics on Using AutomationDesk with SYNECT (AutomationDesk Basic Practices 🕮)

References

Clear Ignore Flag (AutomationDesk Basic Practices 🕮)
Set Ignore Flag (AutomationDesk Basic Practices 🕮)
Synect (Property).. 400

TAMVersion (Object)

Object

TAMVersion

Syntax

No direct creation.

Purpose

To get information on the AutomationDesk version.

Description

The TAMVersion object contains a major release number, a minor release number, and a revision number. For example, for major release "6", minor release "3", and revision "1", TAMVersion returns "6.3.1".

Properties

The TAMVersion object definition contains the following properties:

Property	Purpose
Major on page 355	To get the AutomationDesk major release number.
Minor on page 357	To get the AutomationDesk minor release number.
Parent on page 368	To get the parent of the specified object.
Revision on page 384	To get the AutomationDesk revision number, i.e., the patch version.

Methods

None

Events

None

Evaluation

Signal

Syntax

No direct creation.

Purpose

To handle a Signal data object.

Description

The Signal object is a data object that contains the shape of a signal in discrete values. The time coordinates are provided in a vector of x-axis values. The function value coordinates are provided in a vector of y-axis values. Both vectors are of the same length.

Properties

The Signal object definition contains the following properties:

Property	Purpose
Author on page 290	To get the author of the Signal.
CreationDate on page 314	To get the date the object is created.
Description on page 317	To get the description of the Signal.
FcnValues on page 323	To get the vector of function values.
HasLibraryLink on page 327	To check whether the data object is linked to the custom library.
HierarchyName on page 328	To get the hierarchy path of the object.
IconPath on page 332	To get the path to the symbol representing the object type.
InOutState on page 338	To set or get the data direction of a data object.
IsLibraryElement on page 345	To check whether the object is a library object.
Length on page 348	To get the length of the contained vectors.
LibraryLink on page 349	To get the library link of the data object.
ModificationDate on page 358	To get the date of the last object modification.
Name on page 360	To set or get the name of the object.
Parent on page 368	To get the parent of the specified object.
Protected on page 373	To check whether the object is protected.
ReferenceName on page 378	To set or get the name of the referenced object.
ResultLevel on page 382	To set or get the result level of the specified object.
StateIconPath on page 394	To get the path to the symbol representing the state of the object.
Type on page 402	To get the type of the specified object.
XVector on page 412	To get the vector of time values.

Methods

The Signal object definition contains the following methods:

Method	Purpose
ClearValue on page 418	To clear the values of the data object.
SetValue on page 474	To set the signal's time and function values.

Events

The Signal object definition contains the following events:

Event	Purpose
OnValueChanged on page 496	To react to a property being modified.

Related topics

References

Signal (AutomationDesk Basic Practices 📖)

Main Library

Where to go from here

Information in this section

Bool

Object

Bool

Syntax

No direct creation.

Purpose

To handle a Bool data object.

Description

The Bool data object is used to provide a data object for the Boolean values `True` and `False`.

The default value is `False`.

Unlike in AutomationDesk, you can assign integer values to Bool data objects and Boolean values to Int data objects. The value `0` represents `False` and any other value represents `True`. The COM interface interprets `True` as `-1`.

Properties

The Bool object definition contains the following properties:

Property	Purpose
Author on page 290	To set or get the name of the person who created the object.
CreationDate on page 314	To get the date the object was created.
Description on page 317	To set or get the description of the object.
HasLibraryLink on page 327	To check whether the data object is linked to the custom library.
HierarchyName on page 328	To get the hierarchy path of the object.
IconPath on page 332	To get the path to the symbol representing the object type.
IsLibraryElement on page 345	To check whether the object is a library object.
LibraryLink on page 349	To get the library link of the data object.
ModificationDate on page 358	To get the date of the last object modification.
Name on page 360	To set or get the name of the object.
Parent on page 368	To get the parent of the specified object.
Protected on page 373	To check whether the object is protected.
ReferenceName on page 378	To set or get the name of the referenced object.
ResultLevel on page 382	To set or get the result level of the specified object.
StateIconPath on page 394	To get the path to the symbol representing the state of the object.
Type on page 402	To get the type of the specified object.
Value on page 404	To set or get a value of the data object's type.

Methods

None

Events

The Bool object definition contains the following event:

Event	Purpose
OnModified on page 485	To react to a Bool data object being modified.
OnValueChanged on page 496	To react to a Bool data object being changed.

Bool1

Syntax

No direct creation.

Purpose

To handle a Bool1 data object.

Description

The Bool1 object is based on the interface definition of the Bool object. It additionally provides the Synect property which, lets you specify to ignore the object during synchronization with SYNECT.

The Bool1 data object is used to provide a data object for the Boolean values `True` and `False`.

The default value is `False`.

Unlike in AutomationDesk, you can assign integer values to Bool1 data objects and Boolean values to Int data objects. The value `0` represents `False` and any other value represents `True`. The COM interface interprets `True` as `-1`.

Properties

The Bool1 object definition contains the following properties:

Property	Purpose
Author on page 290	To set or get the name of the person who created the object.
CreationDate on page 314	To get the date the object was created.
Description on page 317	To set or get the description of the object.
HasLibraryLink on page 327	To check whether the data object is linked to the custom library.
HierarchyName on page 328	To get the hierarchy path of the object.
IconPath on page 332	To get the path to the symbol representing the object type.
IsLibraryElement on page 345	To check whether the object is a library object.
LibraryLink on page 349	To get the library link of the data object.
ModificationDate on page 358	To get the date of the last object modification.
Name on page 360	To set or get the name of the object.
Parent on page 368	To get the parent of the specified object.
Protected on page 373	To check whether the object is protected.
ReferenceName on page 378	To set or get the name of the referenced object.
ResultLevel on page 382	To set or get the result level of the specified object.
StateIconPath on page 394	To get the path to the symbol representing the state of the object.
Synect (Property) on page 400	To get the Synect object.
Type on page 402	To get the type of the specified object.
Value on page 404	To set or get a value of the data object's type.

Methods

None

Events

The Bool1 object definition contains the following event:

Event	Purpose
OnModified on page 485	To react to a Bool1 data object being modified.
OnValueChanged on page 496	To react to a Bool1 data object being changed.

Condition (Object)

Object

Condition

Syntax

No direct creation.

Purpose

To handle a Condition data object.

Description

The Condition object is a data object. It is used in IfThenElse, Repeat and While automation blocks to control the command's execution.

The Condition object is either a library template or a project element.

Properties

The Condition object definition contains the following properties:

Property	Purpose
Author on page 290	To set or get the name of the person who created the object.
Condition (Property) on page 309	To set or get the expression of the Condition object.
CreationDate on page 314	To get the date the object is created.
Description on page 317	To set or get the description of the object.
HasLibraryLink on page 327	To check whether the data object is linked to the custom library.
HierarchyName on page 328	To get the hierarchy path of the object.
IconPath on page 332	To get the path to the symbol representing the object type.
IsLibraryElement on page 345	To check whether the object is a library object.
LibraryLink on page 349	To get the library link of the data object.
ModificationDate on page 358	To get the date of the last object modification.
Name on page 360	To set or get the name of the object.
Parent on page 368	To get the parent of the specified object.
Protected on page 373	To check whether the object is protected.
ResultLevel on page 382	To set or get the result level of the specified object.

Property	Purpose
StateIconPath on page 394	To get the path to the symbol representing the state of the object.
Type on page 402	To get the type of the specified object.

Methods

The Condition object definition contains the following method:

Method	Purpose
CheckSyntax on page 417	To check the syntax of the specified condition object.

Events

The Condition object definition contains the following event:

Event	Purpose
OnModified on page 485	To react to a condition being modified.
OnValueChanged on page 496	To react to a condition property being modified.

Condition1 (Object)

Syntax

No direct creation.

Purpose

To handle a Condition1 data object.

Description

The Condition1 object is based on the interface definition of the Condition (Object) object. It additionally provides the Synect property, which lets you specify to ignore the object during synchronization with SYNECT.

The Condition1 object is a data object. It is used in IfThenElse, Repeat and While automation blocks to control the command’s execution.

The Condition1 object is either a library template or a project element.

Properties

The Condition1 object definition contains the following properties:

Property	Purpose
Author on page 290	To set or get the name of the person who created the object.
Condition (Property) on page 309	To set or get the expression of the Condition1 object.
CreationDate on page 314	To get the date the object is created.
Description on page 317	To set or get the description of the object.

Property	Purpose
HasLibraryLink on page 327	To check whether the data object is linked to the custom library.
HierarchyName on page 328	To get the hierarchy path of the object.
IconPath on page 332	To get the path to the symbol representing the object type.
IsLibraryElement on page 345	To check whether the object is a library object.
LibraryLink on page 349	To get the library link of the data object.
ModificationDate on page 358	To get the date of the last object modification.
Name on page 360	To set or get the name of the object.
Parent on page 368	To get the parent of the specified object.
Protected on page 373	To check whether the object is protected.
ResultLevel on page 382	To set or get the result level of the specified object.
StateIconPath on page 394	To get the path to the symbol representing the state of the object.
Synect (Property) on page 400	To get the Synect object.
Type on page 402	To get the type of the specified object.

Methods

The Condition1 object definition contains the following method:

Method	Purpose
CheckSyntax on page 417	To check the syntax of the specified condition object.

Events

The Condition1 object definition contains the following event:

Event	Purpose
OnModified on page 485	To react to a condition being modified.
OnValueChanged on page 496	To react to a condition property being modified.

DataContainer

Object

DataContainer

Syntax

No direct creation.

Purpose

To handle a DataContainer object.

Description

The DataContainer data object can contain any data object or another DataContainer. You can use it to structure the project tree and to handle a group of data objects via one single project element.

The DataContainer object is either a library template or a project element.

Properties

The DataContainer object definition contains the following properties:

Property	Purpose
Author on page 290	To set or get the name of the person who created the object.
ChildDataObjects on page 308	To get the data objects contained in the DataContainer object.
CreationDate on page 314	To get the date the object is created.
Description on page 317	To set or get the description of the object.
HasLibraryLink on page 327	To check whether the data object is linked to the custom library.
HierarchyName on page 328	To get the hierarchy path of the object.
IconPath on page 332	To get the path to the symbol representing the object type.
IsLibraryElement on page 345	To check whether the object is a library object.
LibraryLink on page 349	To get the library link of the data object.
ModificationDate on page 358	To get the date of the last object modification.
Name on page 360	To set or get the name of the object.
Parent on page 368	To get the parent of the specified object.
Protected on page 373	To check whether the object is protected.
ResultLevel on page 382	To set or get the result level of the specified object.
StateIconPath on page 394	To get the path to the symbol representing the state of the object.
Type on page 402	To get the type of the specified object.

Methods

The DataContainer object definition contains the following method:

Method	Purpose
ClearValues on page 419	To recursively clear the values of all contained output data objects and/or local data objects.

Events

The DataContainer object definition contains the following event:

Event	Purpose
OnModified on page 485	To react to a DataContainer being modified.

DataContainer1

Object

DataContainer

Syntax

No direct creation.

Purpose

To handle a DataContainer1 object.

Description

You can use data containers to structure the project tree and to handle a group of data objects via one single project element. A data container can contain any data object or another data container.

The DataContainer1 data object is based on the interface definition of the DataContainer object and additionally provides the Synect property that lets you configure the synchronization with SYNECT.

The DataContainer1 object is either a library template or a project element.

Properties

The DataContainer1 object definition contains the following properties:

Property	Purpose
Author on page 290	To set or get the name of the person who created the object.
ChildDataObjects on page 308	To get the data objects contained in the DataContainer1 object.
CreationDate on page 314	To get the date the object is created.
Description on page 317	To set or get the description of the object.
HasLibraryLink on page 327	To check whether the data object is linked to the custom library.
HierarchyName on page 328	To get the hierarchy path of the object.
IconPath on page 332	To get the path to the symbol representing the object type.
IsLibraryElement on page 345	To check whether the object is a library object.
LibraryLink on page 349	To get the library link of the data object.
ModificationDate on page 358	To get the date of the last object modification.
Name on page 360	To set or get the name of the object.
Parent on page 368	To get the parent of the specified object.
Protected on page 373	To check whether the object is protected.
ResultLevel on page 382	To set or get the result level of the specified object.
StateIconPath on page 394	To get the path to the symbol representing the state of the object.
Synect (Property) on page 400	To get the Synect object.
Type on page 402	To get the type of the specified object.

Methods

The DataContainer1 object definition contains the following method:

Method	Purpose
ClearValues on page 419	To recursively clear the values of all contained output data objects and/or local data objects.

Events

The DataContainer1 object definition contains the following event:

Event	Purpose
OnModified on page 485	To react to a DataContainer1 being modified.

Dictionary

Object

Dictionary

Syntax

No direct creation.

Purpose

To handle a Dictionary data object.

Description

The Dictionary object is only the top level instance of a Dictionary data object in your AutomationDesk project. Like in AutomationDesk, it provides a root element that let you access the dictionary's contents. For information on the root element's properties and methods, refer to the RootElement property.

A dictionary consists of key-value pairs. A key can be of Int, Float or String data type.

Properties

The Dictionary object definition contains the following properties:

Property	Purpose
Author on page 290	To set or get the name of the person who created the object.
CreationDate on page 314	To get the date the object is created.
Description on page 317	To set or get the description of the object.
HasLibraryLink on page 327	To check whether the data object is linked to the custom library.
HierarchyName on page 328	To get the hierarchy path of the object.
IconPath on page 332	To get the path to the symbol representing the object type.

Property	Purpose
IsLibraryElement on page 345	To check whether the object is a library object.
LibraryLink on page 349	To get the library link of the data object.
ModificationDate on page 358	To get the date of the last object modification.
Name on page 360	To set or get the name of the object.
Parent on page 368	To get the parent object of the specified object.
Protected on page 373	To check whether the object is protected.
ReferenceName on page 378	To set or get the name of the referenced object.
ResultLevel on page 382	To set or get the result level of the specified object.
RootElement on page 385	To get access to the contents of the Dictionary object.
RootElement.Count on page 387	To get the number of items in a RootElement object.
RootElement.Keys on page 387	To get the keys available in a Dictionary or dictionary-based RootElement object.
RootElement.ParentObject on page 388	To get the parent of an item in a RootElement object.
RootElement.RootObject on page 388	To get the parent of a RootElement object.
RootElement.Type on page 389	To get the type of the contents of the RootElement object.
RootElement.Value on page 390	To get the contents of the RootElement object.
StateIconPath on page 394	To get the path to the symbol representing the state of the object.
Type on page 402	To get the type of the specified object.
ValueString on page 406	To set or get the values of the Dictionary object as a string.

Methods

The Dictionary object definition contains the following method:

Method	Purpose
CreateSubItem on page 428	To create a subitem that can be added to the root element.
ClearValue on page 418	To clear the values of the data object.
RootElement.Add on page 461	To add an item to a RootElement object.
RootElement.Clear on page 462	To clear the contents of the RootElement object.
RootElement.Contains on page 463	To check whether the specified key is available in the dictionary-based RootElement object.
RootElement.GetItem on page 464	To get an item of the RootElement object.
RootElement.Remove on page 466	To remove the specified item from the RootElement object.
RootElement.SetItem on page 468	To edit the value of an item of a RootElement object.

Events

The Dictionary object definition contains the following event:

Event	Purpose
OnModified on page 485	To react to a Dictionary being modified.
OnValueChanged on page 496	To react to a Dictionary item being changed.

Dictionary1

Syntax

No direct creation.

Purpose

To handle a Dictionary1 data object.

Description

The Dictionary1 object is only the top level instance of a Dictionary data object in your AutomationDesk project. Like in AutomationDesk, it provides a root element that let you access the dictionary's contents. For information on the root element's properties and methods, refer to the RootElement property.

A dictionary consists of key-value pairs. A key can be of Int, Float or String data type.

The Dictionary1 object is based on the interface definition of the Dictionary object. It additionally provides the ValueList property to get the list of valid values for the data object.

Via the Synect property, you can configure the synchronization with SYNECT.

Properties

The Dictionary1 object definition contains the following properties:

Property	**Purpose**
Author on page 290	To set or get the name of the person who created the object.
CreationDate on page 314	To get the date the object is created.
Description on page 317	To set or get the description of the object.
HasLibraryLink on page 327	To check whether the data object is linked to the custom library.
HierarchyName on page 328	To get the hierarchy path of the object.
IconPath on page 332	To get the path to the symbol representing the object type.
IsLibraryElement on page 345	To check whether the object is a library object.
LibraryLink on page 349	To get the library link of the data object.
ModificationDate on page 358	To get the date of the last object modification.
Name on page 360	To set or get the name of the object.
Parent on page 368	To get the parent object of the specified object.
Protected on page 373	To check whether the object is protected.
ReferenceName on page 378	To set or get the name of the referenced object.
ResultLevel on page 382	To set or get the result level of the specified object.
RootElement on page 385	To get access to the contents of the Dictionary1 object.
RootElement.Count on page 387	To get the number of items in a RootElement object.

Property	Purpose
RootElement.Keys on page 387	To get the keys available in a Dictionary1 or dictionary-based RootElement object.
RootElement.ParentObject on page 388	To get the parent of an item in a RootElement object.
RootElement.RootObject on page 388	To get the parent of a RootElement object.
RootElement.Type on page 389	To get the type of the contents of the RootElement object.
RootElement.Value on page 390	To get the contents of the RootElement object.
StateIconPath on page 394	To get the path to the symbol representing the state of the object.
Synect (Property) on page 400	To get the Synect object.
Type on page 402	To get the type of the specified object.
ValueString on page 406	To set or get the values of the Dictionary1 object as a string.
ValueList on page 406	To get the list of valid values.

Methods

The Dictionary1 object definition contains the following method:

Method	Purpose
CreateSubItem on page 428	To create a subitem that can be added to the root element.
ClearValue on page 418	To clear the values of the data object.
RootElement.Add on page 461	To add an item to a RootElement object.
RootElement.Clear on page 462	To clear the contents of the RootElement object.
RootElement.Contains on page 463	To check whether the specified key is available in the dictionary-based RootElement object.
RootElement.GetItem on page 464	To get an item of the RootElement object.
RootElement.Remove on page 466	To remove the specified item from the RootElement object.
RootElement.SetItem on page 468	To edit the value of an item of a RootElement object.

Events

The Dictionary1 object definition contains the following event:

Event	Purpose
OnModified on page 485	To react to a Dictionary1 being modified.
OnValueChanged on page 496	To react to a Dictionary1 item being changed.

Related topics

Basics

References

File

Object

File

Syntax

No direct creation.

Purpose

To handle a File data object.

Description

The File object is a data object. It stores the absolute path of a file and can be used in automation blocks with a File data object as parameter. The default path of a File object is " ". The File object is either a library template or an instantiated project element.

Properties

The File object definition contains the following properties:

Property	Purpose
Author on page 290	To set or get the name of the person who created the object.
CreationDate on page 314	To get the date the object is created.
Description on page 317	To set or get the description of the object.
HasLibraryLink on page 327	To check whether the data object is linked to the custom library.
HierarchyName on page 328	To get the hierarchy path of the object.
IconPath on page 332	To get the path to the symbol representing the object type.
IsLibraryElement on page 345	To check whether the object is a library object.
LibraryLink on page 349	To get the library link of the data object.
ModificationDate on page 358	To get the date of the last object modification.
Name on page 360	To set or get the name of the object.
Parent on page 368	To get the parent of the specified object.

Property	Purpose
Path on page 369	To set or get the path of the specified object.
Protected on page 373	To check whether the object is protected.
ReferenceName on page 378	To set or get the name of the referenced object.
ResultLevel on page 382	To set or get the result level of the specified object.
StateIconPath on page 394	To get the path to the symbol representing the state of the object.
Type on page 402	To get the type of the specified object.

Methods

None

Events

The File object definition contains the following events:

Event	Purpose
OnModified on page 485	To react to a file being modified.
OnPathChanged on page 486	To react to the path of a File data object being modified.

Related topics

References

DataObject..98

File1

Syntax

No direct creation.

Purpose

To handle a File1 data object.

Description

The File1 object is based on the interface definition of the File object. Additionally, it provides a property to specify whether to interpret the file's Path property as an absolute or a relative path.

The default path of a File1 object is "". By default, the absolute path is used. The File object is either a library template or an instantiated project element.

Properties

The File1 object definition contains the following properties:

Property	Purpose
AbsolutePath on page 288	To set or get the option whether to use the absolute or relative path for the file.
Author on page 290	To set or get the name of the person who created the object.
CreationDate on page 314	To get the date the object was created.
Description on page 317	To set or get the description of the object.
HasLibraryLink on page 327	To check whether the data object is linked to the custom library.
HierarchyName on page 328	To get the hierarchy path of the object.
IconPath on page 332	To get the path to the symbol representing the object type.
IsLibraryElement on page 345	To check whether the object is a library object.
LibraryLink on page 349	To get the library link of the data object.
ModificationDate on page 358	To get the date of the last object modification.
Name on page 360	To set or get the name of the object.
Parent on page 368	To get the parent of the specified object.
Path on page 369	To set or get the path of the specified object.
Protected on page 373	To check whether the object is protected.
ReferenceName on page 378	To set or get the name of the referenced object.
ResultLevel on page 382	To set or get the result level of the specified object.
StateIconPath on page 394	To get the path to the symbol representing the state of the object.
Type on page 402	To get the type of the specified object.

Methods

None

Events

The File1 object definition contains the following events:

Event	Purpose
OnModified on page 485	To react to a file being modified.
OnPathChanged on page 486	To react to the path of a File data object being modified.

Related topics

References

DataObject 98
File 167

File2

Syntax

No direct creation.

Purpose

To handle a File2 data object.

Description

The File2 object is based on the interface definition of the File1 object.

Additionally, it provides a property for you to specify whether to locate the file in the project's attachment folder, i.e., in `<ProjectName>\Attachments` or in the folder that is specified in the file's Path property. The default path of a File2 object is `""`. By default, the Path property is interpreted as the file's absolute path.

The File2 object provides the Synect property, which lets you specify to ignore the object during synchronization with SYNECT.

The File2 object is either a library template or an instantiated project element.

Properties

The File2 object definition contains the following properties:

Property	Purpose
AbsolutePath on page 288	To set or get the option whether to use the absolute or relative path for the file.
Attachment on page 289	To set or get the option whether to locate the file in the project's attachment folder.
Author on page 290	To set or get the name of the person who created the object.
CreationDate on page 314	To get the date the object was created.
Description on page 317	To set or get the description of the object.
HasLibraryLink on page 327	To check whether the data object is linked to the custom library.
HierarchyName on page 328	To get the hierarchy path of the object.
IconPath on page 332	To get the path to the symbol representing the object type.
IsLibraryElement on page 345	To check whether the object is a library object.
LibraryLink on page 349	To get the library link of the data object.
ModificationDate on page 358	To get the date of the last object modification.
Name on page 360	To set or get the name of the object.
Parent on page 368	To get the parent of the specified object.
Path on page 369	To set or get the path of the specified object.
Protected on page 373	To check whether the object is protected.
ReferenceName on page 378	To set or get the name of the referenced object.
ResultLevel on page 382	To set or get the result level of the specified object.
StateIconPath on page 394	To get the path to the symbol representing the state of the object.

Property	Purpose
Synect (Property) on page 400	To get the Synect object.
Type on page 402	To get the type of the specified object.

Methods

None

Events

The File2 object definition contains the following events:

Event	Purpose
OnModified on page 485	To react to a file being modified.
OnPathChanged on page 486	To react to the path of a File data object being modified.

Related topics

References

DataObject........ 98
File........ 167

Float

Object

Syntax

No direct creation.

Purpose

To handle a Float data object.

Description

The Float object is a data object. It stores floating-point numbers. Floating-point numbers are implemented using the C data type double. Their precision depends on the machine and the implementation you are working with. The default value of a Float object is "0.0".

The Float object is either a library template or a project element.

Properties

The Float object definition contains the following properties:

Property	Purpose
Author on page 290	To set or get the name of the person who created the object.
CreationDate on page 314	To get the date the object is created.
Description on page 317	To set or get the description of the object.
HasLibraryLink on page 327	To check whether the data object is linked to the custom library.
HierarchyName on page 328	To get the hierarchy path of the object.
IconPath on page 332	To get the path to the symbol representing the object type.
IsLibraryElement on page 345	To check whether the object is a library object.
LibraryLink on page 349	To get the library link of the data object.
ModificationDate on page 358	To get the date of the last object modification.
Name on page 360	To set or get the name of the object.
Parent on page 368	To get the parent of the specified object.
Protected on page 373	To check whether the object is protected.
ReferenceName on page 378	To set or get the name of the referenced object.
ResultLevel on page 382	To set or get the result level of the specified object.
StateIconPath on page 394	To get the path to the symbol representing the state of the object.
Type on page 402	To get the type of the specified object.
Value on page 404	To set or get a value.

Methods

None

Events

The Float object definition contains the following events:

Event	Purpose
OnModified on page 485	To react to a Float object being modified.
OnValueChanged on page 496	To react to a data object's value being changed.

Related topics

Basics

Float1

Syntax

No direct creation.

Purpose

To handle a Float1 data object.

Description

The Float1 object is a data object. It stores floating-point numbers. Floating-point numbers are implemented using the C data type double. Their precision depends on the machine and the implementation you are working with. The default value of a Float1 object is `"0.0"`. The Float1 object is either a library template or a project element.

The Float1 object is based on the interface definition of the Float object. It additionally provides the ValueList property to get the list of valid values for the data object.

Via the Synect property, you can configure the synchronization with SYNECT.

Properties

The Float1 object definition contains the following properties:

Property	Purpose
Author on page 290	To set or get the name of the person who created the object.
CreationDate on page 314	To get the date the object is created.
Description on page 317	To set or get the description of the object.
HasLibraryLink on page 327	To check whether the data object is linked to the custom library.
HierarchyName on page 328	To get the hierarchy path of the object.
IconPath on page 332	To get the path to the symbol representing the object type.
IsLibraryElement on page 345	To check whether the object is a library object.
LibraryLink on page 349	To get the library link of the data object.
ModificationDate on page 358	To get the date of the last object modification.
Name on page 360	To set or get the name of the object.
Parent on page 368	To get the parent of the specified object.
Protected on page 373	To check whether the object is protected.
ReferenceName on page 378	To set or get the name of the referenced object.
ResultLevel on page 382	To set or get the result level of the specified object.
StateIconPath on page 394	To get the path to the symbol representing the state of the object.
Synect (Property) on page 400	To get the Synect object.
Type on page 402	To get the type of the specified object.
Value on page 404	To set or get a value.
ValueList on page 406	To get the list of valid values.

Methods

None

Events

The Float1 object definition contains the following events:

Event	Purpose
OnModified on page 485	To react to a Float1 object being modified.
OnValueChanged on page 496	To react to a data object's value being changed.

Related topics

Basics

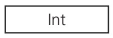
Overview of the Data Types Used..23

References

Float..171

Int

Object



Syntax

No direct creation.

Purpose

To handle an Int data object.

Description

The Int object is a data object. It stores integers. The integers are implemented using the C data type long. The default value for an integer is "0".

The Int object is either a library template or a project element.

Note

The integer value used with the COM API is restricted to 32 bits (long data type). In AutomationDesk, an Int data object is represented by a Python integer with unlimited precision.

Properties

The Int object definition contains the following properties:

Property	Purpose
Author on page 290	To set or get the name of the person who created the object.
CreationDate on page 314	To get the date the object is created.
Description on page 317	To set or get the description of the object.
HasLibraryLink on page 327	To check whether the data object is linked to the custom library.
HierarchyName on page 328	To get the hierarchy path of the object.
IconPath on page 332	To get the path to the symbol representing the object type.
IsLibraryElement on page 345	To check whether the object is a library object.
LibraryLink on page 349	To get the library link of the data object.
ModificationDate on page 358	To get the date of the last object modification.
Name on page 360	To set or get the name of the object.
Parent on page 368	To get the parent of the specified object.
Protected on page 373	To get the information whether the object is protected.
ReferenceName on page 378	To set or get the name of the referenced object.
ResultLevel on page 382	To set or get the result level of the specified object.
StateIconPath on page 394	To get the path to the symbol representing the state of the object.
Type on page 402	To get the type of the specified object.
Value on page 404	To set or get a value.

Methods

None

Events

The Int object definition contains the following events:

Event	Purpose
OnModified on page 485	To react to an Int object being modified.
OnValueChanged on page 496	To react to an Int data object's value being changed.

Related topics

Basics

Overview of the Data Types Used... 23

Int1

Syntax

No direct creation.

Purpose

To handle an Int1 data object.

Description

The Int1 object is a data object. It stores integers. The integers are implemented using the C data type long. The default value for an integer is "0". The Int1 object is either a library template or a project element.

The Int1 object is based on the interface definition of the Int object. It additionally provides the ValueList property to get the list of valid values for the data object.

Via the Synect property, you can configure the synchronization with SYNECT.

Note

The integer value used with the COM API is restricted to 32 bits (long data type). In AutomationDesk, an Int data object is represented by a Python integer with unlimited precision.

Properties

The Int1 object definition contains the following properties:

Property	Purpose
Author on page 290	To set or get the name of the person who created the object.
CreationDate on page 314	To get the date the object is created.
Description on page 317	To set or get the description of the object.
HasLibraryLink on page 327	To check whether the data object is linked to the custom library.
HierarchyName on page 328	To get the hierarchy path of the object.
IconPath on page 332	To get the path to the symbol representing the object type.
IsLibraryElement on page 345	To check whether the object is a library object.
LibraryLink on page 349	To get the library link of the data object.
ModificationDate on page 358	To get the date of the last object modification.
Name on page 360	To set or get the name of the object.
Parent on page 368	To get the parent of the specified object.
Protected on page 373	To get the information whether the object is protected.
ReferenceName on page 378	To set or get the name of the referenced object.
ResultLevel on page 382	To set or get the result level of the specified object.
StateIconPath on page 394	To get the path to the symbol representing the state of the object.
Synect (Property) on page 400	To get the Synect object.

Property	Purpose
Type on page 402	To get the type of the specified object.
Value on page 404	To set or get a value.
ValueList on page 406	To get the list of valid values.

Methods

None

Events

The Int1 object definition contains the following events:

Event	Purpose
OnModified on page 485	To react to an Int1 object being modified.
OnValueChanged on page 496	To react to an Int1 data object's value being changed.

Related topics

Basics

Overview of the Data Types Used.. 23

References

Int.. 174

LabeledValue

Object

LabeledValue

Syntax

No direct creation.

Purpose

To handle a LabeledValue data object.

Description

The LabeledValue object is based on the interface definition of the DataObject2 object.

Additionally, it provides access to the value mapping dictionary that defines the available labels and values for the specific LabeledValue data object. It provides the properties to hold the label and its related value that are currently assigned to the LabeledValue data object. The LabelReferenceName property can hold the name of a String data object that is the reference for the current label.

Properties

The LabeledValue object definition contains the following properties:

Property	Purpose
Author on page 290	To set or get the name of the person who created the object.
CreationDate on page 314	To get the date the object is created.
Description on page 317	To set or get the description of the object.
HasLibraryLink on page 327	To check whether the data object is linked to the custom library.
HierarchyName on page 328	To get the hierarchy path of the object.
Hyperlink on page 329	To get the AutomationDesk hyperlink of the object.
IconPath on page 332	To get the path to the symbol representing the object type.
InOutState on page 338	To set or get the data direction of a data object.
IsLibraryElement on page 345	To check whether the object is a library object.
Label on page 347	To set or get the label of the the object's current value.
LabelReferenceName on page 348	To set or get the name of a reference for the current label.
LibraryLink on page 349	To get the library link of the data object.
Mapping (Property) on page 354	To get the root element of the value mapping dictionary.
ModificationDate on page 358	To get the date of the last object modification.
Name on page 360	To set or get the name of the object.
Parent on page 368	To get the parent of the specified object.
Protected on page 373	To check whether the object is protected.
ReferenceName on page 378	To set or get the name of the referenced object.
ResultLevel on page 382	To set or get the result level of the object.
StateIconPath on page 394	To get the path to the symbol representing the state of the object.
Type on page 402	To set or get the type of the specified object.
Value on page 404	To set or get the current value.

Methods

The LabeledValue object definition contains the following method:

Method	Purpose
Highlight on page 441	To highlight the object's element.

Events

None

Related topics

References

LabeledValue1

Syntax

No direct creation.

Purpose

To handle a LabeledValue1 data object.

Description

The LabelValue1 object is based on the interface definition of the LabeledValue object. It additionally provides the Synect property, which lets you specify to ignore the object during synchronization with SYNECT.

It provides access to the value mapping dictionary that defines the available labels and values for the specific LabeledValue1 data object. It provides the properties to hold the label and its related value that are currently assigned to the LabeledValue1 data object. The LabelReferenceName property can hold the name of a String data object that is the reference for the current label.

Properties

The LabeledValue1 object definition contains the following properties:

Property	Purpose
Author on page 290	To set or get the name of the person who created the object.
CreationDate on page 314	To get the date the object is created.
Description on page 317	To set or get the description of the object.
HasLibraryLink on page 327	To check whether the data object is linked to the custom library.
HierarchyName on page 328	To get the hierarchy path of the object.
Hyperlink on page 329	To get the AutomationDesk hyperlink of the object.
IconPath on page 332	To get the path to the symbol representing the object type.
InOutState on page 338	To set or get the data direction of a data object.
IsLibraryElement on page 345	To check whether the object is a library object.
Label on page 347	To set or get the label of the the object's current value.
LabelReferenceName on page 348	To set or get the name of a reference for the current label.
LibraryLink on page 349	To get the library link of the data object.
Mapping (Property) on page 354	To get the root element of the value mapping dictionary.

Property	Purpose
ModificationDate on page 358	To get the date of the last object modification.
Name on page 360	To set or get the name of the object.
Parent on page 368	To get the parent of the specified object.
Protected on page 373	To check whether the object is protected.
ReferenceName on page 378	To set or get the name of the referenced object.
ResultLevel on page 382	To set or get the result level of the object.
StateIconPath on page 394	To get the path to the symbol representing the state of the object.
Synect (Property) on page 400	To get the Synect object.
Type on page 402	To set or get the type of the specified object.
Value on page 404	To set or get the current value.

Methods

The LabeledValue1 object definition contains the following method:

Method	Purpose
Highlight on page 441	To highlight the object's element.

Events

None

Related topics

References

DataObject2 99
Edit Value Mapping (AutomationDesk Basic Practices 📖)
LabeledValue (AutomationDesk Basic Practices 📖)

List

Object

List

Syntax

No direct creation.

Purpose

To handle a List data object.

Description

The List object is only the top level instance of a List data object in your AutomationDesk project. Like in AutomationDesk, it provides a root element that let you access the list's contents. For information on the root element's properties and methods, refer to the RootElement property.

Properties

The List object definition contains the following properties:

Property	Purpose
Author on page 290	To set or get the name of the person who created the object.
CreationDate on page 314	To get the date the object is created.
Description on page 317	To set or get the description of the object.
HierarchyName on page 328	To get the hierarchy path of the object.
HasLibraryLink on page 327	To check whether the data object is linked to the custom library.
IconPath on page 332	To get the path to the symbol representing the object type.
IsLibraryElement on page 345	To check whether the object is a library object.
LibraryLink on page 349	To get the library link of the data object.
ModificationDate on page 358	To get the date of the last object modification.
Name on page 360	To set or get the name of the object.
Parent on page 368	To get the parent object of the specified object.
Protected on page 373	To check whether the object is protected.
ReferenceName on page 378	To set or get the name of the referenced object.
ResultLevel on page 382	To set or get the result level of the specified object.
RootElement on page 385	To get access to the contents of the List object.
RootElement.Count on page 387	To get the number of items in a RootElement object.
RootElement.ParentObject on page 388	To get the parent of an item in a RootElement object.
RootElement.RootObject on page 388	To get the parent of a RootElement object.
RootElement.Type on page 389	To get the type of the contents of the RootElement object.
RootElement.Value on page 390	To get the contents of the RootElement object.
StateIconPath on page 394	To get the path to the symbol representing the state of the object.
Type on page 402	To get the type of the specified object.
ValueString on page 406	To set or get the values of the List object as a string.

Methods

The List object definition contains the following method:

Method	Purpose
CreateSubItem on page 428	To create a subitem that can be added to the root element.
ClearValue on page 418	To clear the values of the data object.
RootElement.Add on page 461	To add an item to a RootElement object.
RootElement.Clear on page 462	To clear the contents of the RootElement object.

Method	Purpose
RootElement.GetItem on page 464	To get an item of the RootElement object.
RootElement.IndexOf on page 465	To get the first index of the specified value in the RootElement object.
RootElement.Insert on page 466	To add an item in the List.RootElement object at a specific position.
RootElement.Remove on page 466	To remove the specified item from the RootElement object.
RootElement.RemoveAt on page 467	To remove an item from the given position in the List.RootElement object.
RootElement.SetItem on page 468	To edit the value of an item of a RootElement object.

Events

The List object definition contains the following event:

Event	Purpose
OnModified on page 485	To react to a List being modified.
OnValueChanged on page 496	To react to a List item being changed.

List1

Syntax

No direct creation.

Purpose

To handle a List1 data object.

Description

The List1 object is only the top level instance of a List data object in your AutomationDesk project. Like in AutomationDesk, it provides a root element that let you access the list's contents. For information on the root element's properties and methods, refer to the RootElement property.

The List1 object is based on the interface definition of the List object. It additionally provides the ValueList property to get the list of valid values for the data object.

Via the Synect property, you can configure the synchronization with SYNECT.

Properties

The List1 object definition contains the following properties:

Property	Purpose
Author on page 290	To set or get the name of the person who created the object.
CreationDate on page 314	To get the date the object is created.
Description on page 317	To set or get the description of the object.
HierarchyName on page 328	To get the hierarchy path of the object.

Property	Purpose
HasLibraryLink on page 327	To check whether the data object is linked to the custom library.
IconPath on page 332	To get the path to the symbol representing the object type.
IsLibraryElement on page 345	To check whether the object is a library object.
LibraryLink on page 349	To get the library link of the data object.
ModificationDate on page 358	To get the date of the last object modification.
Name on page 360	To set or get the name of the object.
Parent on page 368	To get the parent object of the specified object.
Protected on page 373	To check whether the object is protected.
ReferenceName on page 378	To set or get the name of the referenced object.
ResultLevel on page 382	To set or get the result level of the specified object.
RootElement on page 385	To get access to the contents of the List1 object.
RootElement.Count on page 387	To get the number of items in a RootElement object.
RootElement.ParentObject on page 388	To get the parent of an item in a RootElement object.
RootElement.RootObject on page 388	To get the parent of a RootElement object.
RootElement.Type on page 389	To get the type of the contents of the RootElement object.
RootElement.Value on page 390	To get the contents of the RootElement object.
StateIconPath on page 394	To get the path to the symbol representing the state of the object.
Synect (Property) on page 400	To get the Synect object.
Type on page 402	To get the type of the specified object.
ValueString on page 406	To set or get the values of the List1 object as a string.
ValueList on page 406	To get the list of valid values.

Methods

The List1 object definition contains the following method:

Method	Purpose
CreateSubItem on page 428	To create a subitem that can be added to the root element.
ClearValue on page 418	To clear the values of the data object.
RootElement.Add on page 461	To add an item to a RootElement object.
RootElement.Clear on page 462	To clear the contents of the RootElement object.
RootElement.GetItem on page 464	To get an item of the RootElement object.
RootElement.IndexOf on page 465	To get the first index of the specified value in the RootElement object.
RootElement.Insert on page 466	To add an item in the List1.RootElement object at a specific position.
RootElement.Remove on page 466	To remove the specified item from the RootElement object.
RootElement.RemoveAt on page 467	To remove an item from the given position in the List1.RootElement object.
RootElement.SetItem on page 468	To edit the value of an item of a RootElement object.

Events

The List1 object definition contains the following event:

Event	Purpose
OnModified on page 485	To react to a List1 being modified.
OnValueChanged on page 496	To react to a List1 item being changed.

Related topics

Basics

Overview of the Data Types Used..23

References

List..180
RootElement...385

String

Object

Syntax

No direct creation.

Purpose

To handle a specific String data object.

Description

The String object is a data object that stores arbitrary text. String literals are written in single or double quotes. The backslash is used to avoid characters that otherwise have a special meaning, such as newline or quotes. The default is an empty string.

Properties

The String object definition contains the following properties:

Property	Purpose
Author on page 290	To set or get the name of the person who created the object.
CreationDate on page 314	To get the date the object is created.
Description on page 317	To set or get the description of the object.
HasLibraryLink on page 327	To check whether the data object is linked to the custom library.
HierarchyName on page 328	To get the hierarchy path of the object.

Property	Purpose
IconPath on page 332	To get the path to the symbol representing the object type.
IsLibraryElement on page 345	To check whether the object is a library object.
LibraryLink on page 349	To get the library link of the data object.
ModificationDate on page 358	To get the date of the last object modification.
Name on page 360	To set or get the name of the object.
Parent on page 368	To get the parent of the specified object.
Protected on page 373	To check whether the object is protected.
ResultLevel on page 382	To set or get the result level of the specified object.
StateIconPath on page 394	To get the path to the symbol representing the state of the object.
Type on page 402	To get the type of the specified object.
Value on page 404	To set or get a value.

Methods

None

Events

The String object definition contains the following event:

Event	Purpose
OnModified on page 485	To react to a String being modified.
OnValueChanged on page 496	To react to a String item being changed.

Related topics

Basics

Overview of the Data Types Used..23

String1

Syntax

No direct creation.

Purpose

To handle a specific String1 data object.

Description

The String1 object is a data object that stores arbitrary text. String literals are written in single or double quotes. The backslash is used to avoid characters that otherwise have a special meaning, such as newline or quotes. The default is an empty string.

The String1 object is based on the interface definition of the String object. It additionally provides the ValueList property to get the list of valid values for the data object.

Via the Synect property, you can configure the synchronization with SYNECT.

Properties

The String1 object definition contains the following properties:

Property	Purpose
Author on page 290	To set or get the name of the person who created the object.
CreationDate on page 314	To get the date the object is created.
Description on page 317	To set or get the description of the object.
HasLibraryLink on page 327	To check whether the data object is linked to the custom library.
HierarchyName on page 328	To get the hierarchy path of the object.
IconPath on page 332	To get the path to the symbol representing the object type.
IsLibraryElement on page 345	To check whether the object is a library object.
LibraryLink on page 349	To get the library link of the data object.
ModificationDate on page 358	To get the date of the last object modification.
Name on page 360	To set or get the name of the object.
Parent on page 368	To get the parent of the specified object.
Protected on page 373	To check whether the object is protected.
ResultLevel on page 382	To set or get the result level of the specified object.
StateIconPath on page 394	To get the path to the symbol representing the state of the object.
Synect (Property) on page 400	To get the Synect object.
Type on page 402	To get the type of the specified object.
Value on page 404	To set or get a value.
ValueList on page 406	To get the list of valid values.

Methods

None

Events

The String1 object definition contains the following events:

Event	Purpose
OnModified on page 485	To react to a String1 object being modified.
OnValueChanged on page 496	To react to a String1 data object's value being changed.

Related topics

Basics

Overview of the Data Types Used........ 23

References

String........ 184

Tuple

Object

Tuple

Syntax

No direct creation.

Purpose

To handle a Tuple data object.

Description

The Tuple object is only the top level instance of a Tuple data object in your AutomationDesk project. Like in AutomationDesk, it provides a root element that let you access the tuple's contents. For information on the root element's properties and methods, refer to the RootElement property.

Properties

The Tuple object definition contains the following properties:

Property	Purpose
Author on page 290	To set or get the name of the person who created the object.
CreationDate on page 314	To get the date the object is created.
Description on page 317	To set or get the description of the object.
HasLibraryLink on page 327	To check whether the data object is linked to the custom library.
HierarchyName on page 328	To get the hierarchy path of the object.
IconPath on page 332	To get the path to the symbol representing the object type.
IsLibraryElement on page 345	To check whether the object is a library object.
LibraryLink on page 349	To get the library link of the data object.
ModificationDate on page 358	To get the library link of the data object.
Name on page 360	To set or get the name of the object.
Parent on page 368	To get the parent object of the specified object.

Property	Purpose
Protected on page 373	To check whether the object is protected.
ReferenceName on page 378	To set or get the name of the referenced object.
ResultLevel on page 382	To set or get the result level of the specified object.
RootElement on page 385	To get access to the contents of the Tuple object.
RootElement.Count on page 387	To get the number of items in a RootElement object.
RootElement.ParentObject on page 388	To get the parent of an item in a RootElement object.
RootElement.RootObject on page 388	To get the parent of a RootElement object.
RootElement.Type on page 389	To get the type of the contents of the RootElement object.
RootElement.Value on page 390	To get the contents of the RootElement object.
StateIconPath on page 394	To get the path to the symbol representing the state of the object.
Type on page 402	To get the type of the specified object.
ValueString on page 406	To set or get the values of the Tuple object as a string.

Methods

The Tuple object definition contains the following method:

Method	Purpose
CreateSubItem on page 428	To create a subitem that can be added to the root element.
ClearValue on page 418	To clear the values of the data object.
RootElement.GetItem on page 464	To get an item of the RootElement object.
RootElement.IndexOf on page 465	To get the first index of the specified value in the RootElement object.

Events

The Tuple object definition contains the following event:

Event	Purpose
OnModified on page 485	To react to a property being modified.

Tuple1

Syntax

No direct creation.

Purpose

To handle a Tuple11 data object.

Description

The Tuple1 object is only the top level instance of a Tuple data object in your AutomationDesk project. Like in AutomationDesk, it provides a root element that let you access the tuple's contents. For information on the root element's properties and methods, refer to the RootElement property.

The Tuple1 object is based on the interface definition of the Tuple object. It additionally provides the ValueList property to get the list of valid values for the data object.

Via the Synect property, you can configure the synchronization with SYNECT.

Properties

The Tuple1 object definition contains the following properties:

Property	Purpose
Author on page 290	To set or get the name of the person who created the object.
CreationDate on page 314	To get the date the object is created.
Description on page 317	To set or get the description of the object.
HasLibraryLink on page 327	To check whether the data object is linked to the custom library.
HierarchyName on page 328	To get the hierarchy path of the object.
IconPath on page 332	To get the path to the symbol representing the object type.
IsLibraryElement on page 345	To check whether the object is a library object.
LibraryLink on page 349	To get the library link of the data object.
ModificationDate on page 358	To get the library link of the data object.
Name on page 360	To set or get the name of the object.
Parent on page 368	To get the parent object of the specified object.
Protected on page 373	To check whether the object is protected.
ReferenceName on page 378	To set or get the name of the referenced object.
ResultLevel on page 382	To set or get the result level of the specified object.
RootElement on page 385	To get access to the contents of the Tuple1 object.
RootElement.Count on page 387	To get the number of items in a RootElement object.
RootElement.ParentObject on page 388	To get the parent of an item in a RootElement object.
RootElement.RootObject on page 388	To get the parent of a RootElement object.
RootElement.Type on page 389	To get the type of the contents of the RootElement object.
RootElement.Value on page 390	To get the contents of the RootElement object.
StateIconPath on page 394	To get the path to the symbol representing the state of the object.
Synect (Property) on page 400	To get the Synect object.
Type on page 402	To get the type of the specified object.
ValueString on page 406	To set or get the values of the Tuple1 object as a string.
ValueList on page 406	To get the list of valid values.

Methods

The Tuple1 object definition contains the following method:

Method	Purpose
CreateSubItem on page 428	To create a subitem that can be added to the root element.
ClearValue on page 418	To clear the values of the data object.
RootElement.GetItem on page 464	To get an item of the RootElement object.
RootElement.IndexOf on page 465	To get the first index of the specified value in the RootElement object.

Events

The Tuple1 object definition contains the following event:

Event	Purpose
OnModified on page 485	To react to a property being modified.

Related topics

Basics

Overview of the Data Types Used.. 23

References

RootElement.. 385
Tuple.. 187

Variant

Object

Variant

Syntax

No direct creation.

Purpose

To handle a Variant data object.

Description

The Variant object is a data object. It can be of any type.

The Variant object is either a library template or a project element.

Properties

The Variant object definition contains the following properties:

Property	Purpose
Author on page 290	To set or get the name of the person who created the object.
CreationDate on page 314	To get the date the object is created.
Description on page 317	To set or get the description of the object.
HasLibraryLink on page 327	To check whether the data object is linked to the custom library.
HierarchyName on page 328	To get the hierarchy path of the object.
IconPath on page 332	To get the path to the symbol representing the object type.
IsLibraryElement on page 345	To check whether the object is a library object.
LibraryLink on page 349	To get the library link of the data object.
ModificationDate on page 358	To get the date of the last object modification.
Name on page 360	To set or get the name of the object.
Parent on page 368	To get the parent of the specified object.
Protected on page 373	To check whether the object is protected.
ResultLevel on page 382	To set or get the result level of the specified object.
StateIconPath on page 394	To get the path to the symbol representing the state of the object.
Type on page 402	To get the type of the specified object.
Value on page 404	To set or get a value.

Methods

The Variant object definition contains the following method:

Method	Purpose
ClearValue on page 418	To clear the values of the data object.

Events

The Variant object definition contains the following event:

Event	Purpose
OnModified on page 485	To react to a Variant being modified.
OnValueChanged on page 496	To react to a Variant item being changed.

Related topics

Basics

Overview of the Data Types Used..23

Variant1

Syntax

No direct creation.

Purpose

To handle a Variant1 data object.

Description

The Variant1 object is a data object. It can be of any type. The Variant1 object is either a library template or a project element.

The Variant1 object is based on the interface definition of the Variant object. It additionally provides the ValueList property to get the list of valid values for the data object.

Via the Synect property, you can configure the synchronization with SYNECT.

Properties

The Variant1 object definition contains the following properties:

Property	Purpose
Author on page 290	To set or get the name of the person who created the object.
CreationDate on page 314	To get the date the object is created.
Description on page 317	To set or get the description of the object.
HasLibraryLink on page 327	To check whether the data object is linked to the custom library.
HierarchyName on page 328	To get the hierarchy path of the object.
IconPath on page 332	To get the path to the symbol representing the object type.
IsLibraryElement on page 345	To check whether the object is a library object.
LibraryLink on page 349	To get the library link of the data object.
ModificationDate on page 358	To get the date of the last object modification.
Name on page 360	To set or get the name of the object.
Parent on page 368	To get the parent of the specified object.
Protected on page 373	To check whether the object is protected.
ResultLevel on page 382	To set or get the result level of the specified object.
StateIconPath on page 394	To get the path to the symbol representing the state of the object.
Synect (Property) on page 400	To get the Synect object.
Type on page 402	To get the type of the specified object.
Value on page 404	To set or get a value.
ValueList on page 406	To get the list of valid values.

Methods

The Variant1 object definition contains the following method:

Method	Purpose
ClearValue on page 418	To clear the values of the data object.

Events

The Variant1 object definition contains the following event:

Event	Purpose
OnModified on page 485	To react to a Variant1 being modified.
OnValueChanged on page 496	To react to a Variant1 item being changed.

Related topics

Basics

Overview of the Data Types Used.. 23

References

Variant... 190

Verdict (Object)

Object

Verdict

Syntax

No direct creation.

Purpose

To handle a Verdict data object.

Description

The Verdict data object is used to provide a verdict for automation elements which return a verdict or use a verdict as input. Verdicts are used to qualify the test execution. The values of a verdict are of Integer data type (long).

Properties

The Verdict object definition contains the following properties:

Property	Purpose
Author on page 290	To set or get the name of the person who created the object.
CreationDate on page 314	To get the date the object is created.
Description on page 317	To set or get the description of the object.
HasLibraryLink on page 327	To check whether the data object is linked to the custom library.
HierarchyName on page 328	To get the hierarchy path of the object.
IconPath on page 332	To get the path to the symbol representing the object type.
IsLibraryElement on page 345	To check whether the object is a library object.
LibraryLink on page 349	To get the library link of the data object.
ModificationDate on page 358	To get the date of the last object modification.
Name on page 360	To set or get the name of the object.
Parent on page 368	To get the parent of the specified object.
Protected on page 373	To check whether the object is protected.
ReferenceName on page 378	To set or get the name of the referenced object.
ResultLevel on page 382	To set or get the result level of the specified object.
StateIconPath on page 394	To get the path to the symbol representing the state of the object.
Type on page 402	To get the type of the specified object.
Value on page 404	To set or get a value of type VerdictConstant.

Methods

None

Events

The Verdict object definition contains the following event:

Event	Purpose
OnModified on page 485	To react to a Verdict being modified.
OnValueChanged on page 496	To react to a Verdict item being changed.

Related topics

Basics

Main Library.. 153

Verdict1 (Object)

Syntax

No direct creation.

Purpose

To handle a Verdict1 data object.

Description

The Verdict object is based on the interface definition of the Verdict (Object) object. It additionally provides the Synect property, which lets you specify to ignore the object during synchronization with SYNECT.

The Verdict1 data object is used to provide a verdict for automation elements which return a verdict or use a verdict as input. Verdicts are used to qualify the test execution. The values of a verdict are of Integer data type (long).

Properties

The Verdict1 object definition contains the following properties:

Property	Purpose
Author on page 290	To set or get the name of the person who created the object.
CreationDate on page 314	To get the date the object is created.
Description on page 317	To set or get the description of the object.
HasLibraryLink on page 327	To check whether the data object is linked to the custom library.
HierarchyName on page 328	To get the hierarchy path of the object.
IconPath on page 332	To get the path to the symbol representing the object type.
IsLibraryElement on page 345	To check whether the object is a library object.
LibraryLink on page 349	To get the library link of the data object.
ModificationDate on page 358	To get the date of the last object modification.
Name on page 360	To set or get the name of the object.
Parent on page 368	To get the parent of the specified object.
Protected on page 373	To check whether the object is protected.
ReferenceName on page 378	To set or get the name of the referenced object.
ResultLevel on page 382	To set or get the result level of the specified object.
StateIconPath on page 394	To get the path to the symbol representing the state of the object.
Synect (Property) on page 400	To get the Synect object.
Type on page 402	To get the type of the specified object.
Value on page 404	To set or get a value of type VerdictConstant.

Methods

None

Events

The Verdict1 object definition contains the following event:

Event	Purpose
OnModified on page 485	To react to a Verdict1 being modified.
OnValueChanged on page 496	To react to a Verdict1 item being changed.

Related topics

Basics

MATLAB Access

Where to go from here

Information in this section

MATLAB

Syntax

No direct creation.

Purpose

To handle 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 open and close this MATLAB instance by using the **`Open`** and **`Close`** methods. The MATLAB data object provides the **`ConvertToDouble`** property. By default, no data type conversion for integer types is done.

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.

Properties

The MATLAB object definition contains the following properties:

Property	Purpose
Author on page 290	To set or get the name of the person who created the object.
ConvertToDouble on page 311	To set or get the conversion mode for integer values of the MATLAB object.
CreationDate on page 314	To get the date the object is created.
Description on page 317	To set or get the description of the object.
HasLibraryLink on page 327	To check whether the data object is linked to the custom library.
HierarchyName on page 328	To get the hierarchy path of the object.
IconPath on page 332	To get the path to the symbol representing the object type.
IsLibraryElement on page 345	To check whether the object is a library object.
LibraryLink on page 349	To get the library link of the data object.
ModificationDate on page 358	To get the date of the last object modification.
Name on page 360	To set or get the name of the object.
Parent on page 368	To get the parent of the specified object.
Protected on page 373	To check whether the object is protected.
ReferenceName on page 378	To set or get the name of the referenced object.
ResultLevel on page 382	To set or get the result level of the specified object.
StateIconPath on page 394	To get the path to the symbol representing the state of the object.
Type on page 402	To get the type of the specified object.

Methods

The MATLAB object definition contains the following methods:

Method	Purpose
Close on page 421	To close a MATLAB instance.
Open on page 455	To open a MATLAB instance.

Events

The MATLAB object definition contains the following events:

Event	Purpose
OnModified on page 485	To react to a MATLAB object being modified.

Related topics

References

MATFile

Syntax

No direct creation.

Purpose

To handle a MAT file.

Description

The MATFile data object is used to handle a MAT file used in your sequence for reading and writing according to the specified file access mode.

Properties

The MATFile object definition contains the following properties:

Property	Purpose
AbsolutePath on page 288	To set or get the option whether to use the absolute or relative path for the MAT file.
Author on page 290	To set or get the name of the person who created the object.
AvailableModes on page 297	To get a list of the available values for the Mode property.
ConvertToDouble on page 311	To set or get the conversion mode for integer values of the MATFile object.
CreationDate on page 314	To get the date the object is created.
Description on page 317	To set or get the description of the object.
FileName on page 324	To set or get the path and name of the instantiated MAT file.
HasLibraryLink on page 327	To check whether the data object is linked to the custom library.
HierarchyName on page 328	To get the hierarchy path of the object.
IconPath on page 332	To get the path to the symbol representing the object type.
IsLibraryElement on page 345	To check whether the object is a library object.
LibraryLink on page 349	To get the library link of the data object.
Mode on page 357	To set or get the file access mode of a MATFile object.
ModificationDate on page 358	To get the date of the last object modification.
Name on page 360	To set or get the name of the object.
Parent on page 368	To get the parent of the specified object.
Protected on page 373	To check whether the object is protected.
ReferenceName on page 378	To set or get the name of the referenced object.

Property	Purpose
ResultLevel on page 382	To set or get the result level of the specified object.
StateIconPath on page 394	To get the path to the symbol representing the state of the object.
Type on page 402	To get the type of the specified object.

Methods

None

Events

The MATFile object definition contains the following events:

Event	Purpose
OnModified on page 485	To react to a MATFile object being modified.

Related topics

References

MATLAB Access.. 196

Remote Calibration (COM)

Where to go from here

Information in this section

MC3System

Syntax

No direct creation.

Purpose

To handle a MC3System data object.

Description

The MC3System object is a data object. It stores the configuration of the connection to an MC system including the IP address of the host and the name of the interface. If the connection is configured correctly, the MC3System object is used to establish the connection, to check the status of the connection, and to disconnect again. The MC3System object is either a library template or an instantiated project element.

Properties

The MC3System object definition contains the following properties:

Property	Purpose
Author on page 290	To set or get the name of the person who created the object.
AvailableInterfaceNames on page 296	To get the available interface names for connecting to a MC system.
CreationDate on page 314	To get the date the object is created.
Description on page 317	To set or get the description of the object.
HasLibraryLink on page 327	To check whether the data object is linked to the custom library.
HierarchyName on page 328	To get the hierarchy path of the object.
Host on page 329	To set or get the host of the MC3System object.
IconPath on page 332	To get the path to the symbol representing the object type.
Interface on page 339	To set or get the interface of the MC3System object.
IsConnected on page 341	To get the status of the connection to the MC3System object.
IsLibraryElement on page 345	To check whether the object is a library object.
LibraryLink on page 349	To get the library link of the data object.
ModificationDate on page 358	To get the date of the last object modification.
Name on page 360	To set or get the name of the object.
Parent on page 368	To get the parent of the specified object.
Projects (Property) on page 372	To get the projects of the MC3System object.

Property	Purpose
Protected on page 373	To check whether the object is protected.
ResultLevel on page 382	To set or get the result level of the specified object.
StateIconPath on page 394	To get the path to the symbol representing the state of the object.
Type on page 402	To get the type of the specified object.

Methods

The MC3System object definition contains the following methods:

Method	Purpose
Connect on page 423	To connect AutomationDesk with the configured MC system.
Disconnect on page 431	To disconnect AutomationDesk from the configured MC system.

Events

The MC3System object definition contains the following event:

Event	Purpose
OnModified on page 485	To react to an MC3System property being modified.

Related topics

References

Remote Calibration (COM)... 199

MC3Project

Syntax

No direct creation.

Purpose

To handle a MC3Project data object.

Description

The MC3Project object is a data object. It stores the configuration of the calibration project to be used. If AutomationDesk is connected to the MC system, you can get a list of all the available calibration projects. If you have specified the calibration project, the MC3Project object is used to select it, to check the status of the project selection, and to deselect it again. The MC3Project object is either a library template or an instantiated project element.

Properties

The MC3Project object definition contains the following properties:

Property	Purpose
Author on page 290	To set or get the name of the person who created the object.
AvailableProjectNames on page 299	To get the available interface names for connecting to a MC system.
CreationDate on page 314	To get the date the object is created.
Description on page 317	To set or get the description of the object.
HasLibraryLink on page 327	To check whether the data object is linked to the custom library.
HierarchyName on page 328	To get the hierarchy path of the object.
IconPath on page 332	To get the path to the symbol representing the object type.
IsLibraryElement on page 345	To check whether the object is a library object.
IsSelected on page 345	To get the status of the project selection.
LibraryLink on page 349	To get the library link of the data object.
LogicalLinks on page 351	To get the logical links of the MC3Project object.
ModificationDate on page 358	To get the date of the last object modification.
Name on page 360	To set or get the name of the object.
Parent on page 368	To get the parent of the specified object.
ProjectName on page 371	To set or get the name of the calibration project.
Protected on page 373	To check whether the object is protected.
ReferenceName on page 378	To set or get the name of the referenced object.
ResultLevel on page 382	To set or get the result level of the specified object.
StateIconPath on page 394	To get the path to the symbol representing the state of the object.
Type on page 402	To get the type of the specified object.

Methods

The MC3Project object definition contains the following methods:

Method	Purpose
DeSelect on page 430	To deselect the currently selected calibration project.
Select on page 472	To select a calibration project from the connected MC system.

Events

The MC3Project object definition contains the following event:

Event	Purpose
OnModified on page 485	To react to an MC3Project property being modified.

Related topics

References

MC3LogicalLink

Syntax

No direct creation.

Purpose

To handle a LogicalLink data object of the Remote Calibration COM library.

Description

The MC3LogicalLink object is a data object. It specifies the connection settings to the physical ECU board. The MC3LogicalLink object is either a library template or an instantiated project element.

Properties

The MC3LogicalLink object definition contains the following properties:

Property	Purpose
Author on page 290	To set or get the name of the person who created the object.
AvailableBinaryFileNames on page 291	To get the names of the available binary files.
AvailableLogicalLinkNames on page 296	To get the names of the available LogicalLinks.
BinaryName on page 303	To set or get the name of the binary file.
Characteristics on page 307	To get the Characteristics data container of the MC3LogicalLink object.
Collectors on page 309	To get the Collectors data container of the MC3LogicalLink object.
CreationDate on page 314	To get the date the object is created.
Description on page 317	To set or get the description of the object.
HasLibraryLink on page 327	To check whether the data object is linked to the custom library.
HierarchyName on page 328	To get the hierarchy path of the object.
IconPath on page 332	To get the path to the symbol representing the object type.
IsLibraryElement on page 345	To check whether the object is a library object.
LibraryLink on page 349	To get the library link of the data object.
LogicalLinkName on page 351	To set or get the name of the logical link.
ModificationDate on page 358	To get the date of the last object modification.
Name on page 360	To set or get the name of the object.
Parent on page 368	To get the parent of the specified object.
Protected on page 373	To check whether the object is protected.

Property	Purpose
ReferenceName on page 378	To set or get the name of the referenced object.
ResultLevel on page 382	To set or get the result level of the specified object.
StateIconPath on page 394	To get the path to the symbol representing the state of the object.
Type on page 402	To get the type of the specified object.

Methods

None

Events

The MC3LogicalLink object definition contains the following event:

Event	Purpose
OnModified on page 485	To react to an MC3LogicalLink property being modified.

Related topics

References

MC3Characteristics

Syntax

No direct creation.

Purpose

To handle a Characteristic data object of the Remote Calibration COM library.

Description

The MC3Characteristic object is a data object. It stores the configuration of the characteristic to be used. If AutomationDesk is connected to the MC system, a project and a logical link is selected, you can get a list of all the available characteristics. The MC3Characteristic object is either a library template or an instantiated project element.

Properties

The MC3Characteristic object definition contains the following properties:

Property	Purpose
Author on page 290	To set or get the name of the person who created the object.
AvailableCharacteristicTypeNames on page 293	To get the available characteristic type names.
AvailableRepresentationTypeNames on page 299	To get the available characteristic type names.

Property	Purpose
AvailableValueTypeNames on page 301	To get the available value type names.
CharacteristicName on page 307	To set or get the name of the characteristic object.
CharacteristicType on page 308	To get the available interface names for connecting to a MC system.
CreationDate on page 314	To get the date the object is created.
Description on page 317	To set or get the description of the object.
HasLibraryLink on page 327	To check whether the data object is linked to the custom library.
HierarchyName on page 328	To get the hierarchy path of the object.
IconPath on page 332	To get the path to the symbol representing the object type.
IsLibraryElement on page 345	To check whether the object is a library object.
LibraryLink on page 349	To get the library link of the data object.
ModificationDate on page 358	To get the date of the last object modification.
Name on page 360	To set or get the name of the object.
Parent on page 368	To get the parent of the specified object.
Protected on page 373	To check whether the object is protected.
ReferenceName on page 378	To set or get the name of the referenced object.
RepresentationType on page 381	To set or get the representation type of the characteristic object.
ResultLevel on page 382	To set or get the result level of the specified object.
StateIconPath on page 394	To get the path to the symbol representing the state of the object.
Type on page 402	To get the type of the specified object.
ValueType on page 407	To set or get the type of the characteristic value.
XStartIndex on page 411	To set or get the start index of the x-axis.
XStopIndex on page 411	To set or get the stop index of the x-axis.
YStartIndex on page 413	To set or get the start index of the y-axis.
YStopIndex on page 413	To set or get the stop index of the y-axis.

Methods None

Events The MC3Characteristic object definition contains the following event:

Event	Purpose
OnModified on page 485	To react to an MC3Characteristic property being modified.

Related topics

References

Remote Calibration (COM).. 199

MC3Collector

Syntax

No direct creation.

Purpose

To handle a Collector data object of the Remote Calibration COM library.

Description

The MC3Collector object is a data object. It stores the configuration of the collector to be used. The MC3Collector object is either a library template or an instantiated project element.

Properties

The MC3Collector object definition contains the following properties:

Property	Purpose
AbsolutePath on page 288	To set or get the option whether to use the absolute or relative path for the result file.
Author on page 290	To set or get the name of the person who created the object.
AvailableBufferRateNames on page 292	To get the names of the available buffer rates.
AvailableRepresentationTypeNames on page 299	To get the names of the available representation types.
AvailableStorageTypeNames on page 301	To get the names of the available storage types.
BufferRate on page 304	To set or get the buffer rate of the MC3Collector object.
BufferSize on page 305	To set or get the buffer size of the MC3Collector object.
CreationDate on page 314	To get the date the object is created.
Description on page 317	To set or get the description of the object.
DownSampling on page 318	To set or get the downsampling rate of the MC3Collector object.
FileName on page 324	To set or get the file name to store the collector results in.
HasLibraryLink on page 327	To check whether the data object is linked to the custom library.
HierarchyName on page 328	To get the hierarchy path of the object.
IconPath on page 332	To get the path to the symbol representing the object type.
IsLibraryElement on page 345	To check whether the object is a library object.
LibraryLink on page 349	To get the library link of the data object.
MeasurementVariables on page 356	To get the measurement variables of the MC3Collector object.
ModificationDate on page 358	To get the date of the last object modification.
Name on page 360	To set or get the name of the object.
NumberOfSamples on page 362	To set or get the number of samples of the MC3Collector object.
Parent on page 368	To get the parent of the specified object.
Protected on page 373	To check whether the object is protected.
ReferenceName on page 378	To set or get the name of the referenced object.
RepresentationType on page 381	To set or get the representation type of the MC3Collector object.

Property	Purpose
ResultLevel on page 382	To set or get the result level of the specified object.
StateIconPath on page 394	To get the path to the symbol representing the state of the object.
StorageType on page 397	To set or get the storage type of the MC3Collector object.
Type on page 402	To get the type of the specified object.

Methods

The MC3Collector object definition contains the following method:

Method	Purpose
ClearValue on page 418	To clear the values of the data object.

Events

The MC3Collector object definition contains the following event:

Event	Purpose
OnModified on page 485	To react to an MC3Collector property being modified.

Related topics

References

Remote Calibration (COM)... 199

MC3Measurement

Syntax

No direct creation.

Purpose

To handle a Measurement data object of the Remote Calibration COM library.

Description

The MC3Measurement object is a data object. It specifies a new measurement for the current calibration project. It always belong to a Collector object and can be created from the MeasurementVariables collection that you get by the Collector's MeasurementVariables property. The MC3Measurement object is either a library template or an instantiated project element.

Properties

The MC3Measurement object definition contains the following properties:

Property	Purpose
Author on page 290	To set or get the name of the person who created the object.
CreationDate on page 314	To get the date the object is created.
Description on page 317	To set or get the description of the object.
HasLibraryLink on page 327	To look up whether the data object ís linked to the custom library.
HierarchyName on page 328	To get the hierarchy path of the object.
IconPath on page 332	To get the path to the symbol representing the object type.
IsLibraryElement on page 345	To check whether the object is a library object.
LibraryLink on page 349	To get the library link of the data object.
MeasurementName on page 356	To set or get the name of the measurement accessed by the MC3Measurement object.
ModificationDate on page 358	To get the date of the last object modification.
Name on page 360	To set or get the name of the object.
Parent on page 368	To get the parent of the specified object.
Protected on page 373	To check whether the object is protected.
ReferenceName on page 378	To set or get the name of the referenced object.
ResultLevel on page 382	To set or get the result level of the specified object.
StateIconPath on page 394	To get the path to the symbol representing the state of the object.
Type on page 402	To get the type of the specified object.

Methods

None

Events

The MC3Measurement object definition contains the following event:

Event	Purpose
OnModified on page 485	To react to an MC3Measurement property being modified.

Related topics

References

MC3Collector..206
MeasurementVariables..356
Remote Calibration (COM)..199

Remote Diagnostics (COM)

Where to go from here

Information in this section

D3System

Syntax

No direct creation.

Purpose

To handle a System data object of the Remote Diagnostics (COM) library.

Description

The D3System object is a data object. It stores the configuration of the connection to a diagnostic system including the IP address of the host and the name of the interface. If the connection is configured correctly, the D3System object is used to establish the connection, to check the status of the connection, and to disconnect again. The D3System object is either a library template or an instantiated project element.

You can use ControlDesk with ControlDesk ECU Diagnostics module to perform diagnostic tasks based on ASAM MCD-3 D 2.0.2.

Properties

The D3System object definition contains the following properties:

Property	Purpose
Author on page 290	To set or get the name of the person who created the object.
AvailableInterfaceNames on page 296	To get the available interface names for connecting to a diagnostic system.
CreationDate on page 314	To get the date the object is created.
Description on page 317	To set or get the description of the object.
HasLibraryLink on page 327	To check whether the data object is linked to the custom library.
HierarchyName on page 328	To get the hierarchy path of the object.
Host on page 329	To set or get the host of the D3System object.
IconPath on page 332	To get the path to the symbol representing the object type.
Interface on page 339	To set or get the interface of the D3System object.
IsConnected on page 341	To get the status of the connection to the diagnostic system.
IsLibraryElement on page 345	To check whether the object is a library object.
LibraryLink on page 349	To get the library link of the data object.
ModificationDate on page 358	To get the date of the last object modification.
Name on page 360	To set or get the name of the object.
Parent on page 368	To get the parent of the specified object.
Projects (Property) on page 372	To get the projects of the D3System object.
Protected on page 373	To check whether the object is protected.
ResultLevel on page 382	To set or get the result level of the specified object.
StateIconPath on page 394	To get the path to the symbol representing the state of the object.
Type on page 402	To get the type of the specified object.

Methods

The D3System object definition contains the following methods:

Method	Purpose
Connect on page 423	To connect AutomationDesk with the configured diagnostic system.
Disconnect on page 431	To disconnect AutomationDesk from the configured diagnostic system.

Events

The D3System object definition contains the following events:

Event	Purpose
OnModified on page 485	To react to a D3System being modified.

Related topics

References

D3Project

Syntax

No direct creation.

Purpose

To handle the Project data object from the Remote Diagnostics (COM) library.

Description

The D3Project object is a data object. It stores the configuration of the diagnostic project to be used. If AutomationDesk is connected to the diagnostic system, you can get a list of all the available diagnostic projects. If you have specified the diagnostic project, the D3Project object is used to select it, to check the status of the project selection, and to deselect it again. The D3Project object is either a library template or an instantiated project element.

Properties

The D3Project object definition contains the following properties:

Property	Purpose
Author on page 290	To set or get the name of the person who created the object.
AvailableProjectNames on page 299	To get the available project names from the connected diagnostic system.
CreationDate on page 314	To get the date the object is created.
Description on page 317	To set or get the description of the object.
HasLibraryLink on page 327	To check whether the data object is linked to the custom library.
HierarchyName on page 328	To get the hierarchy path of the object.
IconPath on page 332	To get the path to the symbol representing the object type.
IsLibraryElement on page 345	To check whether the object is a library object.
IsSelected on page 345	To get the status of the project selection.
LibraryLink on page 349	To get the library link of the data object.
ModificationDate on page 358	To get the date of the last object modification.

Property	Purpose
Name on page 360	To set or get the name of the object.
Parent on page 368	To get the parent of the specified object.
ProjectName on page 371	To set or get the name of the diagnostic project.
Protected on page 373	To check whether the object is protected.
ResultLevel on page 382	To set or get the result level of the specified object.
ReferenceName on page 378	To set or get the name of the referenced object.
StateIconPath on page 394	To get the path to the symbol representing the state of the object.
Type on page 402	To get the type of the specified object.
VehicleInformations on page 408	To get the VehicleInformations collection object of the D3Project object.

Methods

The D3Project object definition contains the following methods:

Method	Purpose
DeSelect on page 430	To deselect the currently selected diagnostic project.
Select on page 472	To select a diagnostic project from the connected diagnostic system.

Events

The D3Project object definition contains the following events:

Event	Purpose
OnModified on page 485	To react to a D3Project object being modified.

Related topics

References

Remote Diagnostics (COM).. 209

D3VehicleInformation

Syntax

No direct creation.

Purpose

To handle a VehicleInformation data object of the Remote Diagnostics (COM) library.

Description

The D3VehicleInformation object is a data object. It describes which ECUs are installed in the vehicle. The VehicleInformation data object can contain one ore more LogicalLink data objects. The D3VehicleInformation object is either a library template or an instantiated project element.

Properties

The D3VehicleInformation object definition contains the following properties:

Property	Purpose
Author on page 290	To set or get the name of the person who created the object.
AvailableVehicleInformationNames on page 302	To get the available VehicleInformation names from the selected diagnostic project.
CreationDate on page 314	To get the date the object is created.
Description on page 317	To set or get the description of the object.
HasLibraryLink on page 327	To check whether the data object is linked to the custom library.
HierarchyName on page 328	To get the hierarchy path of the object.
IconPath on page 332	To get the path to the symbol representing the object type.
IsLibraryElement on page 345	To check whether the object is a library object.
LibraryLink on page 349	To get the library link of the data object.
LogicalLinks on page 351	To get the LogicalLinks collection object of the D3VehicleInformation object.
ModificationDate on page 358	To get the date of the last object modification.
Name on page 360	To set or get the name of the object.
Parent on page 368	To get the parent of the specified object.
Protected on page 373	To check whether the object is protected.
ReferenceName on page 378	To set or get the name of the referenced object.
ResultLevel on page 382	To set or get the result level of the specified object.
StateIconPath on page 394	To get the path to the symbol representing the state of the object.
Type on page 402	To get the type of the specified object.
VehicleInformationName on page 408	To set or get the name of the instantiated VehicleInformation object.

Methods

None

Events

The D3VehicleInformation object definition contains the following event:

Event	Purpose
OnModified on page 485	To react to a D3VehicleInformation object being modified.

Related topics

References

D3LogicalLink

Syntax

No direct creation.

Purpose

To handle a LogicalLink data object of the Remote Diagnostics (COM) library.

Description

The D3LogicalLink object is a data object. It specifies the connection settings to the physical ECU board. The D3LogicalLink object is either a library template or an instantiated project element.

Properties

The D3LogicalLink object definition contains the following properties:

Property	Purpose
Author on page 290	To set or get the name of the person who created the object.
AvailableLogicalLinkNames on page 296	To get the names of the available LogicalLinks.
ControlPrimitives on page 311	To get the ControlPrimitives collection object of the D3LogicalLink object.
CreationDate on page 314	To get the date the object is created.
Description on page 317	To set or get the description of the object.
HasLibraryLink on page 327	To check whether the data object is linked to the custom library.
HierarchyName on page 328	To get the hierarchy path of the object.
IconPath on page 332	To get the path to the symbol representing the object type.
IsLibraryElement on page 345	To check whether the object is a library object.
LibraryLink on page 349	To get the library link of the data object.
LogicalLinkName on page 351	To set or get the name of the logical link.
ModificationDate on page 358	To get the date of the last object modification.
Name on page 360	To set or get the name of the object.
Parent on page 368	To get the parent of the specified object.
Protected on page 373	To check whether the object is protected.
ReferenceName on page 378	To set or get the name of the referenced object.
ResultLevel on page 382	To set or get the result level of the specified object.
Services on page 392	To get the Services collection object of the D3LogicalLink object.

Property	Purpose
SingleJobs on page 393	To get the SingleJobs collection object of the D3LogicalLink object.
StateIconPath on page 394	To get the path to the symbol representing the state of the object.
Type on page 402	To get the type of the specified object.

Methods

None

Events

The D3LogicalLink object definition contains the following event:

Event	Purpose
OnModified on page 485	To react to a D3LogicalLink object being modified.

Related topics

References

Remote Diagnostics (COM)..209

D3ControlPrimitive

Syntax

No direct creation.

Purpose

To handle a ControlPrimitive data object of the Remote Diagnostics (COM) library.

Description

The D3ControlPrimtive object is a data object. It describes the diagnostic parameters for communication with the ECU. After setting up a connection to the diagnostic tool, all parameters of the ControlPrimitive data object are available in a parameter list and can be customized.

Properties

The D3ControlPrimitive object definition contains the following properties:

Property	Purpose
Author on page 290	To set or get the name of the person who created the object.
AvailableControlPrimitiveNames on page 293	To get the names of the available ControlPrimitives.
ControlPrimitiveName on page 310	To set or get the name of the ControlPrimitive.
CreationDate on page 314	To get the date the object is created.

Property	Purpose
Description on page 317	To set or get the description of the object.
HasLibraryLink on page 327	To check whether the data object is linked to the custom library.
HierarchyName on page 328	To get the hierarchy path of the object.
IconPath on page 332	To get the path to the symbol representing the object type.
IsLibraryElement on page 345	To check whether the object is a library object.
LibraryLink on page 349	To get the library link of the data object.
ModificationDate on page 358	To get the date of the last object modification.
Name on page 360	To set or get the name of the object.
Parameters on page 367	To set or get the diagnostic parameters of the D3ControlPrimitive object.
Parent on page 368	To get the parent of the specified object.
Protected on page 373	To check whether the object is protected.
ReferenceName on page 378	To set or get the name of the referenced object.
ResultLevel on page 382	To set or get the result level of the specified object.
StateIconPath on page 394	To get the path to the symbol representing the state of the object.
Type on page 402	To get the type of the specified object.

Methods

The D3ControlPrimitive object definition contains the following methods:

Method	Purpose
AddParameter on page 415	To add a parameter to a D3ControlPrimitive object.
DeleteParameter on page 429	To delete a parameter from the parameter list of the D3ControlPrimitive object.
EditParameter on page 432	To edit a parameter of the D3ControlPrimitive object.
GetParameterDefaultValues on page 439	To get the default values of a Parameter object.
GetParameterValue on page 440	To get the value and unit of the specified parameter.
SetParameterValue on page 473	To set the parameter value of the specified Parameter object.

Events

The D3ControlPrimitive object definition contains the following event:

Event	Purpose
OnModified on page 485	To react to a D3ControlPrimitive object being modified.

Related topics

References

Remote Diagnostics (COM).. 209

D3Service

Syntax

No direct creation.

Purpose

To handle a Service data object of the Remote Diagnostics (COM) library.

Description

The D3Service object is a data object. It describes the diagnostic parameters for communication with the ECU. After setting up a connection to the diagnostic tool, all parameters of the Service data object are available in a parameter list and can be customized.

Properties

The D3Service object definition contains the following properties:

Property	Purpose
Author on page 290	To set or get the name of the person who created the object.
AvailableFunctionalClassNames on page 294	To get a list of the available functional class names.
AvailableServiceNames on page 300	To get a list of the available services from the selected D3Service object.
CreationDate on page 314	To get the date the object is created.
Description on page 317	To set or get the description of the object.
FunctionalClassName on page 325	To set or get a functional class.
HasLibraryLink on page 327	To check whether the data object is linked to the custom library.
HierarchyName on page 328	To get the hierarchy path of the object.
IconPath on page 332	To get the path to the symbol representing the object type.
IsLibraryElement on page 345	To check whether the object is a library object.
LibraryLink on page 349	To get the library link of the data object.
ModificationDate on page 358	To get the date of the last object modification.
Name on page 360	To set or get the name of the object.
Parameters on page 367	To set or get the diagnostic parameters of the D3Service object.
Parent on page 368	To get the parent of the specified object.
Protected on page 373	To check whether the object is protected.
ReferenceName on page 378	To set or get the name of the referenced object.
ResultLevel on page 382	To set or get the result level of the specified object.
ServiceName on page 391	To set or get the name of a service.
StateIconPath on page 394	To get the path to the symbol representing the state of the object.
Type on page 402	To get the type of the specified object.

Methods

The D3Service object definition contains the following methods:

Method	Purpose
AddParameter on page 415	To add a parameter to a D3Service object.
DeleteParameter on page 429	To delete a parameter from the parameter list of the D3Service object.
EditParameter on page 432	To edit a parameter of the D3Service object.
GetParameterDefaultValues on page 439	To get the default values of a Parameter object.
GetParameterValue on page 440	To get the value and unit of the specified parameter.
SetParameterValue on page 473	To set the parameter value of the specified Parameter object.

Events

The D3Service object definition contains the following event:

Event	Purpose
OnModified on page 485	To react to a D3Service object being modified.

Related topics

References

Remote Diagnostics (COM).. 209

D3SingleJob

Syntax

No direct creation.

Purpose

To handle a SingleJob data object of the Remote Diagnostics (COM) library.

Description

The D3SingleJob object is a data object. It describes the diagnostic parameters for communication with the ECU. After setting up a connection to the diagnostic tool, all parameters of the SingelJob data object are available in a parameter list and can be customized.

Properties

The D3SingleJob object definition contains the following properties:

Property	Purpose
Author on page 290	To set or get the name of the person who created the object.
AvailableSingleJobNames on page 300	To get a list of the available single jobs from the selected D3SingleJob object.
CreationDate on page 314	To get the date the object is created.
Description on page 317	To set or get the description of the object.
HasLibraryLink on page 327	To check whether the data object is linked to the custom library.
HierarchyName on page 328	To get the hierarchy path of the object.
IconPath on page 332	To get the path to the symbol representing the object type.
IsLibraryElement on page 345	To check whether the object is a library object.
LibraryLink on page 349	To get the library link of the data object.
ModificationDate on page 358	To get the date of the last object modification.
Name on page 360	To set or get the name of the object.
Parameters on page 367	To set or get the diagnostic parameters of the D3SingleJob object.
Parent on page 368	To get the parent of the specified object.
Protected on page 373	To check whether the object is protected.
ReferenceName on page 378	To set or get the name of the referenced object.
ResultLevel on page 382	To set or get the result level of the specified object.
SingleJobName on page 392	To set or get the name of a single job.
StateIconPath on page 394	To get the path to the symbol representing the state of the object.
Type on page 402	To get the type of the specified object.

Methods

The D3SingleJob object definition contains the following methods:

Method	Purpose
AddParameter on page 415	To add a parameter to a D3SingleJob object.
DeleteParameter on page 429	To delete a parameter from the parameter list of the D3SingleJob object.
EditParameter on page 432	To edit a parameter of the D3SingleJob object.
GetParameterDefaultValues on page 439	To get the default values of a Parameter object.
GetParameterValue on page 440	To get the value and unit of the specified parameter.
SetParameterValue on page 473	To set the parameter value of the specified Parameter object.

Events

The D3SingleJob object definition contains the following event:

Event	Purpose
OnModified on page 485	To react to a D3SingleJob object being modified.

Related topics

References

D3Results

Syntax

No direct creation.

Purpose

To handle a Results data object of the Remote Diagnostics (COM) library.

Description

The D3Results object is a data object. It contains the diagnostic results that are delivered by the diagnostic tool. These results are available during the execution of the project. After the execution, the Results data object is set to *None* again.

Properties

The D3Results object definition contains the following properties:

Property	Purpose
Author on page 290	To set or get the name of the person who created the object.
CreationDate on page 314	To get the date the object is created.
Description on page 317	To set or get the description of the object.
HasLibraryLink on page 327	To check whether the data object is linked to the custom library.
HierarchyName on page 328	To get the hierarchy path of the object.
IconPath on page 332	To get the path to the symbol representing the object type.
IsLibraryElement on page 345	To check whether the object is a library object.
LibraryLink on page 349	To get the library link of the data object.
ModificationDate on page 358	To get the date of the last object modification.
Name on page 360	To set or get the name of the object.
Parent on page 368	To get the parent of the specified object.
Protected on page 373	To check whether the object is protected.
ReferenceName on page 378	To set or get the name of the referenced object.
ResultLevel on page 382	To set or get the result level of the specified object.
StateIconPath on page 394	To get the path to the symbol representing the state of the object.
Type on page 402	To get the type of the specified object.

Methods

The D3Results object definition contains the following method:

Method	Purpose
ClearValue on page 418	To clear the values of the data object.

Events

The D3Results object definition contains the following event:

Event	Purpose
OnModified on page 485	To react to a D3Results object being modified.

Related topics

References

Report

Color

Syntax

No direct creation.

Purpose

To handle a Color data object.

Description

The Color object is a data object. It stores the color in RGB format as a tuple. It can be used in automation blocks with a Color data object as parameter.

Properties

The Color object definition contains the following properties:

Property	Purpose
Author on page 290	To set or get the name of the person who created the object.
Blue on page 304	To set or get the blue portion of the Color object.
CreationDate on page 314	To get the date the object is created.
Description on page 317	To set or get the description of the object.
Green on page 326	To set or get the green portion of the Color object.

Property	Purpose
HasLibraryLink on page 327	To check whether the data object is linked to the custom library.
IconPath on page 332	To get the path to the symbol representing the object type.
IsLibraryElement on page 345	To check whether the object is a library object.
LibraryLink on page 349	To get the library link of the data object.
ModificationDate on page 358	To get the date of the last object modification.
Name on page 360	To set or get the name of the object.
Parent on page 368	To get the parent of the specified object.
Protected on page 373	To check whether the object is protected.
Red on page 377	To set or get the red portion of the Color object.
ResultLevel on page 382	To set or get the result level of the specified object.
StateIconPath on page 394	To get the path to the symbol representing the state of the object.
Type on page 402	To get the type of the specified object.
Value on page 404	To set or get the color of the Color object in form of a tuple or a string.

Methods

None

Events

The Color object definition contains the following event:

Event	Purpose
OnModified on page 485	To react to a color being modified.

Related topics

References

Report..221
Tuple...187

RS232

RS232Configuration

Syntax

No direct creation.

Purpose

To handle a RS232Configuration data object.

Description

The RS232Configuration object is a data object. It stores the configuration of the serial interface including the PC port, the baud rate, the number of data and stop bits, the parity scheme and the sizes of the input and output buffers. The RS232Configuration object is either a library template or an instantiated project element.

Properties

The RS232Configuration object definition contains the following properties:

Property	Purpose
Author on page 290	To set or get the name of the person who created the object.
AvailableBitsPerSecond on page 292	To get a list of the available values for the BitsPerSecond property.
AvailableDataBits on page 294	To get a list of the available values for the DataBits property.
AvailableInBufferSize on page 295	To get a list of the available values for the InBufferSize property.
AvailableOutBufferSize on page 297	To get a list of the available values for the OutBufferSize property.
AvailableParity on page 298	To get a list of the available values for the Parity property.
AvailablePorts on page 298	To get a list of the available values for the Ports property.
AvailableStopBits on page 300	To get a list of the available values for the StopBits property.
BitsPerSecond on page 303	To set or get the baud rate value in bits per second.
CreationDate on page 314	To get the date the object is created.
DataBits on page 316	To set or get the number of data bits used for the RS232 interface.
Description on page 317	To set or get the description of the object.
HasLibraryLink on page 327	To check whether the data object is linked to the custom library.
HierarchyName on page 328	To get the hierarchy path of the object.
IconPath on page 332	To get the path to the symbol representing the object type.
InBufferSize on page 334	To set or get the size of the input buffer of the RS232 interface.
IsLibraryElement on page 345	To check whether the object is a library object.
LibraryLink on page 349	To get the library link of the data object.
ModificationDate on page 358	To get the date of the last object modification.
Name on page 360	To set or get the name of the object.
OutBufferSize on page 365	To set or get the size of the output buffer of the RS232 interface.
Parent on page 368	To get the parent of the specified object.
Parity on page 368	To set or get the parity scheme of the RS232 interface.
Port on page 371	To set or get the serial port of the PC used for the RS232 interface.
Protected on page 373	To check whether the object is protected.
ReferenceName on page 378	To set or get the name of the referenced object.
ResultLevel on page 382	To set or get the result level of the specified object.
StateIconPath on page 394	To get the path to the symbol representing the state of the object.

Property	Purpose
StopBits on page 396	To set or get the number of stop bits used for the RS232 interface.
Type on page 402	To get the type of the specified object.

Methods

None

Events

The RS232Configuration object definition contains the following events:

Event	Purpose
OnModified on page 485	To react to a RS232Configuration being modified.

Related topics

References

XIL API

Where to go from here

Information in this section

Attributes

Syntax

No direct creation.

Purpose

To handle an Attributes data object.

Description

An Attributes object is a data object. It is used to provide additional information for an instantiated value.

Properties

The Attributes object definition contains the following properties:

Property	Purpose
Author on page 290	To set or get the name of the person who created the object.
CreationDate on page 314	To get the date the object is created.
Description on page 317	To set or get the description of the object.
HasLibraryLink on page 327	To check whether the data object is linked to the custom library.
HierarchyName on page 328	To get the hierarchy path of the object.
IconPath on page 332	To get the path to the symbol representing the object type.
InOutState on page 338	To set or get the data direction of a data object.
IsLibraryElement on page 345	To check whether the object is a library object.
LibraryLink on page 349	To get the library link of the data object.
ModificationDate on page 358	To get the date of the last object modification.
Name on page 360	To set or get the name of the object.
Parent on page 368	To get the parent of the specified object.
Protected on page 373	To check whether the object is protected.
ReferenceName on page 378	To set or get the name of the referenced object.
ResultLevel on page 382	To set or get the result level of the object.
StateIconPath on page 394	To get the path to the symbol representing the state of the object.
Type on page 402	To set or get the type of the specified object.

Methods

None

Events

The Attributes object definition contains the following event:

Event	Purpose
OnModified on page 485	To react to a property being modified.

Related topics

References

Attributes (Data Object) (AutomationDesk Accessing Simulation Platforms 📖)

BaseError

Syntax

No direct creation.

Purpose

To handle a BaseError data object.

Description

A BaseError data object is used to get information on a specific error in the error set. An error is basically specified by its error category, error type and load.

Properties

The BaseError object definition contains the following properties:

Property	Purpose
Author on page 290	To set or get the name of the person who created the object.
CreationDate on page 314	To get the date the object is created.
Description on page 317	To set or get the description of the object.
HasLibraryLink on page 327	To check whether the data object is linked to the custom library.
HierarchyName on page 328	To get the hierarchy path of the object.
IconPath on page 332	To get the path to the symbol representing the object type.
InOutState on page 338	To set or get the data direction of a data object.
IsLibraryElement on page 345	To check whether the object is a library object.
LibraryLink on page 349	To get the library link of the data object.
ModificationDate on page 358	To get the date of the last object modification.
Name on page 360	To set or get the name of the object.
Parent on page 368	To get the parent of the specified object.
Protected on page 373	To check whether the object is protected.
ReferenceName on page 378	To set or get the name of the referenced object.
ResultLevel on page 382	To set or get the result level of the object.
StateIconPath on page 394	To get the path to the symbol representing the state of the object.
Type on page 402	To set or get the type of the specified object.

Methods

None

Events

The BaseError object definition contains the following event:

Event	Purpose
OnModified on page 485	To react to a property being modified.

Related topics

References

BaseError (Data Object) (AutomationDesk Simulating Electrical Errors 📖)

BaseErrorBuilder

Syntax

No direct creation.

Purpose

To handle a BaseErrorBuilder data object.

Description

The BaseErrorBuilder data object is used to provide an error builder for errors of BaseError type, such as simple errors (default), dynamic errors or resistor errors.

Properties

The BaseErrorBuilder object definition contains the following properties:

Property	Purpose
Author on page 290	To set or get the name of the person who created the object.
CreationDate on page 314	To get the date the object is created.
Description on page 317	To set or get the description of the object.
HasLibraryLink on page 327	To check whether the data object is linked to the custom library.
HierarchyName on page 328	To get the hierarchy path of the object.
IconPath on page 332	To get the path to the symbol representing the object type.
InOutState on page 338	To set or get the data direction of a data object.
IsLibraryElement on page 345	To check whether the object is a library object.
LibraryLink on page 349	To get the library link of the data object.
ModificationDate on page 358	To get the date of the last object modification.
Name on page 360	To set or get the name of the object.
Parent on page 368	To get the parent of the specified object.
Protected on page 373	To check whether the object is protected.
ReferenceName on page 378	To set or get the name of the referenced object.

Property	Purpose
ResultLevel on page 382	To set or get the result level of the object.
StateIconPath on page 394	To get the path to the symbol representing the state of the object.
Type on page 402	To set or get the type of the specified object.

Methods

None

Events

The BaseErrorBuilder object definition contains the following event:

Event	Purpose
OnModified on page 485	To react to a property being modified.

Related topics

References

BaseErrorBuilder (Data Object) (AutomationDesk Simulating Electrical Errors 🕮)

BaseValue

Syntax

No direct creation.

Purpose

To handle a specific BaseValue data object.

Description

A BaseValue object is a data object. It handles physical values, for example, a parameter.

Properties

The BaseValue object definition contains the following properties:

Property	Purpose
Author on page 290	To set or get the name of the person who created the object.
CreationDate on page 314	To get the date the object is created.
Description on page 317	To set or get the description of the object.
HasLibraryLink on page 327	To check whether the data object is linked to the custom library.
HierarchyName on page 328	To get the hierarchy path of the object.
IconPath on page 332	To get the path to the symbol representing the object type.

Property	Purpose
InOutState on page 338	To set or get the data direction of a data object.
IsLibraryElement on page 345	To check whether the object is a library object.
LibraryLink on page 349	To get the library link of the data object.
ModificationDate on page 358	To get the date of the last object modification.
Name on page 360	To set or get the name of the object.
Parent on page 368	To get the parent of the specified object.
Protected on page 373	To check whether the object is protected.
ReferenceName on page 378	To set or get the name of the referenced object.
ResultLevel on page 382	To set or get the result level of the object.
StateIconPath on page 394	To get the path to the symbol representing the state of the object.
Type on page 402	To set or get the type of the specified object.

Methods

None

Events

The BaseValue object definition contains the following event:

Event	Purpose
OnModified on page 485	To react to a property being modified.

Related topics

References

BaseValue (Data Object) (AutomationDesk Accessing Simulation Platforms 🕮)

Capture

Syntax

No direct creation.

Purpose

To handle a specific Capture data object.

Description

A Capture object is a data object. It configures the variables to be captured, the trigger conditions, the duration of the measurement and the real-time model task.

Properties

The Capture object definition contains the following properties:

Property	Purpose
Author on page 290	To set or get the name of the person who created the object.
CreationDate on page 314	To get the date the object is created.
Description on page 317	To set or get the description of the object.
HasLibraryLink on page 327	To check whether the data object is linked to the custom library.
HierarchyName on page 328	To get the hierarchy path of the object.
IconPath on page 332	To get the path to the symbol representing the object type.
InOutState on page 338	To set or get the data direction of a data object.
IsLibraryElement on page 345	To check whether the object is a library object.
LibraryLink on page 349	To get the library link of the data object.
ModificationDate on page 358	To get the date of the last object modification.
Name on page 360	To set or get the name of the object.
Parent on page 368	To get the parent of the specified object.
Protected on page 373	To check whether the object is protected.
ReferenceName on page 378	To set or get the name of the referenced object.
ResultLevel on page 382	To set or get the result level of the object.
StateIconPath on page 394	To get the path to the symbol representing the state of the object.
Type on page 402	To set or get the type of the specified object.

Methods

None

Events

The Capture object definition contains the following event:

Event	Purpose
OnModified on page 485	To react to a property being modified.

Related topics

References

Capture (Data Object) (AutomationDesk Accessing Simulation Platforms 🕮)

CaptureResult (XIL API)

Syntax	No direct creation.
Purpose	To handle a specific CaptureResult data object.
Description	A CaptureResult object is a data object. It holds one time axis and at least one measurement of a variable.

Properties

The CaptureResult object definition contains the following properties:

Property	Purpose
Author on page 290	To set or get the name of the person who created the object.
CreationDate on page 314	To get the date the object is created.
Description on page 317	To set or get the description of the object.
HasLibraryLink on page 327	To check whether the data object is linked to the custom library.
HierarchyName on page 328	To get the hierarchy path of the object.
IconPath on page 332	To get the path to the symbol representing the object type.
InOutState on page 338	To set or get the data direction of a data object.
IsLibraryElement on page 345	To check whether the object is a library object.
LibraryLink on page 349	To get the library link of the data object.
ModificationDate on page 358	To get the date of the last object modification.
Name on page 360	To set or get the name of the object.
Parent on page 368	To get the parent of the specified object.
Protected on page 373	To check whether the object is protected.
ReferenceName on page 378	To set or get the name of the referenced object.
ResultLevel on page 382	To set or get the result level of the object.
StateIconPath on page 394	To get the path to the symbol representing the state of the object.
Type on page 402	To set or get the type of the specified object.

Methods

The CaptureResult object definition contains the following method:

Method	Purpose
ClearValue on page 418	To clear the values of the data object.

Events

The CaptureResult object definition contains the following event:

Event	Purpose
OnModified on page 485	To react to a property being modified.

Related topics

References

CaptureResult (Data Object) (AutomationDesk Accessing Simulation Platforms 🕮)

CaptureResultReader

Syntax

No direct creation.

Purpose

To handle a specific CaptureResultReader data object.

Description

A CaptureResultReader object is a data object. It holds an instantiated CaptureResultReader object, for example, a CaptureResultMDFReader.

Properties

The CaptureResultReader object definition contains the following properties:

Property	Purpose
Author on page 290	To set or get the name of the person who created the object.
CreationDate on page 314	To get the date the object is created.
Description on page 317	To set or get the description of the object.
HasLibraryLink on page 327	To check whether the data object is linked to the custom library.
HierarchyName on page 328	To get the hierarchy path of the object.
IconPath on page 332	To get the path to the symbol representing the object type.
InOutState on page 338	To set or get the data direction of a data object.
IsLibraryElement on page 345	To check whether the object is a library object.
LibraryLink on page 349	To get the library link of the data object.
ModificationDate on page 358	To get the date of the last object modification.
Name on page 360	To set or get the name of the object.
Parent on page 368	To get the parent of the specified object.
Protected on page 373	To check whether the object is protected.
ReferenceName on page 378	To set or get the name of the referenced object.

Property	Purpose
ResultLevel on page 382	To set or get the result level of the object.
StateIconPath on page 394	To get the path to the symbol representing the state of the object.
Type on page 402	To set or get the type of the specified object.

Methods

None

Events

The CaptureResultReader object definition contains the following event:

Event	Purpose
OnModified on page 485	To react to a property being modified.

Related topics

References

CaptureResultReader (Data Object) (AutomationDesk Accessing Simulation Platforms 📖)

CaptureResultWriter

Syntax

No direct creation.

Purpose

To handle a specific CaptureResultWriter data object.

Description

A CaptureResultWriter object is a data object. It provides a file writer for a capture result.

Properties

The CaptureResultWriter object definition contains the following properties:

Property	Purpose
Author on page 290	To set or get the name of the person who created the object.
CreationDate on page 314	To get the date the object is created.
Description on page 317	To set or get the description of the object.
HasLibraryLink on page 327	To check whether the data object is linked to the custom library.
HierarchyName on page 328	To get the hierarchy path of the object.
IconPath on page 332	To get the path to the symbol representing the object type.

Property	Purpose
InOutState on page 338	To set or get the data direction of a data object.
IsLibraryElement on page 345	To check whether the object is a library object.
LibraryLink on page 349	To get the library link of the data object.
ModificationDate on page 358	To get the date of the last object modification.
Name on page 360	To set or get the name of the object.
Parent on page 368	To get the parent of the specified object.
Protected on page 373	To check whether the object is protected.
ReferenceName on page 378	To set or get the name of the referenced object.
ResultLevel on page 382	To set or get the result level of the object.
StateIconPath on page 394	To get the path to the symbol representing the state of the object.
Type on page 402	To set or get the type of the specified object.

Methods

None

Events

The CaptureResultWriter object definition contains the following event:

Event	Purpose
OnModified on page 485	To react to a property being modified.

Related topics

References

CaptureResultWriter (Data Object) (AutomationDesk Accessing Simulation Platforms 📖)

CaptureState

Syntax

No direct creation.

Purpose

To handle a specific CaptureState data object.

Description

A CaptureState object is a data object. It indicates the state of a capture as an enumeration element.

Tip

You can find a capturing state diagram ("state diagram of capturing") in the ASAM documentation ASAM_AE_XIL_Generic-Simulator-Interface_BS-1-4-Programmers-Guide_V2-1-0.pdf at `C:\Program Files\Common Files\dSPACE\HelpDesk <ReleaseNumber>\Print`.

Properties

The CaptureState object definition contains the following properties:

Property	Purpose
Author on page 290	To set or get the name of the person who created the object.
CreationDate on page 314	To get the date the object is created.
Description on page 317	To set or get the description of the object.
HasLibraryLink on page 327	To check whether the data object is linked to the custom library.
HierarchyName on page 328	To get the hierarchy path of the object.
IconPath on page 332	To get the path to the symbol representing the object type.
InOutState on page 338	To set or get the data direction of a data object.
IsLibraryElement on page 345	To check whether the object is a library object.
LibraryLink on page 349	To get the library link of the data object.
ModificationDate on page 358	To get the date of the last object modification.
Name on page 360	To set or get the name of the object.
Parent on page 368	To get the parent of the specified object.
Protected on page 373	To check whether the object is protected.
ReferenceName on page 378	To set or get the name of the referenced object.
ResultLevel on page 382	To set or get the result level of the object.
StateIconPath on page 394	To get the path to the symbol representing the state of the object.
Type on page 402	To set or get the type of the specified object.

Methods

None

Events

The CaptureState object definition contains the following event:

Event	Purpose
OnModified on page 485	To react to a property being modified.

Related topics

References

CaptureState (Data Object) (AutomationDesk Accessing Simulation Platforms 🕮)

CapturingFactory

Syntax

No direct creation.

Purpose

To handle a CapturingFactory data object.

Description

A CapturingFactory object is a data object. It is used to instantiate a Capture object and to create further objects required for capturing.

Properties

The CapturingFactory object definition contains the following properties:

Property	Purpose
Author on page 290	To set or get the name of the person who created the object.
CreationDate on page 314	To get the date the object is created.
Description on page 317	To set or get the description of the object.
HasLibraryLink on page 327	To check whether the data object is linked to the custom library.
HierarchyName on page 328	To get the hierarchy path of the object.
IconPath on page 332	To get the path to the symbol representing the object type.
InOutState on page 338	To set or get the data direction of a data object.
IsLibraryElement on page 345	To check whether the object is a library object.
LibraryLink on page 349	To get the library link of the data object.
ModificationDate on page 358	To get the date of the last object modification.
Name on page 360	To set or get the name of the object.
Parent on page 368	To get the parent of the specified object.
Protected on page 373	To check whether the object is protected.
ReferenceName on page 378	To set or get the name of the referenced object.
ResultLevel on page 382	To set or get the result level of the object.
StateIconPath on page 394	To get the path to the symbol representing the state of the object.
Type on page 402	To set or get the type of the specified object.

Methods

None

Events

The CapturingFactory object definition contains the following event:

Event	Purpose
OnModified on page 485	To react to a property being modified.

Related topics

References

CapturingFactory (Data Object) (AutomationDesk Accessing Simulation Platforms 🕮)

DataType

Syntax

No direct creation.

Purpose

To handle a specific DataType data object.

Description

A DataType object is a data object. It indicates the data type of a variable value as an enumeration element.

Properties

The DataType object definition contains the following properties:

Property	Purpose
Author on page 290	To set or get the name of the person who created the object.
CreationDate on page 314	To get the date the object is created.
Description on page 317	To set or get the description of the object.
HasLibraryLink on page 327	To check whether the data object is linked to the custom library.
HierarchyName on page 328	To get the hierarchy path of the object.
IconPath on page 332	To get the path to the symbol representing the object type.
InOutState on page 338	To set or get the data direction of a data object.
IsLibraryElement on page 345	To check whether the object is a library object.
LibraryLink on page 349	To get the library link of the data object.
ModificationDate on page 358	To get the date of the last object modification.
Name on page 360	To set or get the name of the object.
Parent on page 368	To get the parent of the specified object.
Protected on page 373	To check whether the object is protected.
ReferenceName on page 378	To set or get the name of the referenced object.

Property	Purpose
ResultLevel on page 382	To set or get the result level of the object.
StateIconPath on page 394	To get the path to the symbol representing the state of the object.
Type on page 402	To set or get the type of the specified object.

Methods

None

Events

The DataType object definition contains the following event:

Event	Purpose
OnModified on page 485	To react to a property being modified.

Related topics

References

DataType (Data Object) (AutomationDesk Accessing Simulation Platforms 🕮)

EESConfigurationReader

Syntax

No direct creation.

Purpose

To handle a specific EESConfigurationReader data object.

Description

A EESConfigurationReader object is a data object. It is used to provide a file reader object for loading an already existing error configuration file.

Properties

The EESConfigurationReader object definition contains the following properties:

Property	Purpose
Author on page 290	To set or get the name of the person who created the object.
CreationDate on page 314	To get the date the object is created.
Description on page 317	To set or get the description of the object.
HasLibraryLink on page 327	To check whether the data object is linked to the custom library.
HierarchyName on page 328	To get the hierarchy path of the object.
IconPath on page 332	To get the path to the symbol representing the object type.

Property	Purpose
InOutState on page 338	To set or get the data direction of a data object.
IsLibraryElement on page 345	To check whether the object is a library object.
LibraryLink on page 349	To get the library link of the data object.
ModificationDate on page 358	To get the date of the last object modification.
Name on page 360	To set or get the name of the object.
Parent on page 368	To get the parent of the specified object.
Protected on page 373	To check whether the object is protected.
ReferenceName on page 378	To set or get the name of the referenced object.
ResultLevel on page 382	To set or get the result level of the object.
StateIconPath on page 394	To get the path to the symbol representing the state of the object.
Type on page 402	To set or get the type of the specified object.

Events

The EESConfigurationReader object definition contains the following event:

Event	Purpose
OnModified on page 485	To react to a property being modified.

Related topics

References

EESConfigurationReader (Data Object) (AutomationDesk Simulating Electrical Errors 📖)

EESConfigurationWriter

Syntax

No direct creation.

Purpose

To handle a specific EESConfigurationWriter data object.

Description

A EESConfigurationWriter object is a data object. It is used to provide a file writer object for saving the current error configuration to the specified error configuration file.

Properties

The EESConfigurationWriter object definition contains the following properties:

Property	Purpose
Author on page 290	To set or get the name of the person who created the object.
CreationDate on page 314	To get the date the object is created.
Description on page 317	To set or get the description of the object.
HasLibraryLink on page 327	To check whether the data object is linked to the custom library.
HierarchyName on page 328	To get the hierarchy path of the object.
IconPath on page 332	To get the path to the symbol representing the object type.
InOutState on page 338	To set or get the data direction of a data object.
IsLibraryElement on page 345	To check whether the object is a library object.
LibraryLink on page 349	To get the library link of the data object.
ModificationDate on page 358	To get the date of the last object modification.
Name on page 360	To set or get the name of the object.
Parent on page 368	To get the parent of the specified object.
Protected on page 373	To check whether the object is protected.
ReferenceName on page 378	To set or get the name of the referenced object.
ResultLevel on page 382	To set or get the result level of the object.
StateIconPath on page 394	To get the path to the symbol representing the state of the object.
Type on page 402	To set or get the type of the specified object.

Methods

None

Events

The EESConfigurationWriter object definition contains the following event:

Event	Purpose
OnModified on page 485	To react to a property being modified.

Related topics

References

EESConfigurationWriter (Data Object) (AutomationDesk Simulating Electrical Errors 📖)

EESPort

Syntax

No direct creation.

Purpose

To handle a specific EESPort data object.

Description

An EESPort object is a data object. It is the central point for downloading an error configuration and executing errors simulated on the HIL simulator.

An EESPort instance can only be used for one simulation application. If you want to load another simulation application, you must release the EESPort instance beforehand.

Properties

The EESPort object definition contains the following properties:

Property	Purpose
Author on page 290	To set or get the name of the person who created the object.
CreationDate on page 314	To get the date the object is created.
Description on page 317	To set or get the description of the object.
HasLibraryLink on page 327	To check whether the data object is linked to the custom library.
HierarchyName on page 328	To get the hierarchy path of the object.
IconPath on page 332	To get the path to the symbol representing the object type.
InOutState on page 338	To set or get the data direction of a data object.
IsLibraryElement on page 345	To check whether the object is a library object.
LibraryLink on page 349	To get the library link of the data object.
ModificationDate on page 358	To get the date of the last object modification.
Name on page 360	To set or get the name of the object.
Parent on page 368	To get the parent of the specified object.
Protected on page 373	To check whether the object is protected.
ReferenceName on page 378	To set or get the name of the referenced object.
ResultLevel on page 382	To set or get the result level of the object.
StateIconPath on page 394	To get the path to the symbol representing the state of the object.
Type on page 402	To set or get the type of the specified object.

Methods

None

Events

The EESPort object definition contains the following event:

Event	Purpose
OnModified on page 485	To react to a property being modified.

Related topics

References

EESPort (Data Object) (AutomationDesk Simulating Electrical Errors 📖)

EESPortFactory

Syntax

No direct creation.

Purpose

To handle an EESPortFactory data object.

Description

An EESPortFactory object is a data object. It is used to create and configure an EESPort object.

Properties

The EESPortFactory object definition contains the following properties:

Property	Purpose
Author on page 290	To set or get the name of the person who created the object.
CreationDate on page 314	To get the date the object is created.
Description on page 317	To set or get the description of the object.
HasLibraryLink on page 327	To check whether the data object is linked to the custom library.
HierarchyName on page 328	To get the hierarchy path of the object.
IconPath on page 332	To get the path to the symbol representing the object type.
InOutState on page 338	To set or get the data direction of a data object.
IsLibraryElement on page 345	To check whether the object is a library object.
LibraryLink on page 349	To get the library link of the data object.
ModificationDate on page 358	To get the date of the last object modification.
Name on page 360	To set or get the name of the object.
Parent on page 368	To get the parent of the specified object.
Protected on page 373	To check whether the object is protected.
ReferenceName on page 378	To set or get the name of the referenced object.

Property	Purpose
ResultLevel on page 382	To set or get the result level of the object.
StateIconPath on page 394	To get the path to the symbol representing the state of the object.
Type on page 402	To set or get the type of the specified object.

Methods

None

Events

The EESPortFactory object definition contains the following event:

Event	Purpose
OnModified on page 485	To react to a property being modified.

Related topics

References

EESPortFactory (Data Object) (AutomationDesk Simulating Electrical Errors 🕮)

ErrorConfiguration

Syntax

No direct creation.

Purpose

To handle an ErrorConfiguration data object.

Description

An ErrorConfiguration object is a data object. It is used to provide access to an error configuration. An error configuration consists of at least one error set. The properties and methods of the ErrorConfiguration data object lets you create error sets, save a created or modified error configuration and load an existing error configuration.

Properties

The ErrorConfiguration object definition contains the following properties:

Property	Purpose
Author on page 290	To set or get the name of the person who created the object.
CreationDate on page 314	To get the date the object is created.
Description on page 317	To set or get the description of the object.
HasLibraryLink on page 327	To check whether the data object is linked to the custom library.

Property	Purpose
HierarchyName on page 328	To get the hierarchy path of the object.
IconPath on page 332	To get the path to the symbol representing the object type.
InOutState on page 338	To set or get the data direction of a data object.
IsLibraryElement on page 345	To check whether the object is a library object.
LibraryLink on page 349	To get the library link of the data object.
ModificationDate on page 358	To get the date of the last object modification.
Name on page 360	To set or get the name of the object.
Parent on page 368	To get the parent of the specified object.
Protected on page 373	To check whether the object is protected.
ReferenceName on page 378	To set or get the name of the referenced object.
ResultLevel on page 382	To set or get the result level of the object.
StateIconPath on page 394	To get the path to the symbol representing the state of the object.
Type on page 402	To set or get the type of the specified object.

Methods

None

Events

The ErrorConfiguration object definition contains the following event:

Event	Purpose
OnModified on page 485	To react to a property being modified.

Related topics

References

ErrorConfiguration (Data Object) (AutomationDesk Simulating Electrical Errors 🕮)

ErrorFactory

Syntax

No direct creation.

Purpose

To handle an ErrorFactory data object.

Description

An ErrorFactory object is a data object. It is used to create and configure errors. The errors that you can simulate depends on your EES hardware. For information on the supported error types, refer to the user documentation of your hardware.

Properties

The ErrorFactory object definition contains the following properties:

Property	Purpose
Author on page 290	To set or get the name of the person who created the object.
CreationDate on page 314	To get the date the object is created.
Description on page 317	To set or get the description of the object.
HasLibraryLink on page 327	To check whether the data object is linked to the custom library.
HierarchyName on page 328	To get the hierarchy path of the object.
IconPath on page 332	To get the path to the symbol representing the object type.
InOutState on page 338	To set or get the data direction of a data object.
IsLibraryElement on page 345	To check whether the object is a library object.
LibraryLink on page 349	To get the library link of the data object.
ModificationDate on page 358	To get the date of the last object modification.
Name on page 360	To set or get the name of the object.
Parent on page 368	To get the parent of the specified object.
Protected on page 373	To check whether the object is protected.
ReferenceName on page 378	To set or get the name of the referenced object.
ResultLevel on page 382	To set or get the result level of the object.
StateIconPath on page 394	To get the path to the symbol representing the state of the object.
Type on page 402	To set or get the type of the specified object.

Methods

None

Events

The ErrorFactory object definition contains the following event:

Event	Purpose
OnModified on page 485	To react to a property being modified.

Related topics

References

ErrorFactory (Data Object) (AutomationDesk Simulating Electrical Errors 📖)

ErrorSet

Syntax

No direct creation.

Purpose

To handle an ErrorSet data object.

Description

An ErrorSet object is a data object. An error set contains all the errors that you want to execute subsequently when the error set is triggered. An error configuration consists of one or more error sets.

Properties

The ErrorSet object definition contains the following properties:

Property	Purpose
Author on page 290	To set or get the name of the person who created the object.
CreationDate on page 314	To get the date the object is created.
Description on page 317	To set or get the description of the object.
HasLibraryLink on page 327	To check whether the data object is linked to the custom library.
HierarchyName on page 328	To get the hierarchy path of the object.
IconPath on page 332	To get the path to the symbol representing the object type.
InOutState on page 338	To set or get the data direction of a data object.
IsLibraryElement on page 345	To check whether the object is a library object.
LibraryLink on page 349	To get the library link of the data object.
ModificationDate on page 358	To get the date of the last object modification.
Name on page 360	To set or get the name of the object.
Parent on page 368	To get the parent of the specified object.
Protected on page 373	To check whether the object is protected.
ReferenceName on page 378	To set or get the name of the referenced object.
ResultLevel on page 382	To set or get the result level of the object.
StateIconPath on page 394	To get the path to the symbol representing the state of the object.
Type on page 402	To set or get the type of the specified object.

Methods

None

Events

The ErrorSet object definition contains the following event:

Event	Purpose
OnModified on page 485	To react to a property being modified.

Related topics

References

ErrorSet (Data Object) (AutomationDesk Simulating Electrical Errors 📖)

MAPort

Syntax

No direct creation.

Purpose

To handle a specific MAPort data object.

Description

A MAPort object is a data object. It provides access to the simulation application.

Properties

The MAPort object definition contains the following properties:

Property	Purpose
Author on page 290	To set or get the name of the person who created the object.
CreationDate on page 314	To get the date the object is created.
Description on page 317	To set or get the description of the object.
HasLibraryLink on page 327	To check whether the data object is linked to the custom library.
HierarchyName on page 328	To get the hierarchy path of the object.
IconPath on page 332	To get the path to the symbol representing the object type.
InOutState on page 338	To set or get the data direction of a data object.
IsLibraryElement on page 345	To check whether the object is a library object.
LibraryLink on page 349	To get the library link of the data object.
ModificationDate on page 358	To get the date of the last object modification.
Name on page 360	To set or get the name of the object.
Parent on page 368	To get the parent of the specified object.
Protected on page 373	To check whether the object is protected.
ReferenceName on page 378	To set or get the name of the referenced object.
ResultLevel on page 382	To set or get the result level of the object.
StateIconPath on page 394	To get the path to the symbol representing the state of the object.
Type on page 402	To set or get the type of the specified object.

Methods

None

Events

The MAPort object definition contains the following event:

Event	Purpose
OnModified on page 485	To react to a property being modified.

Related topics

References

MAPort (Data Object) (AutomationDesk Accessing Simulation Platforms 📖)

MAPortConfiguration

Syntax

No direct creation.

Purpose

To handle a MAPortConfiguration data object.

Description

The MAPortConfiguration object is only the top level instance of a MAPortConfiguration data object in your AutomationDesk project. Like in AutomationDesk, it provides a root element that let you access the MAPortConfiguration's contents. For information on the root element's properties and methods, refer to the RootElement property.

A MAPortConfiguration contains a Dictionary object that consists of key-value pairs. A key can be of Int, Float or String data type.

Tip

A Dictionary data object and the MAPortConfiguration data object behave identical.

Properties

The MAPortConfiguration object definition contains the following properties:

Property	Purpose
Description on page 317	To set or get the description of the object.
HierarchyName on page 328	To get the hierarchy path of the object.
IconPath on page 332	To get the path to the symbol representing the object type.
IsLibraryElement on page 345	To check whether the object is a library object.
Name on page 360	To set or get the name of the object.

Property	Purpose
Parent on page 368	To get the parent object of the specified object.
Protected on page 373	To check whether the object is protected.
ReferenceName on page 378	To set or get the name of the referenced object.
ResultLevel on page 382	To set or get the result level of the specified object.
RootElement on page 385	To get access to the contents of the Dictionary object.
RootElement.Count on page 387	To get the number of items in a RootElement object.
RootElement.Keys on page 387	To get the keys available in a Dictionary or dictionary-based RootElement object.
RootElement.ParentObject on page 388	To get the parent of an item in a RootElement object.
RootElement.RootObject on page 388	To get the parent of a RootElement object.
RootElement.Type on page 389	To get the type of the contents of the RootElement object.
RootElement.Value on page 390	To get the contents of the RootElement object.
StateIconPath on page 394	To get the path to the symbol representing the state of the object.
Type on page 402	To get the type of the specified object.
ValueString on page 406	To set or get the values of the Dictionary object as a string.

Methods

The MAPortConfiguration object definition contains the following method:

Method	Purpose
CreateSubItem on page 428	To create a subitem that can be added to the root element.
ClearValue on page 418	To clear the values of the data object.
RootElement.Add on page 461	To add an item to a RootElement object.
RootElement.Clear on page 462	To clear the contents of the RootElement object.
RootElement.Contains on page 463	To check whether the specified key is available in the dictionary-based RootElement object.
RootElement.GetItem on page 464	To get an item of the RootElement object.
RootElement.Remove on page 466	To remove the specified item from the RootElement object.
RootElement.SetItem on page 468	To edit the value of an item of a RootElement object.

Events

The MAPortConfiguration object definition contains the following event:

Event	Purpose
OnModified on page 485	To react to a MAPortConfiguration property being modified.
OnValueChanged on page 496	To react to a MAConfiguration item beeing modified.

Related topics

References

MAPortConfiguration (Data Object) (AutomationDesk Accessing Simulation Platforms 📖)

MAPortFactory

Syntax

No direct creation.

Purpose

To handle an MAPortFactory data object.

Description

An MAPortFactory object is a data object. It is used to create and configure an MAPort object.

Properties

The MAPortFactory object definition contains the following properties:

Property	Purpose
Author on page 290	To set or get the name of the person who created the object.
CreationDate on page 314	To get the date the object is created.
Description on page 317	To set or get the description of the object.
HasLibraryLink on page 327	To check whether the data object is linked to the custom library.
HierarchyName on page 328	To get the hierarchy path of the object.
IconPath on page 332	To get the path to the symbol representing the object type.
InOutState on page 338	To set or get the data direction of a data object.
IsLibraryElement on page 345	To check whether the object is a library object.
LibraryLink on page 349	To get the library link of the data object.
ModificationDate on page 358	To get the date of the last object modification.
Name on page 360	To set or get the name of the object.
Parent on page 368	To get the parent of the specified object.
Protected on page 373	To check whether the object is protected.
ReferenceName on page 378	To set or get the name of the referenced object.
ResultLevel on page 382	To set or get the result level of the object.
StateIconPath on page 394	To get the path to the symbol representing the state of the object.
Type on page 402	To set or get the type of the specified object.

Methods

None

Events

The MAPortFactory object definition contains the following event:

Event	Purpose
OnModified on page 485	To react to a property being modified.

Related topics

References

MAPortFactory (Data Object) (AutomationDesk Accessing Simulation Platforms 📖)

Mapping (Object)

Syntax

No direct creation.

Purpose

To handle the XIL API Mapping data object.

Description

A Mapping object is a data object that contains a mapping of aliases to the model paths of variables.

Properties

The Mapping object definition contains the following properties:

Property	Purpose
Author on page 290	To set or get the name of the person who created the object.
CreationDate on page 314	To get the date the object is created.
Description on page 317	To set or get the description of the object.
HasLibraryLink on page 327	To check whether the data object is linked to the custom library.
HierarchyName on page 328	To get the hierarchy path of the object.
IconPath on page 332	To get the path to the symbol representing the object type.
InOutState on page 338	To set or get the data direction of a data object.
IsLibraryElement on page 345	To check whether the object is a library object.
LibraryLink on page 349	To get the library link of the data object.
ModificationDate on page 358	To get the date of the last object modification.
Name on page 360	To set or get the name of the object.
Parent on page 368	To get the parent of the specified object.

Property	Purpose
Protected on page 373	To check whether the object is protected.
ReferenceName on page 378	To set or get the name of the referenced object.
ResultLevel on page 382	To set or get the result level of the object.
StateIconPath on page 394	To get the path to the symbol representing the state of the object.
Type on page 402	To set or get the type of the specified object.

Methods

The Mapping object definition contains the following methods:

Method	Purpose
ClearValue on page 418	To clear the values of the data object.
Import on page 442	To import the variable mapping from an XML file.

Events

The Mapping object definition contains the following event:

Event	Purpose
OnModified on page 485	To react to a property being modified.

Related topics

References

Mapping (Data Object) (AutomationDesk Accessing Simulation Platforms 🕮)

PortConfig

Syntax

No direct creation.

Purpose

To handle an PortConfig data object.

Description

A PortConfig object is a data object. It is used to provide the base configuration for a formerly instantiated MAPort or EESPort object.

You must use this data object only, if you want to create and configure an MAPort or EESPort via a Testbench instance.

Properties

The PortConfig object definition contains the following properties:

Property	Purpose
Author on page 290	To set or get the name of the person who created the object.
CreationDate on page 314	To get the date the object is created.
Description on page 317	To set or get the description of the object.
HasLibraryLink on page 327	To check whether the data object is linked to the custom library.
HierarchyName on page 328	To get the hierarchy path of the object.
IconPath on page 332	To get the path to the symbol representing the object type.
InOutState on page 338	To set or get the data direction of a data object.
IsLibraryElement on page 345	To check whether the object is a library object.
LibraryLink on page 349	To get the library link of the data object.
ModificationDate on page 358	To get the date of the last object modification.
Name on page 360	To set or get the name of the object.
Parent on page 368	To get the parent of the specified object.
Protected on page 373	To check whether the object is protected.
ReferenceName on page 378	To set or get the name of the referenced object.
ResultLevel on page 382	To set or get the result level of the object.
StateIconPath on page 394	To get the path to the symbol representing the state of the object.
Type on page 402	To set or get the type of the specified object.

Methods

None

Events

The PortConfig object definition contains the following event:

Event	Purpose
OnModified on page 485	To react to a property being modified.

Related topics

References

PortConfig (Data Object) (AutomationDesk Accessing Simulation Platforms 📖)

SignalDescription

Syntax

No direct creation.

Purpose

To handle a specific SignalDescription data object.

Description

A SignalDescription object is a data object. It provides access to the description of one signal.

Properties

The SignalDescription object definition contains the following properties:

Property	Purpose
Author on page 290	To set or get the name of the person who created the object.
CreationDate on page 314	To get the date the object is created.
Description on page 317	To set or get the description of the object.
HasLibraryLink on page 327	To check whether the data object is linked to the custom library.
HierarchyName on page 328	To get the hierarchy path of the object.
IconPath on page 332	To get the path to the symbol representing the object type.
InOutState on page 338	To set or get the data direction of a data object.
IsLibraryElement on page 345	To check whether the object is a library object.
LibraryLink on page 349	To get the library link of the data object.
ModificationDate on page 358	To get the date of the last object modification.
Name on page 360	To set or get the name of the object.
Parent on page 368	To get the parent of the specified object.
Protected on page 373	To check whether the object is protected.
ReferenceName on page 378	To set or get the name of the referenced object.
ResultLevel on page 382	To set or get the result level of the object.
StateIconPath on page 394	To get the path to the symbol representing the state of the object.
Type on page 402	To set or get the type of the specified object.

Methods

None

Events

The SignalDecription object definition contains the following event:

Event	Purpose
OnModified on page 485	To react to a property being modified.

Related topics

References

SignalDescription (Data Object) (AutomationDesk Accessing Simulation Platforms 🕮)

SignalDescriptionSet

Syntax

No direct creation.

Purpose

To handle a specific SignalDescriptionSet data object.

Description

A SignalDescriptionSet object is a data object. It provides access to a container of the description of one or more signals.

Properties

The SignalDescriptionSet object definition contains the following properties:

Property	Purpose
Author on page 290	To set or get the name of the person who created the object.
CreationDate on page 314	To get the date the object is created.
Description on page 317	To set or get the description of the object.
HasLibraryLink on page 327	To check whether the data object is linked to the custom library.
HierarchyName on page 328	To get the hierarchy path of the object.
IconPath on page 332	To get the path to the symbol representing the object type.
InOutState on page 338	To set or get the data direction of a data object.
IsLibraryElement on page 345	To check whether the object is a library object.
LibraryLink on page 349	To get the library link of the data object.
ModificationDate on page 358	To get the date of the last object modification.
Name on page 360	To set or get the name of the object.
Parent on page 368	To get the parent of the specified object.
Protected on page 373	To check whether the object is protected.
ReferenceName on page 378	To set or get the name of the referenced object.
ResultLevel on page 382	To set or get the result level of the object.
StateIconPath on page 394	To get the path to the symbol representing the state of the object.
Type on page 402	To set or get the type of the specified object.

Methods

None

Events

The SignalDecriptionSet object definition contains the following event:

Event	Purpose
OnModified on page 485	To react to a property being modified.

Related topics

References

SignalDescriptionSet (Data Object) (AutomationDesk Accessing Simulation Platforms 🕮)

SignalDescriptionsReader

Syntax

No direct creation.

Purpose

To handle a specific SignalDescriptionsReader data object.

Description

A SignalDescriptionsReader object is a data object. It provides read access to a signal description.

Properties

The SignalDescriptionsReader object definition contains the following properties:

Property	Purpose
Author on page 290	To set or get the name of the person who created the object.
CreationDate on page 314	To get the date the object is created.
Description on page 317	To set or get the description of the object.
HasLibraryLink on page 327	To check whether the data object is linked to the custom library.
HierarchyName on page 328	To get the hierarchy path of the object.
IconPath on page 332	To get the path to the symbol representing the object type.
InOutState on page 338	To set or get the data direction of a data object.
IsLibraryElement on page 345	To check whether the object is a library object.
LibraryLink on page 349	To get the library link of the data object.
ModificationDate on page 358	To get the date of the last object modification.
Name on page 360	To set or get the name of the object.

Property	Purpose
Parent on page 368	To get the parent of the specified object.
Protected on page 373	To check whether the object is protected.
ReferenceName on page 378	To set or get the name of the referenced object.
ResultLevel on page 382	To set or get the result level of the object.
StateIconPath on page 394	To get the path to the symbol representing the state of the object.
Type on page 402	To set or get the type of the specified object.

Methods

None

Events

The SignalDecriptionReader object definition contains the following event:

Event	Purpose
OnModified on page 485	To react to a property being modified.

Related topics

References

SignalDescriptionSetReader (Data Object) (AutomationDesk Accessing Simulation Platforms 📖)

SignalDescriptionsWriter

Syntax

No direct creation.

Purpose

To handle a specific SignalDescriptionsWriter data object.

Description

A SignalDescriptionsWriter object is a data object. It provides write access to a signal description.

Properties

The SignalDescriptionsWriter object definition contains the following properties:

Property	Purpose
Author on page 290	To set or get the name of the person who created the object.
CreationDate on page 314	To get the date the object is created.
Description on page 317	To set or get the description of the object.

Property	Purpose
HasLibraryLink on page 327	To check whether the data object is linked to the custom library.
HierarchyName on page 328	To get the hierarchy path of the object.
IconPath on page 332	To get the path to the symbol representing the object type.
InOutState on page 338	To set or get the data direction of a data object.
IsLibraryElement on page 345	To check whether the object is a library object.
LibraryLink on page 349	To get the library link of the data object.
ModificationDate on page 358	To get the date of the last object modification.
Name on page 360	To set or get the name of the object.
Parent on page 368	To get the parent of the specified object.
Protected on page 373	To check whether the object is protected.
ReferenceName on page 378	To set or get the name of the referenced object.
ResultLevel on page 382	To set or get the result level of the object.
StateIconPath on page 394	To get the path to the symbol representing the state of the object.
Type on page 402	To set or get the type of the specified object.

Methods

None

Events

The SignalDecriptionWriter object definition contains the following event:

Event	Purpose
OnModified on page 485	To react to a property being modified.

Related topics

References

SignalDescriptionSetWriter (Data Object) (AutomationDesk Accessing Simulation Platforms 🕮)

SignalFactory

Syntax

No direct creation.

Purpose

To handle an SignalFactory data object.

Description

A SignalFactory object is a data object. It is used to instantiate a Signal object. You can build a signal with the same segment types as available in AutomationDesk's Signal Editor. In addition, you can create signal description sets and reader and writer for signal description sets.

Properties

The SignalFactory object definition contains the following properties:

Property	Purpose
Author on page 290	To set or get the name of the person who created the object.
CreationDate on page 314	To get the date the object is created.
Description on page 317	To set or get the description of the object.
HasLibraryLink on page 327	To check whether the data object is linked to the custom library.
HierarchyName on page 328	To get the hierarchy path of the object.
IconPath on page 332	To get the path to the symbol representing the object type.
InOutState on page 338	To set or get the data direction of a data object.
IsLibraryElement on page 345	To check whether the object is a library object.
LibraryLink on page 349	To get the library link of the data object.
ModificationDate on page 358	To get the date of the last object modification.
Name on page 360	To set or get the name of the object.
Parent on page 368	To get the parent of the specified object.
Protected on page 373	To check whether the object is protected.
ReferenceName on page 378	To set or get the name of the referenced object.
ResultLevel on page 382	To set or get the result level of the object.
StateIconPath on page 394	To get the path to the symbol representing the state of the object.
Type on page 402	To set or get the type of the specified object.

Methods

None

Events

The SignalFactory object definition contains the following event:

Event	Purpose
OnModified on page 485	To react to a property being modified.

Related topics

References

SignalFactory (Data Object) (AutomationDesk Accessing Simulation Platforms 📖)

SignalGenerator

Syntax

No direct creation.

Purpose

To handle a specific SignalGenerator data object.

Description

A SignalGenerator object is a data object. It provides the stimulation of model variables of a running simulation application.

Properties

The SignalGenerator object definition contains the following properties:

Property	Purpose
Author on page 290	To set or get the name of the person who created the object.
CreationDate on page 314	To get the date the object is created.
Description on page 317	To set or get the description of the object.
HasLibraryLink on page 327	To check whether the data object is linked to the custom library.
HierarchyName on page 328	To get the hierarchy path of the object.
IconPath on page 332	To get the path to the symbol representing the object type.
InOutState on page 338	To set or get the data direction of a data object.
IsLibraryElement on page 345	To check whether the object is a library object.
LibraryLink on page 349	To get the library link of the data object.
ModificationDate on page 358	To get the date of the last object modification.
Name on page 360	To set or get the name of the object.
Parent on page 368	To get the parent of the specified object.
Protected on page 373	To check whether the object is protected.
ReferenceName on page 378	To set or get the name of the referenced object.
ResultLevel on page 382	To set or get the result level of the object.
StateIconPath on page 394	To get the path to the symbol representing the state of the object.
Type on page 402	To set or get the type of the specified object.

Methods

None

Events

The SignalGenerator object definition contains the following event:

Event	Purpose
OnModified on page 485	To react to a property being modified.

Related topics	References
	SignalGenerator (Data Object) (AutomationDesk Accessing Simulation Platforms 📖)

SignalGeneratorFactory

Syntax

No direct creation.

Purpose

To handle a SignalGeneratorFactory data object.

Description

A SignalGeneratorFactory object is a data object. It is used to instantiate a SignalGeneratorReader or SignalGeneratorWriter object.

Properties

The SignalGeneratorFactory object definition contains the following properties:

Property	Purpose
Author on page 290	To set or get the name of the person who created the object.
CreationDate on page 314	To get the date the object is created.
Description on page 317	To set or get the description of the object.
HasLibraryLink on page 327	To check whether the data object is linked to the custom library.
HierarchyName on page 328	To get the hierarchy path of the object.
IconPath on page 332	To get the path to the symbol representing the object type.
InOutState on page 338	To set or get the data direction of a data object.
IsLibraryElement on page 345	To check whether the object is a library object.
LibraryLink on page 349	To get the library link of the data object.
ModificationDate on page 358	To get the date of the last object modification.
Name on page 360	To set or get the name of the object.
Parent on page 368	To get the parent of the specified object.
Protected on page 373	To check whether the object is protected.
ReferenceName on page 378	To set or get the name of the referenced object.
ResultLevel on page 382	To set or get the result level of the object.
StateIconPath on page 394	To get the path to the symbol representing the state of the object.
Type on page 402	To set or get the type of the specified object.

Methods

None

Events

The SignalGeneratorFactory object definition contains the following event:

Event	Purpose
OnModified on page 485	To react to a property being modified.

Related topics

References

SignalGeneratorFactory (Data Object) (AutomationDesk Accessing Simulation Platforms 🕮)

SignalGeneratorReader

Syntax

No direct creation.

Purpose

To handle a specific SignalGeneratorReader data object.

Description

A SignalGeneratorReader object is a data object. It configures a signal generator to read its stimulus signal from a file.

Properties

The SignalGeneratorReader object definition contains the following properties:

Property	Purpose
Author on page 290	To set or get the name of the person who created the object.
CreationDate on page 314	To get the date the object is created.
Description on page 317	To set or get the description of the object.
HasLibraryLink on page 327	To check whether the data object is linked to the custom library.
HierarchyName on page 328	To get the hierarchy path of the object.
IconPath on page 332	To get the path to the symbol representing the object type.
InOutState on page 338	To set or get the data direction of a data object.
IsLibraryElement on page 345	To check whether the object is a library object.
LibraryLink on page 349	To get the library link of the data object.
ModificationDate on page 358	To get the date of the last object modification.
Name on page 360	To set or get the name of the object.

Property	Purpose
Parent on page 368	To get the parent of the specified object.
Protected on page 373	To check whether the object is protected.
ReferenceName on page 378	To set or get the name of the referenced object.
ResultLevel on page 382	To set or get the result level of the object.
StateIconPath on page 394	To get the path to the symbol representing the state of the object.
Type on page 402	To set or get the type of the specified object.

Methods

None

Events

The SignalGeneratorReader object definition contains the following event:

Event	Purpose
OnModified on page 485	To react to a property being modified.

Related topics

References

SignalGeneratorReader (Data Object) (AutomationDesk Accessing Simulation Platforms 🕮)

SignalGeneratorWriter

Syntax

No direct creation.

Purpose

To handle a specific SignalGeneratorWriter data object.

Description

A SignalGeneratorWriter object is a data object. It configures a signal generator to write its stimulus signal to a file.

Properties

The SignalGeneratorWriter object definition contains the following properties:

Property	Purpose
Author on page 290	To set or get the name of the person who created the object.
CreationDate on page 314	To get the date the object is created.
Description on page 317	To set or get the description of the object.

Property	Purpose
HasLibraryLink on page 327	To check whether the data object is linked to the custom library.
HierarchyName on page 328	To get the hierarchy path of the object.
IconPath on page 332	To get the path to the symbol representing the object type.
InOutState on page 338	To set or get the data direction of a data object.
IsLibraryElement on page 345	To check whether the object is a library object.
LibraryLink on page 349	To get the library link of the data object.
ModificationDate on page 358	To get the date of the last object modification.
Name on page 360	To set or get the name of the object.
Parent on page 368	To get the parent of the specified object.
Protected on page 373	To check whether the object is protected.
ReferenceName on page 378	To set or get the name of the referenced object.
ResultLevel on page 382	To set or get the result level of the object.
StateIconPath on page 394	To get the path to the symbol representing the state of the object.
Type on page 402	To set or get the type of the specified object.

Methods

None

Events

The SignalGeneratorWriter object definition contains the following event:

Event	Purpose
OnModified on page 485	To react to a property being modified.

Related topics

References

SignalGeneratorWriter (Data Object) (AutomationDesk Accessing Simulation Platforms 🕮)

SignalGroupValue

Syntax

No direct creation.

Purpose

To handle a specific SignalGroupValue data object.

Description

A SignalGroupValue object is a data object. It accesses the captured data of a measurement including the times stamps.

Properties

The SignalGroupValue object definition contains the following properties:

Property	Purpose
Author on page 290	To set or get the name of the person who created the object.
CreationDate on page 314	To get the date the object is created.
Description on page 317	To set or get the description of the object.
HasLibraryLink on page 327	To check whether the data object is linked to the custom library.
HierarchyName on page 328	To get the hierarchy path of the object.
IconPath on page 332	To get the path to the symbol representing the object type.
InOutState on page 338	To set or get the data direction of a data object.
IsLibraryElement on page 345	To check whether the object is a library object.
LibraryLink on page 349	To get the library link of the data object.
ModificationDate on page 358	To get the date of the last object modification.
Name on page 360	To set or get the name of the object.
Parent on page 368	To get the parent of the specified object.
Protected on page 373	To check whether the object is protected.
ReferenceName on page 378	To set or get the name of the referenced object.
ResultLevel on page 382	To set or get the result level of the object.
StateIconPath on page 394	To get the path to the symbol representing the state of the object.
Type on page 402	To set or get the type of the specified object.

Methods

The SignalGroupValue object definition contains the following method:

Method	Purpose
ClearValue on page 418	To clear the values of the data object.

Events

The SignalGroupValue object definition contains the following event:

Event	Purpose
OnModified on page 485	To react to a property being modified.

Related topics

References

SignalGroupValue (Data Object) (AutomationDesk Accessing Simulation Platforms 📖)

SignalSegment

Syntax

No direct creation.

Purpose

To handle a specific SignalSegment data object.

Description

A SignalSegment object is a data object. It accesses a specific segment of a signal for reading or writing.

Properties

The SignalSegment object definition contains the following properties:

Property	Purpose
Author on page 290	To set or get the name of the person who created the object.
CreationDate on page 314	To get the date the object is created.
Description on page 317	To set or get the description of the object.
HasLibraryLink on page 327	To check whether the data object is linked to the custom library.
HierarchyName on page 328	To get the hierarchy path of the object.
IconPath on page 332	To get the path to the symbol representing the object type.
InOutState on page 338	To set or get the data direction of a data object.
IsLibraryElement on page 345	To check whether the object is a library object.
LibraryLink on page 349	To get the library link of the data object.
ModificationDate on page 358	To get the date of the last object modification.
Name on page 360	To set or get the name of the object.
Parent on page 368	To get the parent of the specified object.
Protected on page 373	To check whether the object is protected.
ReferenceName on page 378	To set or get the name of the referenced object.
ResultLevel on page 382	To set or get the result level of the object.
StateIconPath on page 394	To get the path to the symbol representing the state of the object.
Type on page 402	To set or get the type of the specified object.

Methods

None

Events

The SignalSegment object definition contains the following event:

Event	Purpose
OnModified on page 485	To react to a property being modified.

Related topics

References

SignalSegment (Data Object) (AutomationDesk Accessing Simulation Platforms 📖)

SignalValue

Syntax

No direct creation.

Purpose

To handle a specific SignalValue data object.

Description

A SignalValue object is a data object. It accesses a specific signal with its associated time stamp.

Properties

The SignalValue object definition contains the following properties:

Property	Purpose
Author on page 290	To set or get the name of the person who created the object.
CreationDate on page 314	To get the date the object is created.
Description on page 317	To set or get the description of the object.
HasLibraryLink on page 327	To check whether the data object is linked to the custom library.
HierarchyName on page 328	To get the hierarchy path of the object.
IconPath on page 332	To get the path to the symbol representing the object type.
InOutState on page 338	To set or get the data direction of a data object.
IsLibraryElement on page 345	To check whether the object is a library object.
LibraryLink on page 349	To get the library link of the data object.
ModificationDate on page 358	To get the date of the last object modification.
Name on page 360	To set or get the name of the object.
Parent on page 368	To get the parent of the specified object.

Property	Purpose
Protected on page 373	To check whether the object is protected.
ReferenceName on page 378	To set or get the name of the referenced object.
ResultLevel on page 382	To set or get the result level of the object.
StateIconPath on page 394	To get the path to the symbol representing the state of the object.
Type on page 402	To set or get the type of the specified object.

Methods

The SignalValue object definition contains the following method:

Method	Purpose
ClearValue on page 418	To clear the values of the data object.

Events

The SignalValue object definition contains the following event:

Event	Purpose
OnModified on page 485	To react to a property being modified.

Related topics

References

SignalValue (Data Object) (AutomationDesk Accessing Simulation Platforms 🕮)

SpecificErrorFactory

Syntax

No direct creation.

Purpose

To handle a SpecificErrorFactory data object.

Description

A SpecificErrorFactory object is a data object. It is used to create most of the available errors with additional attributes, such as dynamic errors.

The following errors must be created by using a different error factory object:
- InterchangedPins errors are created by using the BaseErrorBuilder data object.
- MultiPin2Pin errors are created by using the SpecificErrorFactory2 data object.

Properties

The SpecificErrorFactory object definition contains the following properties:

Property	Purpose
Author on page 290	To set or get the name of the person who created the object.
CreationDate on page 314	To get the date the object is created.
Description on page 317	To set or get the description of the object.
HasLibraryLink on page 327	To check whether the data object is linked to the custom library.
HierarchyName on page 328	To get the hierarchy path of the object.
IconPath on page 332	To get the path to the symbol representing the object type.
InOutState on page 338	To set or get the data direction of a data object.
IsLibraryElement on page 345	To check whether the object is a library object.
LibraryLink on page 349	To get the library link of the data object.
ModificationDate on page 358	To get the date of the last object modification.
Name on page 360	To set or get the name of the object.
Parent on page 368	To get the parent of the specified object.
Protected on page 373	To check whether the object is protected.
ReferenceName on page 378	To set or get the name of the referenced object.
ResultLevel on page 382	To set or get the result level of the object.
StateIconPath on page 394	To get the path to the symbol representing the state of the object.
Type on page 402	To set or get the type of the specified object.

Methods

None

Events

The SpecificErrorFactory object definition contains the following event:

Event	Purpose
OnModified on page 485	To react to a property being modified.

Related topics

References

SpecificErrorFactory (Data Object) (AutomationDesk Simulating Electrical Errors 📖)

SpecificError2Factory

Syntax

No direct creation.

Purpose

To handle a SpecificError2Factory data object.

Description

A SpecificError2Factory object is a data object. It is used to create MultiPin2Pin errors with additional attributes, such as dynamic errors.

Properties

The SpecificError2Factory object definition contains the following properties:

Property	Purpose
Author on page 290	To set or get the name of the person who created the object.
CreationDate on page 314	To get the date the object is created.
Description on page 317	To set or get the description of the object.
HasLibraryLink on page 327	To check whether the data object is linked to the custom library.
HierarchyName on page 328	To get the hierarchy path of the object.
IconPath on page 332	To get the path to the symbol representing the object type.
InOutState on page 338	To set or get the data direction of a data object.
IsLibraryElement on page 345	To check whether the object is a library object.
LibraryLink on page 349	To get the library link of the data object.
ModificationDate on page 358	To get the date of the last object modification.
Name on page 360	To set or get the name of the object.
Parent on page 368	To get the parent of the specified object.
Protected on page 373	To check whether the object is protected.
ReferenceName on page 378	To set or get the name of the referenced object.
ResultLevel on page 382	To set or get the result level of the object.
StateIconPath on page 394	To get the path to the symbol representing the state of the object.
Type on page 402	To set or get the type of the specified object.

Methods

None

Events

The SpecificErrorFactory2 object definition contains the following event:

Event	Purpose
OnModified on page 485	To react to a property being modified.

Related topics

References

SpecificErrorFactory2 (Data Object) (AutomationDesk Simulating Electrical Errors 📖)

Symbol

Syntax

No direct creation.

Purpose

To handle a specific Symbol data object.

Description

A Symbol object is a data object. It provides a placeholder for a string or a constant value.

Properties

The Symbol object definition contains the following properties:

Property	Purpose
Author on page 290	To set or get the name of the person who created the object.
CreationDate on page 314	To get the date the object is created.
Description on page 317	To set or get the description of the object.
HasLibraryLink on page 327	To check whether the data object is linked to the custom library.
HierarchyName on page 328	To get the hierarchy path of the object.
IconPath on page 332	To get the path to the symbol representing the object type.
InOutState on page 338	To set or get the data direction of a data object.
IsLibraryElement on page 345	To check whether the object is a library object.
LibraryLink on page 349	To get the library link of the data object.
ModificationDate on page 358	To get the date of the last object modification.
Name on page 360	To set or get the name of the object.
Parent on page 368	To get the parent of the specified object.
Protected on page 373	To check whether the object is protected.
ReferenceName on page 378	To set or get the name of the referenced object.
ResultLevel on page 382	To set or get the result level of the object.
StateIconPath on page 394	To get the path to the symbol representing the state of the object.
Type on page 402	To set or get the type of the specified object.

Methods

None

Events

The Symbol object definition contains the following event:

Event	Purpose
OnModified on page 485	To react to a property being modified.

Related topics

References

Symbol (Data Object) (AutomationDesk Accessing Simulation Platforms 🕮)

SymbolFactory

Syntax

No direct creation.

Purpose

To handle a SymbolFactory data object.

Description

A SymbolFactory object is a data object. It is used to instantiate a Symbol data object. It must only be used if you have created your testbench starting with the TestbenchFactory object according to the ASAM XIL API standard.

Properties

The SymbolFactory object definition contains the following properties:

Property	Purpose
Author on page 290	To set or get the name of the person who created the object.
CreationDate on page 314	To get the date the object is created.
Description on page 317	To set or get the description of the object.
HasLibraryLink on page 327	To check whether the data object is linked to the custom library.
HierarchyName on page 328	To get the hierarchy path of the object.
IconPath on page 332	To get the path to the symbol representing the object type.
InOutState on page 338	To set or get the data direction of a data object.
IsLibraryElement on page 345	To check whether the object is a library object.
LibraryLink on page 349	To get the library link of the data object.
ModificationDate on page 358	To get the date of the last object modification.
Name on page 360	To set or get the name of the object.

Property	Purpose
Parent on page 368	To get the parent of the specified object.
Protected on page 373	To check whether the object is protected.
ReferenceName on page 378	To set or get the name of the referenced object.
ResultLevel on page 382	To set or get the result level of the object.
StateIconPath on page 394	To get the path to the symbol representing the state of the object.
Type on page 402	To set or get the type of the specified object.

Methods

None

Events

The SymbolFactory object definition contains the following event:

Event	Purpose
OnModified on page 485	To react to a property being modified.

Related topics

References

SymbolFactory (Data Object) (AutomationDesk Accessing Simulation Platforms 📖)

TaskInfoFactory

Syntax

No direct creation.

Purpose

To handle a TaskInfoFactory data object.

Description

A TaskInfoFactory object is a data object. It is used to read the available information on a task that is assigned to a testbench's port, for example, a capture task on a model access port.

Properties

The TaskInfoFactory object definition contains the following properties:

Property	Purpose
Author on page 290	To set or get the name of the person who created the object.
CreationDate on page 314	To get the date the object is created.

Property	Purpose
Description on page 317	To set or get the description of the object.
HasLibraryLink on page 327	To check whether the data object is linked to the custom library.
HierarchyName on page 328	To get the hierarchy path of the object.
IconPath on page 332	To get the path to the symbol representing the object type.
InOutState on page 338	To set or get the data direction of a data object.
IsLibraryElement on page 345	To check whether the object is a library object.
LibraryLink on page 349	To get the library link of the data object.
ModificationDate on page 358	To get the date of the last object modification.
Name on page 360	To set or get the name of the object.
Parent on page 368	To get the parent of the specified object.
Protected on page 373	To check whether the object is protected.
ReferenceName on page 378	To set or get the name of the referenced object.
ResultLevel on page 382	To set or get the result level of the object.
StateIconPath on page 394	To get the path to the symbol representing the state of the object.
Type on page 402	To set or get the type of the specified object.

Methods

None

Events

The TaskInfo object definition contains the following event:

Event	Purpose
OnModified on page 485	To react to a property being modified.

Related topics

References

TaskInfo (Data Object) (AutomationDesk Accessing Simulation Platforms 🕮)

Testbench

Syntax

No direct creation.

Purpose

To handle a Testbench data object.

Description

A Testbench object is a data object. It is used as a wrapper for the Testbench class. It lets you read information given in the testbench configuration, such as the ASAM XIL API version, the name of the vendor-specific implementation and the available port types. If you do not use the default XIL API implementation by dSPACE, the vendor-specific information must be specified when you initialize the TestbenchFactory data object.

Properties

The Testbench object definition contains the following properties:

Property	Purpose
Author on page 290	To set or get the name of the person who created the object.
CreationDate on page 314	To get the date the object is created.
Description on page 317	To set or get the description of the object.
HasLibraryLink on page 327	To check whether the data object is linked to the custom library.
HierarchyName on page 328	To get the hierarchy path of the object.
IconPath on page 332	To get the path to the symbol representing the object type.
InOutState on page 338	To set or get the data direction of a data object.
IsLibraryElement on page 345	To check whether the object is a library object.
LibraryLink on page 349	To get the library link of the data object.
ModificationDate on page 358	To get the date of the last object modification.
Name on page 360	To set or get the name of the object.
Parent on page 368	To get the parent of the specified object.
Protected on page 373	To check whether the object is protected.
ReferenceName on page 378	To set or get the name of the referenced object.
ResultLevel on page 382	To set or get the result level of the object.
StateIconPath on page 394	To get the path to the symbol representing the state of the object.
Type on page 402	To set or get the type of the specified object.

Methods

None

Events

The Testbench object definition contains the following event:

Event	Purpose
OnModified on page 485	To react to a property being modified.

Related topics

References

Testbench (Data Object) (AutomationDesk Accessing Simulation Platforms 📖)

TestbenchFactory

Syntax

No direct creation.

Purpose

To handle a TestbenchFactory data object.

Description

A TestbenchFactory object is a data object. It is used to instantiate a testbench based on the vendor-specific port configuration.

Properties

The TestbenchFactory object definition contains the following properties:

Property	Purpose
Author on page 290	To set or get the name of the person who created the object.
CreationDate on page 314	To get the date the object is created.
Description on page 317	To set or get the description of the object.
HasLibraryLink on page 327	To check whether the data object is linked to the custom library.
HierarchyName on page 328	To get the hierarchy path of the object.
IconPath on page 332	To get the path to the symbol representing the object type.
InOutState on page 338	To set or get the data direction of a data object.
IsLibraryElement on page 345	To check whether the object is a library object.
LibraryLink on page 349	To get the library link of the data object.
ModificationDate on page 358	To get the date of the last object modification.
Name on page 360	To set or get the name of the object.
Parent on page 368	To get the parent of the specified object.
Protected on page 373	To check whether the object is protected.
ReferenceName on page 378	To set or get the name of the referenced object.
ResultLevel on page 382	To set or get the result level of the object.
StateIconPath on page 394	To get the path to the symbol representing the state of the object.
Type on page 402	To set or get the type of the specified object.

Methods

None

Events

The TestbenchFactory object definition contains the following event:

Event	Purpose
OnModified on page 485	To react to a property being modified.

Related topics

References

TestbenchFactory (Data Object) (AutomationDesk Accessing Simulation Platforms 📖)

ValueFactory

Syntax

No direct creation.

Purpose

To handle a ValueFactory data object.

Description

A ValueFactory object is a data object. It is used to instantiate and configure a Value object.

Properties

The ValueFactory object definition contains the following properties:

Property	Purpose
Author on page 290	To set or get the name of the person who created the object.
CreationDate on page 314	To get the date the object is created.
Description on page 317	To set or get the description of the object.
HasLibraryLink on page 327	To check whether the data object is linked to the custom library.
HierarchyName on page 328	To get the hierarchy path of the object.
IconPath on page 332	To get the path to the symbol representing the object type.
InOutState on page 338	To set or get the data direction of a data object.
IsLibraryElement on page 345	To check whether the object is a library object.
LibraryLink on page 349	To get the library link of the data object.
ModificationDate on page 358	To get the date of the last object modification.
Name on page 360	To set or get the name of the object.
Parent on page 368	To get the parent of the specified object.
Protected on page 373	To check whether the object is protected.
ReferenceName on page 378	To set or get the name of the referenced object.
ResultLevel on page 382	To set or get the result level of the object.
StateIconPath on page 394	To get the path to the symbol representing the state of the object.
Type on page 402	To set or get the type of the specified object.

Methods

None

Events

The ValueFactory object definition contains the following event:

Event	Purpose
OnModified on page 485	To react to a property being modified.

Related topics

References

ValueFactory (Data Object) (AutomationDesk Accessing Simulation Platforms 🕮)

ValueInfo

Syntax

No direct creation.

Purpose

To handle a ValueInfo data object.

Description

A ValueInfo object is a data object. It is used to read the available information that is assigned to a testbench's port, for example, an MAPortVariableInfo value

Properties

The ValueInfo object definition contains the following properties:

Property	Purpose
Author on page 290	To set or get the name of the person who created the object.
CreationDate on page 314	To get the date the object is created.
Description on page 317	To set or get the description of the object.
HasLibraryLink on page 327	To check whether the data object is linked to the custom library.
HierarchyName on page 328	To get the hierarchy path of the object.
IconPath on page 332	To get the path to the symbol representing the object type.
InOutState on page 338	To set or get the data direction of a data object.
IsLibraryElement on page 345	To check whether the object is a library object.
LibraryLink on page 349	To get the library link of the data object.
ModificationDate on page 358	To get the date of the last object modification.
Name on page 360	To set or get the name of the object.
Parent on page 368	To get the parent of the specified object.

Property	Purpose
Protected on page 373	To check whether the object is protected.
ReferenceName on page 378	To set or get the name of the referenced object.
ResultLevel on page 382	To set or get the result level of the object.
StateIconPath on page 394	To get the path to the symbol representing the state of the object.
Type on page 402	To set or get the type of the specified object.

Methods

None

Events

The VariableInfo object definition contains the following event:

Event	Purpose
OnModified on page 485	To react to a property being modified.

Related topics

References

VariableInfo (Data Object) (AutomationDesk Accessing Simulation Platforms 🕮)

Watcher

Syntax

No direct creation.

Purpose

To handle a specific Watcher data object.

Description

A Watcher object is a data object. It configures start, stop and duration conditions for capturing.

Properties

The Watcher object definition contains the following properties:

Property	Purpose
Author on page 290	To set or get the name of the person who created the object.
CreationDate on page 314	To get the date the object is created.
Description on page 317	To set or get the description of the object.
HasLibraryLink on page 327	To check whether the data object is linked to the custom library.

Property	Purpose
HierarchyName on page 328	To get the hierarchy path of the object.
IconPath on page 332	To get the path to the symbol representing the object type.
InOutState on page 338	To set or get the data direction of a data object.
IsLibraryElement on page 345	To check whether the object is a library object.
LibraryLink on page 349	To get the library link of the data object.
ModificationDate on page 358	To get the date of the last object modification.
Name on page 360	To set or get the name of the object.
Parent on page 368	To get the parent of the specified object.
Protected on page 373	To check whether the object is protected.
ReferenceName on page 378	To set or get the name of the referenced object.
ResultLevel on page 382	To set or get the result level of the object.
StateIconPath on page 394	To get the path to the symbol representing the state of the object.
Type on page 402	To set or get the type of the specified object.

Methods

None

Events

The Watcher object definition contains the following event:

Event	Purpose
OnModified on page 485	To react to a property being modified.

Related topics

References

Watcher (Data Object) (AutomationDesk Accessing Simulation Platforms 🕮)

WatcherFactory

Syntax

No direct creation.

Purpose

To handle a WatcherFactory data object.

Description

A WatcherFactory object is a data object. It is used to instantiate and configure a Watcher object.

Properties

The WatcherFactory object definition contains the following properties:

Property	Purpose
Author on page 290	To set or get the name of the person who created the object.
CreationDate on page 314	To get the date the object is created.
Description on page 317	To set or get the description of the object.
HasLibraryLink on page 327	To check whether the data object is linked to the custom library.
HierarchyName on page 328	To get the hierarchy path of the object.
IconPath on page 332	To get the path to the symbol representing the object type.
InOutState on page 338	To set or get the data direction of a data object.
IsLibraryElement on page 345	To check whether the object is a library object.
LibraryLink on page 349	To get the library link of the data object.
ModificationDate on page 358	To get the date of the last object modification.
Name on page 360	To set or get the name of the object.
Parent on page 368	To get the parent of the specified object.
Protected on page 373	To check whether the object is protected.
ReferenceName on page 378	To set or get the name of the referenced object.
ResultLevel on page 382	To set or get the result level of the object.
StateIconPath on page 394	To get the path to the symbol representing the state of the object.
Type on page 402	To set or get the type of the specified object.

Methods

None

Events

The WatcherFactory object definition contains the following event:

Event	Purpose
OnModified on page 485	To react to a property being modified.

Related topics

References

WatcherFactory (Data Object) (AutomationDesk Accessing Simulation Platforms 🕮)

Properties in Alphabetical Order

Introduction

The COM objects of the AutomationDesk COM API provide specific properties. The following list shows you all the available properties. In their descriptions you find the COM objects they are supported by.

Where to go from here

Information in this section

- A -

Where to go from here

Information in this section

AbsolutePath

Syntax

```
GetValue = Obj.AbsolutePath
or
Obj.AbsolutePath = SetValue
```

Purpose

To set or get the option whether to use the absolute or relative path for the specified file.

Description

With the AbsolutePath property you can choose how to handle the file path. A relative path is 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.

Property type

This property is using a Boolean value to set or get the value:

- 0: The specified file is used with its relative path.
- 1: The specified file is used with its absolute path.

Related objects

This property can be accessed by the following objects:

- File1 on page 168
- MATFile on page 198 (MAT file)
- MC3Collector on page 206 (result file, if storage type is set to eST_FILE)

ActiveProject

Syntax

```
GetValue = Obj.ActiveProject
or
Obj.ActiveProject = SetValue
```

Purpose	To set or get the active project.
Description	If you have opened several projects, the project you are working on must be specified as the active project.
Property type	This property uses a Project object (refer to Project on page 121) to set or get the active project.
Related objects	This property can be accessed by the following object: ▪ Projects (Object) on page 125 ▪ Projects1 on page 126 ▪ Projects2 on page 127 ▪ Projects3 on page 129

Attachment

Syntax

```
GetValue = Obj.Attachment
or
Obj.Attachment = SetValue
```

Purpose	To set or get the option whether to use the project's attachment folder or the file's path property to locate the file in the file system.
Description	This property specifies how the file location is specified: ▪ `True`: The file is located in the project's attachment folder, i.e., in `<ProjectName>\Attachments`. Only the file name is specified in the File2 object's Path property. ▪ `False`: The file is located in the path that is specified in the File2 object's Path property.
Property type	This property uses a Bool value to set or get the value.
Related objects	This property can be accessed by the following objects: ▪ File2 on page 170

Related topics

References

Author

Syntax

```
GetValue = Obj.Author
or
Obj.ActiveProject = SetAuthor
```

Purpose

To set or get the name of the person who created the object.

Description

By default, the Author property contains the log-on name of the person who created this object.

The Author property is one of the attributes that you can add to a report, refer to AvailableAttributes on page 291.

Property type

This property uses a string to set or get a value.

Related objects

This is a common property that can be accessed by any object *except for*:

- Any collection object
- Application on page 87
- ExecutionConfiguration on page 102
- Options (Object) on page 120
- Report on page 135
- ReportConfiguration on page 138
- Result on page 140
- ResultState (Object) on page 143
- StaticAttribute on page 149
- TAMVersion (Object) on page 151

Related topics

References

AvailableAttributes

Syntax	`GetValue = Obj.AvailableAttributes`
Purpose	To get the list of available attributes which you can add to the report.
Description	The AvailableAttributes property gives you access to a list of all available attributes that you can add to a report for describing the execution results of a sequence. To customize the list of attributes to be added to the report, refer to VisibleAttributes on page 410. The specified attributes are also added to project and folder items in the report, if you have specified to include project and folder information. You can also add some additional attributes to the report, refer to StaticAttribute on page 395. The layout of the report depends on the specified stylesheet. For further information, refer to StyleSheetPath on page 398.
Property type	This property returns a variant value.
Related objects	This property can be accessed by the following object: ▪ ReportConfiguration on page 138
Related topics	References StaticAttribute........395 StyleSheetPath........398 VisibleAttributes........410

AvailableBinaryFileNames

Syntax	`GetValue = Obj.AvailableBinaryFileNames`
Purpose	To get the names of the available binary files.
Description	Before you can get the names of the available binary files, the calibration system must be connected and the calibration project must be selected.

Property type	This property returns a variant value.
Related objects	This property can be accessed by the following object: ▪ MC3LogicalLink on page 203

AvailableBitsPerSecond

Syntax	`GetValue = Obj.AvailableBitsPerSecond`
Purpose	To get a list of the available values for the BitsPerSecond property.
Description	With this property, you get a list of values that AutomationDesk supports for the baud rate of the RS232 interface. Valid values are: 110, 300, 600, 1200, 2400, 4800, 9600, 14400, 19200, 38400, 56000, 57600, 115200, and 128000 bits per second.
Property type	This property returns a variant value.
Related objects	This property can be accessed by the following object: ▪ RS232Configuration on page 222

AvailableBufferRateNames

Syntax	`GetValue = Obj.AvailableBufferRateNames`
Purpose	To get the names of the available buffer rates.
Description	You can use a preconfigured buffer rate for the collector object.

Property type	This property returns a variant value.
Related objects	This property can be accessed by the following object: ▪ MC3Collector on page 206

AvailableCharacteristicTypeNames

Syntax	`GetValue = Obj.AvailableCharacteristicTypeNames`
Purpose	To get the available characteristic type names.
Description	The selected logical link provides different types of characteristics, for example, Scalar, Curve or Map.
Property type	This property returns a variant value.
Related objects	This property can be accessed by the following object: ▪ MC3Characteristics on page 204

AvailableControlPrimitiveNames

Syntax	`GetValue = Obj.AvailableControlPrimitiveNames`
Purpose	To get the names of the available ControlPrimitives.
Description	Before you can get the names of the available ControlPrimitives, the diagnostic system must be connected, the diagnostic project must be selected, and a VehicleInformation object must be specified.

Property type	This property returns a variant value.
Related objects	This property can be accessed by the following object: ▪ D3ControlPrimitive on page 215

AvailableDataBits

Syntax	`GetValue = Obj.AvailableDataBits`
Purpose	To get a list of the available values for the DataBits property.
Description	With this property, you get a list of values that AutomationDesk supports for the data bits of the RS232 interface. Valid values are: 5, 6, 7, 8
Property type	This property returns a variant value.
Related objects	This property can be accessed by the following object: ▪ RS232Configuration on page 222

AvailableFunctionalClassNames

Syntax	`GetValue = Obj.AvailableFunctionalClassNames`
Purpose	To get a list of the available functional class names.
Property type	This property returns a variant value.
Related objects	This property can be accessed by the following objects: ▪ D3Service on page 217

AvailableImplementations

Syntax	`GetValue = Obj.AvailableImplementations`
Purpose	To get the list of available XIL API implementations.
Description	The AvailableImplementations property provides a list of the installed XIL API implementations.
Property type	This property returns a list of string values.
Related objects	This property can be accessed by the following object: ▪ FrameworkConfiguration on page 109

AvailableInBufferSize

Syntax	`GetValue = Obj.AvailableInBufferSize`
Purpose	To get a list of the available values for the InBufferSize property.
Description	With this property, you get a list of values that AutomationDesk supports for the size of the input buffer of the RS232 interface. Valid values are: 1024, 2048, 5120 bits (selectable values in AutomationDesk) but also any even number.
Property type	This property returns a variant value.
Related objects	This property can be accessed by the following object: ▪ RS232Configuration on page 222

AvailableInterfaceNames

Syntax	`GetValue = Obj.AvailableInterfaceNames`
Purpose	To get the available interface names for connecting to a diagnostic or calibration system.
Description	The system that you want to connect to is specified by an interface name and the IP address of the host. With the AvailableInterfaceNames property you can get the names of all the diagnostic or calibration systems that are supported by AutomationDesk.
Property type	This property returns a variant value.
Related objects	This property can be accessed by the following objects: ▪ D3System on page 209 ▪ MC3System on page 200

AvailableLogicalLinkNames

Syntax	`GetValue = Obj.AvailableLogicalLinkNames`
Purpose	To get the names of the available LogicalLinks.
Description	Before you can get the names of the available logical links, the diagnostic system must be connected and the diagnostic project must be selected.
Property type	This property returns a variant value.
Related objects	This property can be accessed by the following objects: ▪ D3LogicalLink on page 214 ▪ MC3LogicalLink on page 203

AvailableModes

Syntax	`GetValue = Obj.AvailableModes`
Purpose	To get a list of the available values for the Mode property.
Description	With this property, you get a list of values that AutomationDesk supports for the file access mode of a MAT file. Valid values are: `"r"`, `"u"`, `"w"`, `"w4"`, `"wL"`, `"wz"`. For detailed information, refer to Mode on page 357.
Property type	This property returns a variant value.
Related objects	This property can be accessed by the following object: ▪ MATFile on page 198

AvailableOutBufferSize

Syntax	`GetValue = Obj.AvailableOutBufferSize`
Purpose	To get a list of the available values for the OutBufferSize property.
Description	With this property, you get a list of values that AutomationDesk supports for the size of the output buffer of the RS232 interface. Valid values are: 1024, 2048, 5120 bits (selectable values in AutomationDesk) but also any even number.
Property type	This property returns a variant value.
Related objects	This property can be accessed by the following object: ▪ RS232Configuration on page 222

AvailableParity

Syntax	`GetValue = Obj.AvailableParity`
Purpose	To get a list of the available values for the Parity property.
Description	With this property, you get a list of values that AutomationDesk supports for the parity scheme of the RS232 interface. Valid values are: "No", "Odd", "Even", "Mark", and "Space"
Property type	This property returns a variant value.
Related objects	This property can be accessed by the following object: ▪ RS232Configuration on page 222

AvailablePorts

Syntax	`GetValue = Obj.AvailablePorts`
Purpose	To get a list of the available values for the Ports property.
Description	With this property, you get a list of values that AutomationDesk supports for the port of the RS232 interface. Valid values are: COM1, COM2, COM3, COM4
Property type	This property returns a variant value.
Related objects	This property can be accessed by the following object: ▪ RS232Configuration on page 222

AvailableProjectNames

Syntax	`GetValue = Obj.AvailableProjectNames`
Purpose	To get the available project names from the connected diagnostic or calibration system.
Description	If AutomationDesk is connected to a diagnostic or calibration system, you can use this property to get the available projects specified on the diagnostic or calibration system.
Property type	This property returns a variant value.
Related objects	This property can be accessed by the following objects: ▪ D3Project on page 211 ▪ MC3Project on page 201

AvailableRepresentationTypeNames

Syntax	`GetValue = Obj.AvailableRepresentationTypeNames`
Purpose	To get the available representation type names.
Description	You can specify the conversion mode for each value you read or write by selecting a representation type, for example, eRT_ECU or eRT_PHYSICAL.
Property type	This property returns a variant value.
Related objects	This property can be accessed by the following objects: ▪ MC3Characteristics on page 204 ▪ MC3Collector on page 206

AvailableServiceNames

Syntax	`GetValue = Obj.AvailableServiceNames`
Purpose	To get a list of the available services from the selected Service object.
Property type	This property returns a variant value.
Related objects	This property can be accessed by the following objects: ▪ D3Service on page 217

AvailableSingleJobNames

Syntax	`GetValue = Obj.AvailableSingleJobNames`
Purpose	To get a list of the available single jobs from the selected D3SingleJob object.
Property type	This property returns a variant value.
Related objects	This property can be accessed by the following object: ▪ D3SingleJob on page 218

AvailableStopBits

Syntax	`GetValue = Obj.AvailableStopBits`
Purpose	To get a list of the available values for the StopBits property.
Description	With this property, you get a list of values that AutomationDesk supports for the number of stop bits to be used. Valid values are: 1, 1.5, 2

Property type	This property returns a variant value.
Related objects	This property can be accessed by the following object: ▪ RS232Configuration on page 222

AvailableStorageTypeNames

Syntax	`GetValue = Obj.AvailableStorageTypeNames`
Purpose	To get the names of the available storage types.
Description	You can use a preconfigured storage type for the collector object.
Property type	This property returns a variant value.
Related objects	This property can be accessed by the following object: ▪ MC3Collector on page 206

AvailableValueTypeNames

Syntax	`GetValue = Obj.AvailableValueTypeNames`
Purpose	To get the available value type names.
Description	The selected logical link provides different value types, for example, eVT_CONST, eVT_OFFSET_NEG, eVT_OFFSET_POS, eVT_VAL.

Property type	This property returns a variant value.
Related objects	This property can be accessed by the following object: ▪ MC3Characteristics on page 204

AvailableVehicleInformationNames

Syntax	`GetValue = Obj.AvailableVehicleInformationNames`
Purpose	To get the available VehicleInformation names from the selected diagnostic project.
Description	If AutomationDesk is connected to a diagnostic system, and you have selected a diagnostic project, you can use this property to get the available VehicleInformation entries.
Property type	This property returns a variant value.
Related objects	This property can be accessed by the following objects: ▪ D3VehicleInformation on page 212

- B -

Where to go from here

Information in this section

BitsPerSecond

Syntax

```
GetValue = Obj.BitsPerSecond
or
Obj.BitsPerSecond = SetValue
```

Purpose

To set or get the baud rate value in bits per second.

Description

With this property, you can set and get the baud rate of the RS232 interface.

Valid values are: 110, 300, 600, 1200, 2400, 4800, 9600, 14400, 19200, 38400, 56000, 57600, 115200, and 128000 bits per second.

If you do not specify the baud rate of the RS232 interface, the default baud rate of 9600 bits per second is used.

Property type

This property uses a long value to set or get a value.

Related objects

This property can be accessed by the following object:
- RS232Configuration on page 222

BinaryName

Syntax

```
GetValue = Obj.BinaryName
or
Obj.BinaryName = SetValue
```

Purpose

To set or get the name of the binary file.

Description	With this property, you can set the binary file that you want to use with the MC3LogicalLink object. You can choose a file from the list given by the **`AvailableBinaryFileNames`** property. You can also get the name of the binary file you have specified beforehand.
Property type	This property uses a string to set or get a value.
Related objects	This property can be accessed by the following object: ▪ MC3LogicalLink on page 203

Blue

Syntax

```
GetValue = Obj.Blue
or
Obj.Blue = SetValue
```

Purpose	To set or get the blue portion of an RGB color definition.
Description	This property gives you access to one specific value of the red, green or blue portions of an RGB color definition.
Property type	This property returns a long value.
Related objects	This property can be accessed by the following object: ▪ Color on page 221

BufferRate

Syntax

```
GetValue = Obj.BufferRate
or
Obj.BufferRate = SetValue
```

Purpose	To set or get the buffer rate of the MC3Collector object.

Description	With this property you can set or get the buffer rate for the values to be collected. For example, a buffer rate of 5 ms and a downsampling value of 10 results in writing measurement values after 50 ms.
Property type	This property uses a string to set or get a value.
Related objects	This property can be accessed by the following object: ▪ MC3Collector on page 206

BufferSize

Syntax

```
GetValue = Obj.BufferSize
or
Obj.BufferSize = SetValue
```

Purpose	To set or get the buffer size of the MC3Collector object.
Description	With this property you can set or get the number of sample times to be recorded in the measurement buffer of the Collector. As the buffer is a ring buffer, earlier values are overwritten by later values when the buffer capacity is exceeded. The buffer size must be greater than the number of samples to allow reading the sampled values before they are overwritten with the next measurement values.
Property type	This property uses a long value to set or get a value.
Related objects	This property can be accessed by the following object: ▪ MC3Collector on page 206

- C -

Where to go from here

Information in this section

CharacteristicName

Syntax

```
GetValue = Obj.CharacteristicName
or
Obj.CharacteristicName = SetValue
```

Purpose

To set or get the name of the characteristic object.

Description

This property represents the characteristic name as specified in the A2L file.

Property type

This property uses a string to set or get a value.

Related objects

This property can be accessed by the following object:

- MC3Characteristics on page 204

Characteristics

Syntax

```
GetValue = Obj.Characteristics
```

Purpose

To get the Characteristics data container of the MC3LogicalLink object.

Description

If you create a LogicalLink object, it automatically provides the *Characteristics* data container for all characteristic data objects and the *Collectors* data container for all collector data objects that you want to configure for your calibration task.

Property type

This property uses a LogicalLinkChildBase object to get the value.

Related objects

This property can be accessed by the following object:

- MC3LogicalLink on page 203

Related topics

References

Collectors..309

CharacteristicType

Syntax	`GetValue = Obj.CharacteristicType`
	`or`
	`Obj.CharacteristicType = SetValue`

Purpose	To set or get the type of the characteristic object.
Description	This property represents the type of the configured characteristic: ▪ Scalar ▪ Curve ▪ Map
Property type	This property uses a string to set or get a value.
Related objects	This property can be accessed by the following object: ▪ MC3Characteristics on page 204

ChildDataObjects

Syntax	`GetValue = Obj.ChildDataObjects`
Purpose	To get the ChildDataObjects collection of a data container object.
Description	If the current object provides one or more data container, for example, a DataContainer object or a LogicalLinkChildBase object, you can use this property to get the objects in this container.
Property type	This property returns a DataObjects (Object) object.
Related objects	This property can be accessed by the following object: ▪ DataContainer on page 160 ▪ LogicalLinkChildBase on page 119

Collectors

Syntax	`GetValue = Obj.Collectors`
Purpose	To get the Collectors data container of the MC3LogicalLink object.
Description	If you create a LogicalLink object, it automatically provides the *Characteristics* data container for all characteristic data objects and the *Collectors* data container for all collector data objects that you want to configure for your calibration task.
Property type	This property uses a LogicalLinkChildBase object to get the value.
Related objects	This property can be accessed by the following object: ▪ MC3LogicalLink on page 203
Related topics	References Characteristics........ 307

Condition (Property)

Syntax

```
GetValue = Obj.Condition
or
Obj.Condition = SetValue
```

Purpose

To set or get the expression of a Condition object.

Description

The Condition property gives you access to the expression of the Condition object. You can connect two expressions by an OR or an AND operator. For example, you can specify an expression such as `_AD_.Speed <= _AD_.Current AND _AD_.Temperature == _AD_.ABS_Variable1`.

Property type	This property uses a string to set or get a value.
Related objects	This property can be accessed by the following object: ▪ Condition (Object) on page 158

ConfigurationFile

Syntax

```
GetValue = Obj.ConfigurationFile
or
Obj.ConfigurationFile = SetValue
```

Purpose	To get or set the path of the XIL API framework configuration file.
Description	The ConfigurationFile property lets you specify which XIL API framework configuration file is used when the framework is initialized.
Property type	This property returns a string value.
Related objects	This property can be accessed by the following object: ▪ FrameworkConfiguration on page 109

ControlPrimitiveName

Syntax

```
GetValue = Obj.ControlPrimitiveName
or
Obj.ControlPrimitiveName = SetValue
```

Purpose	To set or get the name of the ControlPrimitive.
Description	With this property, you can set the ControlPrimitive that you want to use with the D3ControlPrimitive object. You can choose a ControlPrimitive from the list given by the AvailableControlPrimitiveNames property. You can also get the name of the ControlPrimitive you have specified beforehand.

Property type	This property uses a string to set or get a value.
Related objects	This property can be accessed by the following object: ▪ D3ControlPrimitive on page 215
Related topics	References AvailableControlPrimitiveNames........ 293

ControlPrimitives

Syntax

```
GetValue = Obj.Projects
```

Purpose	To get the ControlPrimitives collection object of the D3LogicalLink object.
Description	If you create a LogicalLink object, it automatically provides the *ControlPrimitives* data container for all ControlPrimitive data objects, the *Services* data container for all Service data objects, and the *SingleJobs* data container for all SingleJob data objects that you want to configure for your diagnostic task. The data containers represent the related collections for the COM API objects.
Property type	This property uses a LogicalLinkChildBase object to get the value.
Related objects	This property can be accessed by the following object: ▪ D3LogicalLink on page 214

ConvertToDouble

Syntax

```
GetValue = Obj.ConvertToDouble
or
Obj.ConvertToDouble = SetValue
```

Purpose	To set or get the conversion mode for integer values of the MATLAB or MATFile object.
Description	The MATLAB and MATFile data objects provide the ConvertToDouble property. Valid values are: 0 (False) and 1 (True) If you do not specify this property, the default value 0 is used (no data type conversion).
Property type	This property uses a Boolean value to set or get a value.
Related objects	This property can be accessed by the following objects: ▪ MATFile on page 198 ▪ MATLAB on page 196

Count

Syntax	`GetValue = Obj.Count`
Purpose	To get the number of instances of the object.
Description	The Count property returns the number of instantiated child elements of a collection.
Property type	This property returns a long value.
Related objects	This property can be accessed by the following objects: ▪ Any collection object ▪ RootElement object of a List, Tuple, and Dictionary object, and related data objects.

CreateReport

Syntax

```
GetValue = Obj.CreateReport
or
Obj.CreateReport = SetValue
```

Purpose

To set or get the option for creating a report directly after the execution.

Description

With the CreateReport property, you can decide whether a report should be generated directly after execution. The report can also be generated later with the GenerateReport method. The report content depends on the ReportConfiguration.

Property type

This property uses a Boolean value to set or get the option for creating a report directly after the execution.

Related objects

This property can be accessed by the following object:

- ExecutionConfiguration on page 102
- ExecutionConfiguration1 on page 103
- ExecutionConfiguration2 on page 104

Related topics

References

GenerateReport 438
ReportConfiguration 138
Reports (Object) 139

CreateResult

Syntax

```
GetValue = Obj.CreateResult
or
Obj.CreateResult = SetValue
```

Purpose

To set or get the option for logging the result of the execution.

Description	With the CreateResult property, you can decide whether the result of the execution should be logged. In some cases it may be not necessary to create an execution result, so you can deactivate the property to save time and disk storage. The Result object becomes the child element of the executed project, folder, or sequence, depending on where the execution started. The content of the result depends on the specified record depth and the result level of each executed object.
Property type	This property uses a Boolean value to set or get the option for logging the result of the execution.
Related objects	This property can be accessed by the following object: ▪ ExecutionConfiguration on page 102 ▪ ExecutionConfiguration1 on page 103 ▪ ExecutionConfiguration2 on page 104
Related topics	References RecordDepth.......... 377 Results (Object).......... 142

CreationDate

Syntax

```
GetValue = Obj.CreationDate
```

Purpose	To get the date the object was created.
Description	The CreationDate property shows you the date and time of the object's creation. It is generated automatically. The CreationDate property is one of the attributes that you can add to a report, refer to AvailableAttributes on page 291.
Property type	This property returns a date value.

Related objects

This is a common property that can be accessed by any object *except for*:

- Any collection object
- Application on page 87
- ExecutionConfiguration on page 102
- Options (Object) on page 120
- Report on page 135
- ReportConfiguration on page 138
- Result on page 140
- ResultState (Object) on page 143
- StaticAttribute on page 149
- TAMVersion (Object) on page 151

Related topics

References

- D -

Where to go from here

Information in this section

DataBits

Syntax

```
GetValue = Obj.DataBits
or
Obj.DataBits = SetValue
```

Purpose

To set or get the number of data bits used for the RS232 interface.

Description

With this property, you can set and get the number of data bits of the RS232 interface.

Valid values are: 5, 6, 7, 8

If you do not specify the number of data bits of the RS232 interface, the default value of 8 is used.

Note

Do not specify the following combinations of data bit and stop bit numbers:
- Number of data bits = 5 and number of stop bits = 2
- Number of data bits = 6, 7, or 8 and number of stop bits = 1.5

These combinations lead to invalid transmissions.

Property type

This property uses a long value to set or get a value.

Related objects

This property can be accessed by the following object:
- RS232Configuration on page 222

DataObjects (Property)

Syntax

```
GetValue = Obj.DataObjects
```

Purpose

To get the collection object for accessing a data object.

Description

The DataObjects property gives you access to the collection object, which manages the data objects of the current context (Project, Folder, Sequence, LibraryFolder, CustomLibraryFolder). With the collection object, you can create new data objects, or you can change the order of the existing data objects. You can modify the data objects, copy them, and also remove them from the current context.

Property type	This property returns a DataObjects (Object) object.
Related objects	This property can be accessed by the following objects: ▪ Project on page 121 ▪ Folder on page 105 ▪ Sequence on page 145 ▪ LibFolder on page 115 ▪ CustomLibraryFolder2 on page 96
Related topics	References DataObjects (Object)........ 101

Description

Syntax	`GetValue = Obj.Description`
	`or`
	`Obj.Description = SetValue`

Purpose	To set or get the description of the object.
Description	The Description property shows you the description of the object. The default description can be modified, unless the object is a library template.
Property type	This property uses a string value to set or get a description text.
Related objects	This is a common property that can be accessed by any object *except for*: ▪ Any collection object ▪ Application on page 87 ▪ ExecutionConfiguration on page 102 ▪ Options (Object) on page 120 ▪ Report on page 135 ▪ ReportConfiguration on page 138

- Result on page 140
- ResultState (Object) on page 143
- StaticAttribute on page 149
- TAMVersion (Object) on page 151

DisplayDataObjectValueUpdates

Syntax

```
GetValue = Obj.DisplayDataObjectValueUpdates
or
Obj.DisplayDataObjectValueUpdates = SetValue
```

Purpose

To set or get the option for updating data object values in the user interface during the execution.

Description

Via the DisplayDataObjectValueUpdates property, you can set or get the option that specifies whether the values of data objects that are displayed in the user interface are updated during the execution. Suspending the update of displayed values improves the performance.

By default, the DisplayDataObjectValueUpdates property is set to `False`.

Property type

This property uses a Boolean value to specify whether the values of the displayed data objects are updated.

Related objects

This property can be accessed by the following object:

- ExecutionConfiguration2 on page 104

Related topics

References

Options (Object).. 120

DownSampling

Syntax

```
GetValue = Obj.DownSampling
or
Obj.DownSampling = SetValue
```

Purpose	To set or get the downsampling rate of the MC3Collector object.
Description	With this property you can set or get the downsampling rate for the values to be collected. For example, a buffer rate of 5 ms and a downsampling value of 10 results in writing measurement values after 50 ms.
Property type	This property uses a long value to set or get a value.
Related objects	This property can be accessed by the following object: ▪ MC3Collector on page 206

- E -

Where to go from here

Information in this section

ErrorCount

Syntax

```
GetValue = Obj.ErrorCount
```

Purpose

To get the error state of an execution, started in a folder, sequence, or project.

Description

> **Note**
>
> The semantic of this property has changed with AutomationDesk 4.0. It is still provided for compatibility reasons. For a proper expression evaluating ResultState1, refer to Verdict (Property) on page 409.

The ErrorCount property shows you the state of an execution.When an error occurs, the execution is aborted because of an exception.

- 0: No error is occured
- 1: An error is occured

Property type

This property returns a long value.

Related objects

This property can be accessed by the following object:

- ResultState (Object) on page 143
- ResultState1 on page 144

Execution

Syntax

```
GetValue = Obj.Execution
```

Purpose

To get the execution configuration object.

Description

The Execution property gives you access to the ExecutionConfiguration object. You can use this object to specify the configuration of the execution, for example, whether to log the result or generate a report after execution, and how much information to put in the result.

Property type

This property returns an ExecutionConfiguration object.

Related objects

This property can be accessed by the following object:

- Options (Object) on page 120

Related topics	References ExecutionConfiguration........ 102

ExecutionDuration

Syntax	`GetValue = Obj.ExecutionDuration`
Purpose	To get the duration time of the execution.
Description	The ExecutionDuration shows you the time that the execution took from the start point to the end. The duration is calculated in seconds, for example, 0.479 s. The ExecutionDuration property is one of the attributes that you can add to the report, refer to AvailableAttributes on page 291.
Property type	This property returns a double value.
Related objects	This property can be accessed by the following object: ▪ ResultState (Object) on page 143 ▪ ResultState1 on page 144
Related topics	References AvailableAttributes........ 291

- F -

Where to go from here

Information in this section

FailedCount

Syntax

```
GetValue = Obj.FailedCount
```

Purpose

To get the failed state of an execution, started in a folder, sequence, or project.

Description

Note

The semantic of this property has changed with AutomationDesk 4.0. It is still provided for compatibility reasons. For a proper expression evaluating ResultState1, refer to Verdict (Property) on page 409.

The FailedCount property shows you the whether the execution led to an expected failure.

- 0: No failure occured.
- 1: A failure occured.

Property type	This property returns a long value.
Related objects	This property can be accessed by the following object: ▪ ResultState (Object) on page 143 ▪ ResultState1 on page 144

Favorites

Syntax

```
GetValue = Obj.Favorites
```

Purpose

To get the library favorites that are available for the library.

Description

The Favorites property gives you access to methods for exporting and importong the available library favorites.

Property type

This property uses a LibraryFavorites object.

Related object

This property can be accessed by the following object:
- Libraries2 on page 112
- Libraries3 on page 113

Related topics

References

Libraries (Object)..110
LibraryFavorites..114

FcnValues

Syntax

```
GetValue = Obj.FcnValues
```

Purpose

To get the Signal data object's vector of function values.

Description	With this property, you can get the Signal data object's vector of y-axis values.
Property type	This property returns a variant value.
Related objects	This property can be accessed by the following object: ▪ Signal on page 152

FileName

Syntax

```
GetValue = Obj.FileName
or
Obj.FileName = SetValue
```

Purpose	To set or get the path and name of a specific file.
Description	With this property, you can specify the path and name of the file you want to work with. You can configure the path as relative or absolute path by using the AbsolutePath property. The property can be used for: ▪ FailurePattern object (to specify the failure simulation system file) ▪ MATFile object (to specify the MAT file) ▪ MC3Collector object (you must specify the file where you want to store the collector results in if you have specified eST_FILE as storage type)
Property type	This property uses a string to set or get a value.
Related objects	This property can be accessed by the following objects: ▪ MATFile on page 198 ▪ MC3Collector on page 206
Related topics	References AbsolutePath..288

Framework (Property)

Syntax	`GetValue = Obj.Framework`
Purpose	To get the Framework object of the application or to get the FrameworkConfiguration object for the option.
Description	The Framework property is provided by different objects: **Application** The Framework property provides an object with methods to initialize and to shut down the framework. **Option** The Framework property provides an object with properties to configure the XIL API framework.
Property type	**Application** This property returns a Framework (Object) object. **Option** This property returns a FrameworkConfiguration object.
Related objects	This property can be accessed by the following objects: ▪ Application2 on page 89 ▪ Options (Object) on page 120

FunctionalClassName

Syntax

```
GetValue = Obj.FunctionalClassName
or
Obj.FunctionalClassName = SetValue
```

Purpose	To set or get a functional class.
Description	You can get a list of the available function classes by using the AvailableFunctionalClassNames property.
Property type	This property uses a string to set or get a value.

Related objects	This property can be accessed by the following objects: ▪ D3Service on page 217
Related topics	References AvailableFunctionalClassNames... 294

- G -

Green

Syntax

```
GetValue = Obj.Green
or
Obj.Green = SetValue
```

Purpose

To set or get the green portion of an RGB color definition.

Description

This property gives you access to one specific value of the red, green or blue portions of an RGB color definition.

Property type

This property returns a long value.

Related objects

This property can be accessed by the following object:

- Color on page 221

- H -

Where to go from here

Information in this section

HasLibraryLink

Syntax

```
GetValue = Obj.HasLibraryLink
```

Purpose

To look up whether the object is linked to the custom library.

Description

The HasLibraryLink property shows you whether there is a link between the object and the custom library. If a link exists, you can synchronize the object with the custom library.

Property type

This property returns a Boolean value.

Related objects

This is a common property that can be accessed by any object *except for*:

- Any collection object
- Application on page 87
- ExecutionConfiguration on page 102
- Folder on page 105
- Options (Object) on page 120
- Project on page 121
- Report on page 135
- ReportConfiguration on page 138
- Result on page 140

- ResultState (Object) on page 143
- StaticAttribute on page 149
- TAMVersion (Object) on page 151

Related topics

References

HierarchyName

Syntax

```
GetValue = Obj.HierarchyName
```

Purpose

To get the hierarchy path of the object.

Description

The HierarchyName property shows you the path of the object in the tree structure. It is an object-oriented description of the element's hierarchy, like *<Project>.<Folder>.<Sequence>*.

The HierarchyName property is one of the attributes that you can add to a report, refer to AvailableAttributes on page 291.

Property type

This property returns a string value.

Related objects

This is a common property that can be accessed by any object *except for*:

- Any collection object
- Application on page 87
- ExecutionConfiguration on page 102
- Options (Object) on page 120
- Report on page 135
- ReportConfiguration on page 138
- Result on page 140
- ResultState (Object) on page 143

- StaticAttribute on page 149
- TAMVersion (Object) on page 151

Related topics

References

AvailableAttributes.......... 291

Host

Syntax

```
GetValue = Obj.Host
or
Obj.Host = SetValue
```

Purpose

To set or get the host of the diagnostic or calibration system.

Description

The connection to the diagnostic or calibrationsystem is specified by an interface name and an IP address of the host. With the Host property you can get the IP address of the connected system, or set the IP address of the system you want to connect to.

Property type

This property uses a string to set or get a value.

Related objects

This property can be accessed by the following objects:
- D3System on page 209
- MC3System on page 200

Related topics

References

AvailableInterfaceNames.......... 296
Interface.......... 339

Hyperlink

Syntax

```
GetValue = Obj.Hyperlink
```

Purpose

To get the AutomationDesk hyperlink of the object.

Description

The provided hyperlink is a *Uniform Resource Identifier* (URI) as defined in the *RFC3986* standard.

The relevant elements of the syntax for AutomationDesk objects are:

```
<scheme>:<hier-part>[#<fragment>]
```

With the following meaning:

- *scheme*: `automationdesk`
- *hier-part*: specifies the project or the library
- *fragment*: specifies the object hierarchy within the project or library

Example The following URI specifies an Exec block in a project:

```
automationdesk:///C:/MyWorkingFolder/MyProject.adpx#MyFolder.MySequence.MySerial.MyExec
```

Property type

This property returns a string value.

Related objects

This method can be accessed by the following objects:

- Block on page 91
- CustomLibraryFolder on page 94
- CustomLibraryFolder1 on page 95
- CustomLibraryFolder2 on page 96
- DataObject on page 98
- DataObject2 on page 99
- Folder on page 105
- Folder1 on page 107
- LibFolder on page 115
- LibFolder1 on page 116
- Project on page 121
- Project1 on page 123
- PythonModule on page 130
- PythonPackage on page 132
- Sequence on page 145
- Sequence1 on page 147

- I -

Where to go from here

Information in this section

IconPath

Syntax	`GetValue = Obj.IconPath`
Purpose	To get the path to the symbol representing the object type.
Description	The IconPath property shows you the path to the symbol representing the object type. The AutomationDesk symbols are available in PNG format.
Property type	This property returns a string value.

Related objects

This is a common property that can be accessed by any object *except for*:
- Any collection object
- Application on page 87
- ExecutionConfiguration on page 102
- Options (Object) on page 120
- Report on page 135
- ReportConfiguration on page 138
- Result on page 140

- ResultState (Object) on page 143
- StaticAttribute on page 149
- TAMVersion (Object) on page 151

Ignore

Syntax

```
GetValue = Obj.Ignore
or
Obj.Ignore = SetValue
```

Purpose

To get or set whether the object and its child objects will be ignored for synchronization with SYNECT.

Description

This property lets you specify whether an object will be ignored for synchronization with SYNECT in the following way:

- When you set this property to `True`, the IsIgnored property of the object and of all of its child objects return to `True`, which specifies that they are ignored.
- When you set this property to `False`, the objects are not ignored for synchronization, except for the child objects that are specified to be ignored by their own Ignore property.

The default value is `False`.

Property type

This property uses a Boolean value.

Related object

This property can be accessed by the following object:

- Synect on page 150

Related topics

Basics

Basics on Using AutomationDesk with SYNECT (AutomationDesk Basic Practices 🕮)

References

Clear Ignore Flag (AutomationDesk Basic Practices 🕮)
IsIgnored..343
Set Ignore Flag (AutomationDesk Basic Practices 🕮)

Implementation

Syntax

```
GetValue = Obj.Implementation
or
Obj.Implementation = SetValue
```

Purpose

To get or set the XIL API implementation to be used.

Description

The Implementation property lets you specify which XIL API implementation is used for working with the framework.

Property type

This property returns a string value.

Related objects

This property can be accessed by the following object:

- FrameworkConfiguration on page 109

InBufferSize

Syntax

```
GetValue = Obj.InBufferSize
or
Obj.InBufferSize = SetValue
```

Purpose

To set or get the size of the input buffer of the RS232 interface.

Description

With this property, you can set and get the size of the input buffer of the RS232 interface.

Valid values are: 1024, 2048, 5120 bits (selectable values in AutomationDesk) but also any even number.

If you do not specify the buffer size of the RS232 interface, the default value of 5120 bits is used.

Property type	This property uses a long value to set or get a value.
Related objects	This property can be accessed by the following object: ▪ RS232Configuration on page 222

IncludeDescription

Syntax

```
GetValue = Obj.IncludeDescription
or
Obj.IncludeDescription = SetValue
```

Purpose

To set or get the option for adding all descriptions to the report.

Description

The IncludeDescription property is one of the attributes of the report configuration. You can use this property to define whether to add the description of a project or folder to the report. You can access the description of an element with the Description property.

Property type

This property uses a Boolean value to set or get the option for adding all descriptions to the report.

Related objects

This property can be accessed by the following object:

▪ StaticAttribute on page 149

Related topics

References

Description..317
ReportConfiguration...138

IncludeFolderAndProject

Syntax

```
GetValue = Obj.IncludeFolderAndProject
or
Obj.IncludeFolderAndProject = SetValue
```

Purpose	To set or get the option for adding folder and project information to the report.
Description	The IncludeFolderAndProject property is one of the attributes of the report configuration. You can use this property to define whether to add the folder and project information to the report. It also depends on the start point of the execution. Only the information for the specific project object and the specific folder objects is added to the report.
Property type	This property uses a Boolean value to set or get the option for adding folder and project information to the report.
Related objects	This property can be accessed by the following object: ▪ StaticAttribute on page 149
Related topics	References ReportConfiguration.. 138

IncludeReportBlocks

Syntax

```
GetValue = Obj.IncludeReportBlocks
or
Obj.IncludeReportBlocks = SetValue
```

Purpose	To set or get the option for adding the output of Report blocks to the report.
Description	The IncludeReportBlocks property is one of the attributes of the report configuration. You can use this property to define whether to add the output of the Report blocks to the report, independently of the specified result level. A Report block is an automation block that is contained in the Report library. For further information, refer to Generating Reports (AutomationDesk Basic Practices 🕮).
Property type	This property uses a Boolean value to set or get the option for adding the output of Report blocks to the report.

Related objects	This property can be accessed by the following object: ▪ StaticAttribute on page 149
Related topics	Basics Generating Reports (AutomationDesk Basic Practices 📖) References ReportConfiguration........ 138

IncludeResultState

Syntax

```
GetValue = Obj.IncludeResultState
or
Obj.IncludeResultState = SetValue
```

Purpose

To set or get the option for adding the result states to the report.

Description

The IncludeResultState property is one of the attributes of the report configuration. You can use this property to define whether to add the result state of an object to the report. If the sequences contain Decision blocks, the execution results can be qualified as passed, failed, and undefined.

Property type

This property uses a Boolean value to set or get the option for adding the result states to the report.

Related objects

This property can be accessed by the following object:
- StaticAttribute on page 149

Related topics

References

ReportConfiguration........ 138

InitializeOnStartUp

Syntax	`GetValue = Obj.InitializeOnStartUp`
	`or`
	`Obj.InitializeOnStartUp = SetValue`

Purpose	To get or set the initialization behavior of the XIL API Framework on the startup of AutomationDesk.
Description	If this property is set to `True`, the framework initializes automatically when you start AutomationDesk.
Property type	This property returns a Bool value.
Related objects	This property can be accessed by the following object: ▪ FrameworkConfiguration on page 109

InOutState

Syntax	`GetValue = Obj.InOutState`
	`or`
	`Obj.InOutState = SetValue`

Purpose	To set or get the data direction of a data object.
Description	Data objects, which you add to the project, sequence or automation block have no data direction by default. You can specify a data object as input data object, output data object or input/output data object.
Property type	This property uses a Int value to set or get the value.
Related objects	This property can be accessed by the following object: ▪ DataObject on page 98

Related topics

Basics

Overview of API Constants..24

Interface

Syntax

```
GetValue = Obj.Interface
or
Obj.Interface = SetValue
```

Purpose

To set or get the interface of the diagnostic or calibration system.

Description

The connection to the diagnostic or calibration system is specified by an interface name and an IP address of the host. With the Interface property you can get the interface name of the connected system, or set the interface name of the system you want to connect to. Before you set the interface name, you can use the AvailableInterfaceNames property to get a name of a diagnostic or calibration system that is supported by AutomationDesk.

Property type

This property uses a string to set or get a value.

Related objects

This property can be accessed by the following objects:

- D3System on page 209
- MC3System on page 200

Related topics

References

AvailableInterfaceNames..296
Host..329

IsAllAttributes

Syntax

```
GetValue = Obj.IsAllAttributes
or
Obj.IsAllAttributes = SetValue
```

Purpose

To set or get the option for adding all attributes or a customized set of attributes to the report.

Description

You can use the IsAllAttributes property to define whether to add all attributes to the report or only the customized set of attributes. To list all attributes, refer to AvailableAttributes on page 291. To specify the customized set of attributes, refer to VisibleAttributes on page 410.

Property type

This property uses a Boolean value to set or get the option for adding all attributes or a customized set of attributes to the report.

Related objects

This property can be accessed by the following object:

- ReportConfiguration on page 138

Related topics

References

IsCollapsed

Syntax

```
GetValue = Obj.IsCollapsed
or
Obj.IsCollapsed = SetValue
```

Purpose

To set or get the option for collapsing the object's structure in the project tree.

Description

With the IsCollapsed property, you can decide whether the object's structure should be collapsed in the project tree. When the AutomationDesk user interface is used, a collapsed project tree is loaded faster than a project tree that is not collapsed, especially if the project is very complex. When the Automation Server is used, the IsCollapsed property is not important.

Property type

This property uses a Boolean value to set or get the collapse state of the object in the project tree.

Related objects

This property can be accessed by the following objects:
- Block on page 91
- CustomLibraryFolder on page 94
- Folder on page 105
- LibFolder on page 115
- Project on page 121
- Sequence on page 145

IsConnected

Syntax

```
GetValue = Obj.IsConnected
```

Purpose

To get the status of the connection to the diagnostic or calibration system.

Description

After you have configured the connection to the diagnostic or calibration system, you must use the Connect method to connect to it. With the IsConnected property, you can check the status of the connection.

Property type

This property returns a Boolean value
- 0: AutomationDesk is not connected to the system.
- 1: AutomationDesk is connected to the system.

Related objects

This property can be accessed by the following objects:
- D3System on page 209
- MC3System on page 200

Related topics

References

Connect..423
Disconnect..431

IsCustomReport

Syntax

```
GetValue = Obj.IsCustomReport
or
Obj.IsCustomReport = SetValue
```

Purpose

To set or get the option for using a custom style sheet for the report generation.

Description

The output format of a report depends on the selected style sheet. You can use the IsCustomReport property to define whether to use your own style sheet or a predefined style sheet. If you want to use your own style sheet, you have to specify its path with the StyleSheetPath property. If you want to use one of the predefined style sheets, you have to specify this with the ReportType property.

Property type

This property uses a Boolean value to set or get the option for using a custom style sheet for report generation.

Related objects

This property can be accessed by the following object:

- ReportConfiguration on page 138

Related topics

References

ReportType..380
StyleSheetPath..398

IsEnabled

Syntax

```
GetValue = Obj.IsEnabled
or
Obj.IsEnabled = SetValue
```

Purpose

To set or get the enable state of an element.

Description

If an element is enabled, it is included in execution, if it is disabled, it is excluded from execution.

Property type	This property uses a Boolean value to set or get the enable state of an element.
Related objects	This property can be accessed by the following objects: ▪ Block on page 91 ▪ Folder on page 105 ▪ Sequence on page 145

IsExecutionRunning

Syntax

```
GetValue = Obj.IsExecutionRunning
```

Purpose

To get the status of the execution.

Description

With the IsExecutionRunning property, you can check the status of the execution.

Note

Immediately after a StopExecution call, cleanup activties might lead to a delayed switch of the IsExecutionRunning property.

Property type

This property returns a Boolean value
- 0: The object is not executed at the time.
- 1: The object is currently executed.

Related object

This property can be accessed by the following object:
- ExecutionConfiguration2 on page 104

IsIgnored

Syntax

```
GetValue = Obj.IsIgnored
```

Purpose

To get whether the object is ignored for synchronization with SYNECT.

Description	This property is used to lookup whether the object is ignored for synchronization with SYNECT: ▪ It returns `True` when the object or one of its parent objects are set to be ignored via the Ignore property. ▪ It returns `False` when the object and all of its parent objects are not set to be ignored. The default value is `False`.
Property type	This property returns a Boolean value.
Related object	This property can be accessed by the following object: ▪ Synect on page 150
Related topics	Basics Basics on Using AutomationDesk with SYNECT (AutomationDesk Basic Practices 🕮) References Clear Ignore Flag (AutomationDesk Basic Practices 🕮) Ignore........333 Set Ignore Flag (AutomationDesk Basic Practices 🕮)

IsInitialized

Syntax	`GetValue = Obj.IsInitialized`
Purpose	To get the initialization state of the XIL API Framework.
Description	This property is set to `True` if the XIL API framework is initialized. Otherwise it is set to `False`.

Property type	This property returns a Bool value.
Related objects	This property can be accessed by the following object: ▪ Framework (Object) on page 108

IsLibraryElement

Syntax	`GetValue = Obj.IsLibraryElement`
Purpose	To check whether the object is a library object.
Description	The IsLibraryElement property shows you whether this object is an element of the Standard Library or the custom library. If it is a library element, you can use it as a template to create a new object in a project.
Property type	This property returns a Boolean value to indicate whether the object is a library element.
Related objects	This is a common property that can be accessed by any object *except for*: ▪ Any collection object ▪ Application on page 87 ▪ ExecutionConfiguration on page 102 ▪ Options (Object) on page 120 ▪ Report on page 135 ▪ ReportConfiguration on page 138 ▪ Result on page 140 ▪ ResultState (Object) on page 143 ▪ StaticAttribute on page 149 ▪ TAMVersion (Object) on page 151

IsSelected

Syntax	`GetValue = Obj.IsSelected`

Purpose

To get the status of the project selection.

Description

After you have configured the diagnostic project, you must use the Select method to activate the project selection. With the **IsSelected** property, you can check the status of the project selection.

Property type

This property returns a Boolean value

- 0: Project is not selected.
- 1: Project is selected.

Related objects

This property can be accessed by the following objects:

- D3Project on page 211
- MC3Project on page 201

Related topics

References

- L -

Where to go from here

Information in this section

Label

Syntax

```
GetValue = Obj.Label
or
Obj.Label = SetValue
```

Purpose

To set or get the label of the current value.

Description

The Label property gives you access to the object's current label. You can only set it to a label that is defined in the value mapping dictionary of the LabeledValue object, otherwise an error occurs.

Property type

This property uses a string to set or get a value.

Related objects

This property can be accessed by the following object:

- LabeledValue on page 177

Related topics

References

LabelReferenceName

Syntax

```
GetValue = Obj.LabelReferenceName
or
Obj.LabelReferenceName = SetValue
```

Purpose

To set or get the name of a reference for the current label.

Description

The LabelReferenceName property gives you access the name of the String object that holds the current label of the LabeledValue object.

Property type

This property uses a string to set or get a value.

Related objects

This property can be accessed by the following object:

- LabeledValue on page 177

Related topics

References

Length

Syntax

```
GetValue = Obj.Length
```

Purpose

To get the length of the vectors that are contained in the Signal data object.

Description

With this property, you can get the number of x-axis values in the time vector of the Signal data object.

Property type

This property returns a long value.

Related objects

This property can be accessed by the following object:

- Signal on page 152

Libraries (Property)

Syntax

```
GetValue = Obj.Libraries
```

Purpose

To get the Libraries collection of the application.

Description

The Libraries property allows you to access the following libraries:

- Standard libraries containing templates for sequences and folders.
- The custom libraries contain templates of your custom sequences.
- Any built-in library providing templates for data objects, for example, the Main Library for accessing Int, Float, or String data objects.

Property type

This property returns a Libraries (Object) object.

Related objects

This property can be accessed by the following objects:

- Application on page 87
- Application1 on page 88
- Application2 on page 89

Related topics

References

Libraries (Object).. 110

LibraryLink

Syntax

```
GetValue = Obj.LibraryLink
```

Purpose

To get the library link of an object.

Description

The LibraryLink property gives you access to the path of the object's template in the custom library to which the instantiated object is linked to. The library link is required for synchronizing instantiated objects with their templates in the custom library. This entry is empty until you use custom objects from the custom library.

The LibraryLink property is one of the attributes that you can add to the report, refer to AvailableAttributes on page 291.

Property type

This property returns a Block object.

Related objects

This is a common property that can be accessed by any object *except for*:

- Any collection object
- Application on page 87
- ExecutionConfiguration on page 102
- Folder on page 105
- Options (Object) on page 120
- Project on page 121
- Report on page 135
- ReportConfiguration on page 138
- Result on page 140
- ResultState (Object) on page 143
- StaticAttribute on page 149
- TAMVersion (Object) on page 151

Related topics

References

AvailableAttributes.. 291
Block.. 91
Libraries (Object).. 110

Log (Property)

Syntax

```
GetValue = Obj.Log
```

Purpose

To get the Log object of the application.

Description

The Log property gives you access to the Log object that provides methods for writing simultaneously to the Message Viewer and to the dSPACE log file.

Property type

This property returns a Log object.

Related objects	This property can be accessed by the following object: ▪ Application2 on page 89
Related topics	References Log (Object) 118

LogicalLinkName

Syntax

```
GetValue = Obj.LogicalLinkName
or
Obj.LogicalLinkName = SetValue
```

Purpose

To set or get the name of the logical link.

Description

With this property, you can set the logical link that you want to use with the LogicalLink object. You can choose a logical link from the list given by the AvailableLogicalLinkNames property. You can also get the name of the logical link you have specified beforehand.

Property type

This property uses a string to set or get a value.

Related objects

This property can be accessed by the following objects:
- D3LogicalLink on page 214
- MC3LogicalLink on page 203

Related topics

References

AvailableLogicalLinkNames 296

LogicalLinks

Syntax

```
GetValue = Obj.LogicalLinks
```

Purpose	To get the LogicalLinks collection object.
Description	The LogicalLinks property is a collection based on the **DataObjects** collection. It provides the same methods, for example, Create, Remove and Copy, as the DataObjects collection.
Property type	This property uses a DataObjects (Object) object to get the value.
Related objects	This property can be accessed by the following objects: ▪ D3VehicleInformation on page 212 ▪ MC3Project on page 201
Related topics	References DataObjects (Object)........ 101

LogoAlignment

Syntax

```
GetValue = Obj.LogoAlignment
or
Obj.LogoAlignment = SetValue
```

Purpose

To set or get the alignment of the logo used in the report.

Description

The logo that you have specified to be displayed at the top of a generated report can be placed on the left, the right, or in the center of the page.

Property type

This property uses a value of the Alignment enumeration to set or get the alignment of the logo used in the report. If you want to use the predefined constants, you must make some preparations beforehand. For further information, refer to Using API Constants on page 67.

Constant	Value	Meaning
adLeft	0	Specifies left alignment for the logo.
adCenter	1	Specifies center alignment for the logo.

Constant	Value	Meaning
adRight	2	Specifies right alignment for the logo.

Related objects

This property can be accessed by the following object:
- ReportConfiguration on page 138

Related topics

References

LogoPath

Syntax

```
GetValue = Obj.LogoPath
or
Obj.LogoPath = SetValue
```

Purpose

To set or get the path to the logo used in the report.

Description

With the LogoPath property, you can use your own logo in the report. If the logo path is not set, a default logo will be used. The default logo is a dSPACE logo located
at .\dSPACE AutomationDesk <Version Number>\Main\DSPythonModules\AutomationDeskPackages\DSTAMReportGen\HTMLResources\ in your AutomationDesk installation.

Note

The maximum path length must not exceed 255 characters.

Property type

This property uses a string value to set or get the path to the logo used in the report.

Related objects

This property can be accessed by the following object:
- ReportConfiguration on page 138

Related topics

References

- M -

Where to go from here

Information in this section

Mapping (Property)

Syntax

```
GetValue = Obj.Mapping
```

Purpose

To get the root element of the value mapping dictionary.

Description

The Mapping property provides the root element that gives you access to the value mapping dictionary of the LabeledValue data object. This dictionary defines, which labels and values are valid for the LabeledValue data object. Refer to LabeledValue (AutomationDesk Basic Practices 📖) and Edit Value Mapping (AutomationDesk Basic Practices 📖).

Property type

This property provides a DictionaryValue object that serves as the root element of a dictionary.

Related objects

This property can be accessed by the following object:

- LabeledValue on page 177

Related topics

References

Label	347
RootElement	385
Value	404

Major

Syntax

```
GetValue = Obj.Major
```

Purpose

To get the AutomationDesk major release number.

Description

The TAMVersion object contains a major release number, a minor release number, and a revision, for example, "6.3.1", in which the Major property returns major release "6".

Property type

This property returns an int value.

Related objects

This property can be accessed by the following object:

- TAMVersion (Object) on page 151

Related topics

References

MeasurementName

Syntax

```
GetValue = Obj.MeasurementName
or
Obj.MeasurementName = SetValue
```

Purpose

To set or get the name of the measurement accessed by the MC3Measurement object.

Property type

This property uses a string to set or get a value.

Related objects

This property can be accessed by the following object:

- MC3Measurement on page 207

MeasurementVariables

Syntax

```
GetValue = Obj.MeasurementVariables
```

Purpose

To get the measurement variables of the MC3Collector object.

Property type

This property uses a DataObjects (Object) object to get the value.

Related objects

This property can be accessed by the following object:

- MC3Collector on page 206

Related topics	References DataObjects (Object) 101

Minor

Syntax

```
GetValue = Obj.Minor
```

Purpose

To get the AutomationDesk minor release number.

Description

The TAMVersion object contains a major release number, a minor release number, and a revision, for example, "6.3.1", in which the Minor property returns minor release "3".

Property type

This property returns an int value.

Related objects

This property can be accessed by the following object:

- TAMVersion (Object) on page 151

Related topics

References

Major	355
Revision	384

Mode

Syntax

```
GetValue = Obj.Mode
or
Obj.Mode = SetValue
```

Purpose

To set or get the file access mode of a MATFile object.

Description

With this property, you can set and get the file access mode of the instantiated MAT file.

Valid values are:

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.

If you do not specify the file access mode, the default mode `"r"` is used.

Property type

This property uses a string to set or get a value.

Related objects

This property can be accessed by the following object:

- MATFile on page 198

ModificationDate

Syntax

```
GetValue = Obj.ModificationDate
```

Purpose

To get the date of the last modification of the object.

Description	The ModificationDate property shows you the date and time of the object's modification. It is generated automatically. The modification date is one of the attributes that you can add to a report, refer to AvailableAttributes on page 291.
Property type	This property returns a date value.
Related objects	This is a common property that can be accessed by any object *except for*: ▪ Any collection object ▪ Application on page 87 ▪ ExecutionConfiguration on page 102 ▪ Options (Object) on page 120 ▪ Report on page 135 ▪ ReportConfiguration on page 138 ▪ Result on page 140 ▪ ResultState (Object) on page 143 ▪ StaticAttribute on page 149 ▪ TAMVersion (Object) on page 151
Related topics	References AvailableAttributes..291

Modified

Syntax	`GetValue = Obj.Modified`
Purpose	To look up whether the project object was modified.
Description	The Modified property is set automatically to `TRUE` after the project is modified. When you save the project the Modified property is reset.

Property type	This property returns a Boolean value.
Related objects	This property can be accessed by the following object: ▪ Project on page 121

- N -

Where to go from here

Information in this section

Name

Syntax

```
GetValue = Obj.Name
or
Obj.Name = SetValue
```

Purpose

To set or get the name of the object.

Description

The Name property gives you access to the object name. When you create an object, it gets a default name. If there is more than one object of the same type in the same project hierarchy, a consecutive number is added to the name. For example, if you add 3 sequences to the same project hierarchy, they are named "Sequence", "Sequence1", and "Sequence2". You can use the Name property to rename instantiated objects. For naming restrictions, refer to General Limitations (AutomationDesk Basic Practices 📖).

The Name property is one of the attributes that you can add to a report, refer to AvailableAttributes on page 291.

Property type	This property uses a string to set or get a value.
Related objects	This is a common property that can be accessed by any object *except for*: ▪ Any collection object ▪ Application on page 87 ▪ ExecutionConfiguration on page 102 ▪ Options (Object) on page 120 ▪ Report on page 135 ▪ ReportConfiguration on page 138 ▪ Result on page 140 ▪ ResultState (Object) on page 143 ▪ StaticAttribute on page 149 ▪ TAMVersion (Object) on page 151
Related topics	References AvailableAttributes..291

Names

Syntax	`GetValue = Obj.Names`
Purpose	To get the child element names of a collection.
Description	The Names property returns the names of the child elements of this collection. The names are listed in the order in which they appear in the project structure. This also gives you the position index of a child element. To access a child element, you can use the Item method with the child element's name or position as a parameter.
Property type	This property returns a variant value.
Related objects	This property can be accessed by the following objects: ▪ Any collection object

NumberOfSamples

Syntax	`GetValue = Obj.NumberOfSamples`
	`or`
	`Obj.NumberOfSamples = SetValue`

Purpose	To set or get the number of samples of the MC3Collector object.
Description	With this property you can set or get the number of samples to be collected. The measurement values of the collector are only returned if the specified number of samples has been reached. If you read the result before the collector is completed, you will get an empty dictionary. If the buffer of the calibration server contains as many samples as specified, a result event is sent.
Property type	This property uses a long value to set or get a value.
Related objects	This property can be accessed by the following object: ▪ MC3Collector on page 206

- O -

Where to go from here

Information in this section

OpenResultBrowser

Syntax

```
GetValue = Obj.OpenResultBrowser
or
Obj.OpenResultBrowser = SetValue
```

Purpose

To set or get the option for opening the Result Browser after execution.

Description

Via the OpenResultBrowser property, you can specify whether to open the Result Browser after the execution has finished.

Property type

This property uses a Boolean value to set or get the option for opening the Result Browser after execution.

Related objects

This property can be accessed by the following object:

- ExecutionConfiguration2 on page 104

Related topics

References

Operator

Syntax

```
GetValue = Obj.Operator
```

Purpose

To get the name of the person who started the execution.

Description

The Operator property shows you the name of the person who started the execution, as identified by the log-on.

Property type

This property returns a string value.

Related objects

This property can be accessed by the following object:
- ResultState (Object) on page 143
- ResultState1 on page 144

OperationMode

Syntax

```
GetValue = Obj.OperationMode
or
Obj.OperationMode = SetValue
```

Purpose

To set or get the operation mode of a built-in library.

Description

With this property, you can set and get the operation mode of a built-in library.

Property type

This property uses a value of the OperationMode enumeration to set or get the operation mode of a built-in library. If you want to use the predefined constants, you must make some preparations beforehand. For further information, refer to Using API Constants on page 67.

Constant	Value	Meaning
adOnline	0	Specifies the online operation mode for the built-in library. Required hardware and external devices are connected.
adOnlineRecording	1	Specifies the online recording operation mode for the built-in library. Required hardware and external devices are connected and their return values are recorded and saved to the offline data objects of the automation blocks used during execution.
adOffline	2	Specifies the offline operation mode for the built-in library. Required hardware and external devices are not connected. The previously parameterized offline data objects by recording or manual editing are used during execution.

Related objects

This property can be accessed by the following object:
- LibFolder1 on page 116

Options (Property)

Syntax

```
GetValue = Obj.Options
```

Purpose

To get the Options object of the application.

Description

The Options property gives you access to the ExecutionConfiguration and the ReportConfiguration objects for configuring the execution and report generation.

Property type

This property returns an Options object (refer to Options (Object) on page 120).

Related objects

This property can be accessed by the following objects:

- Application on page 87
- Application1 on page 88
- Application2 on page 89

Related topics

References

ExecutionConfiguration........ 102
ReportConfiguration........ 138

OutBufferSize

Syntax

```
GetValue = Obj.OutBufferSize
or
Obj.OutBufferSize = SetValue
```

Purpose

To set or get the size of the output buffer of the RS232 interface.

Description

With this property, you can set and get the size of the output buffer of the RS232 interface.

Valid values are: 1024, 2048, 5120 bits (selectable values in AutomationDesk) but also any even number.

If you do not specify the buffer size of the RS232 interface, the default value of 5120 bits is used.

Property type

This property uses a long value to set or get a value.

Related objects

This property can be accessed by the following object:

- RS232Configuration on page 222

- P -

Where to go from here

Information in this section

Parameters

Syntax

```
GetValue = Obj.Parameters
or
Obj.Parameters = SetValue
```

Purpose

To set or get the diagnostic parameters.

Description

If you have instantiated a ComPrimitive/ControlPrimitive, a Service, or a SingleJob, you can access its diagnostic parameters to edit their values. To get the available parameter names from the returned dictionary, you can use the RootElement.Keys property.

Property type

This property uses a Dictionary object to set or get the value.

Related objects

This property can be accessed by the following objects:

- D3ControlPrimitive on page 215
- D3Service on page 217
- D3SingleJob on page 218

Related topics

References

Dictionary........163
RootElement.Keys........387

Parent

Syntax	`GetValue = Obj.Parent`
Purpose	To get the parent object of the specified object.
Description	The Parent property gives you access to the object's parent object. The parent can be another object or the corresponding collection object. For an overview of possible parent objects, refer to Overview of API Object Dependencies on page 29.
Property type	This property returns an object of the parent's object type.
Related objects	This is a common property that can be accessed by any object *except for*: ▪ Application on page 87 ▪ ResultState (Object) on page 143 ▪ ResultState1 on page 144
Related topics	Basics Overview of API Object Dependencies..29

Parity

Syntax	`GetValue = Obj.Parity`
	`or`
	`Obj.Parity = SetValue`

Purpose	To set or get the parity scheme of the RS232 interface.
Description	With this property, you can set and get the parity scheme of the RS232 interface. Valid values are: `"No"`, `"Odd"`, `"Even"`, `"Mask"`, and `"Space"` If you do not specify the parity scheme of the RS232 interface, the default value `"No"` is used.

Property type	This property uses a string to set or get a value.
Related objects	This property can be accessed by the following object: ▪ RS232Configuration on page 222

PassedCount

Syntax

```
GetValue = Obj.PassedCount
```

Purpose

To get the passed state of an execution, started in a folder, sequence, or project.

Description

Note

The semantic of this property has changed with AutomationDesk 4.0. It is still provided for compatibility reasons. For a proper expression evaluating ResultState1, refer to Verdict (Property) on page 409.

The PassedCount property shows you whether the state is passed.

- 0: The state is not passed
- 1: The state is passed

Property type

This property returns a long value.

Related objects

This property can be accessed by the following object:

- ResultState (Object) on page 143
- ResultState1 on page 144

Path

Syntax

```
GetValue = Obj.Path
or
Obj.Path = SetValue
```

Purpose

To get or set the path of the specified object.

Description	The Path property shows you the path where the project, report, result, or file is stored. Only the path of a File object can be modified. The default path of a File object is " ".
Property type	This property returns a string value.
Related objects	This property can be accessed by the following objects: ▪ CustomLibraryFolder2 on page 96 ▪ File on page 167 ▪ File1 on page 168 ▪ Result1 on page 141 (only get method) ▪ Project on page 121 (only get method) ▪ PythonModule on page 130 ▪ PythonPackage on page 132 ▪ Report on page 135 (only get method)

PlatformManagement

Syntax	`GetValue = Obj.PlatformManagement`
Purpose	To get the dispatcher object for platform management of the application.
Description	The PlatformManagement property provides a dispatch object for platform management, for example, for loading a real-time application to a platform. For more information on the available methods and properties of the platform management, refer to **dSPACE Platform Management API Reference** 🕮.
Property type	This property returns an IPmPlatformManagement object. Refer to PlatformManagement / IPmPlatformManagement <<Interface>> (dSPACE Platform Management API Reference 🕮).
Related objects	This property can be accessed by the following object: ▪ Application2 on page 89

Port

Syntax	`GetValue = Obj.Port`
	`or`
	`Obj.Port = SetValue`

Purpose	To set or get the serial port of the PC used for the RS232 interface.
Description	With this property, you can set and get the serial port of the PC to be used for the RS232 interface. Valid values are: "COM1", "COM2", "COM3", and "COM4" If you do not specify the port of the RS232 interface, the default value "COM1" is used.
Property type	This property uses a string to set or get a value.
Related objects	This property can be accessed by the following object: ▪ RS232Configuration on page 222

ProjectName

Syntax	`GetValue = Obj.ProjectName`
	`or`
	`Obj.ProjectName = SetValue`

Purpose	To set or get the name of a diagnostic or calibration project.
Property type	This property uses a string to set or get a value.
Related objects	This property can be accessed by the following objects: ▪ D3Project on page 211 ▪ MC3Project on page 201

Projects (Property)

Syntax	`GetValue = Obj.Projects`
Purpose	To get the Projects collection of an object.
Description	The Projects collection is used with different objects: **Application** With the Projects collection you can create, open, and import AutomationDesk projects. You can also save and close all opened projects. **D3System, MC3System** With the Projects collection you can create, remove and copy COM objects for accessing diagnostic and calibration projects.
Property type	**Application** This property returns a Projects object (refer to Projects (Object) on page 125). **D3System, MC3System** This property returns a DataObjects object (refer to DataObjects (Object) on page 101).
Related objects	This property can be accessed by the following objects: ▪ Application on page 87 ▪ Application1 on page 88 ▪ Application2 on page 89 ▪ D3System on page 209 ▪ MC3System on page 200

ProjectTemplates

Syntax	`GetValue = Obj.ProjectTemplates`
Purpose	To get the available project templates.
Description	The ProjectTemplates property tells you which templates are available for creating a project. Project templates have a predefined structure that serves as a template for creating new project structures. At the moment, only the Standard Project template is available. This provides an unrestricted project structure.

Property type

This property returns a variant value.

Related objects

This property can be accessed by the following object:
- Projects (Object) on page 125
- Projects1 on page 126
- Projects2 on page 127
- Projects3 on page 129

Protected

Syntax

```
GetValue = Obj.Protected
```

Purpose

To check whether the object is protected.

Description

The Protected property shows you whether an object is protected. If an object is protected, you cannot modify its properties, for example, you cannot change its name or description. All library elements are protected.

Property type

This property returns a Boolean value.

Related objects

This is a common property that can be accessed by any object *except for*:
- Any collection object
- Application on page 87
- ExecutionConfiguration on page 102
- Options (Object) on page 120
- Report on page 135
- ReportConfiguration on page 138
- Result on page 140
- ResultState (Object) on page 143
- StaticAttribute on page 149
- TAMVersion (Object) on page 151

PythonModules (Property)

Syntax	`GetValue = Obj.PythonModules`
Purpose	To get the collection object for accessing the contained Python modules and packages.
Description	The PythonModules property gives you access to the collection object, which manages the Python modules and packages of the current context (CustomLibraryFolder, PythonPackage).
Property type	This property returns a PythonModules collection.
Related objects	This property can be accessed by the following objects: ▪ CustomLibraryFolder2 on page 96 ▪ PythonPackage on page 132
Related topics	References PythonModules (Property)......374

Selection (Property)

Syntax	`GetValue = Obj.Selection`
Purpose	To get the selected elements.
Description	The Selection property provides a collection of the elements that are currently selected in the Project Manger or the Library Bowser in the AutomationDesk user interface. This property is useful, for example, to create user defined functions that apply to selected elements. Refer to How to Add External Programs or Scripts as User Functions to AutomationDesk (AutomationDesk Basic Practices 📖).

Property type	This property returns a Selection (Object) object.
Related objects	This property can be accessed by the following object: ▪ Application2 on page 89

- R -

Where to go from here

Information in this section

ReadOnly

Syntax	`GetValue = Obj.ReadOnly`
Purpose	To look up whether the project is read-only.
Description	The ReadOnly property shows you if a project is read-only. If a project is read-only, it cannot be saved, even if it was modified. It is also impossible to execute the project.
Property type	This property returns a Boolean value.
Related objects	This property can be accessed by the following object: ▪ Project on page 121

RecordDepth

Syntax

```
GetValue = Obj.RecordDepth
or
Obj.RecordDepth = SetValue
```

Purpose

To set or get the record depth for the result.

Description

With the RecordDepth property, you can decide how detailed the logged result should be.

Property type

This property uses a value of the RecordDepth enumeration to set or get the record depth for the result:

Constant	Value	Meaning
adRecordHighAndMedium	0	Specifies that the results of all objects with a high or medium result level are logged.
adRecordHigh	1	Specifies that the results of all objects with a high result level are logged.
adRecordNone	2	Specifies that no result is logged.

Related objects

This property can be accessed by the following object:

- ExecutionConfiguration on page 102
- ExecutionConfiguration1 on page 103
- ExecutionConfiguration2 on page 104

Related topics

References

Results (Object)..142

Red

Syntax

```
GetValue = Obj.Red
or
Obj.Red = SetValue
```

Purpose

To set or get the red portion of an RGB color definition.

Description

This property gives you access to one specific value of the red, green or blue portions of an RGB color definition.

Property type

This property returns a long value.

Related objects

This property can be accessed by the following object:

- Color on page 221

ReferenceName

Syntax

```
GetValue = Obj.ReferenceName
or
Obj.ReferenceName = SetValue
```

Purpose

To set or get the name of the referenced object.

Description

The ReferenceName property gives you access to an object of the same type that you have specified in the AutomationDesk project. The values of your current object are set by the referenced object. For naming restrictions, refer to General Limitations (AutomationDesk Basic Practices 📖).

Property type

This property uses a string to set or get a value.

Related objects

This property can be accessed by the following objects:

- Each object representing an AutomationDesk data object, for example, Int, Tuple, or RS232Configuration.

Report

Syntax

```
GetValue = Obj.Report
```

Purpose	To get the report options.
Description	The Report property gives you access to the ReportConfiguration object. You can use this object to specify the configuration of the report, for example, which logo to use and how it should be aligned, and which report type and style sheet to use. **Note** The report configuration is saved in the registry, so that it is available for the report settings of AutomationDesk and the Automation Server.
Property type	This property returns a ReportConfiguration object.
Related objects	This property can be accessed by the following object: ▪ Options (Object) on page 120
Related topics	References ReportConfiguration........ 138

Reports (Property)

Syntax	`GetValue = Obj.Reports`
Purpose	To get the created reports of a result.
Description	The Reports property gives you access to the Reports collection object, which manages the reports of a result. With this collection object, you can generate new reports according to the result, or you can remove them.
Property type	This property returns a Reports (Object) object.

Related objects

This property can be accessed by the following object:

- Result on page 140

Related topics

References

ReportType

Syntax

```
GetValue = Obj.ReportType
or
Obj.ReportType = SetValue
```

Purpose

To set or get the output format of a report.

Description

With the ReportType property of the ReportConfiguration object you can specify or get the output format of the report. With the ReportType property of the Report object, you can only get the configured output format.

AutomationDesk and the Automation Server access the same settings. If you modify the settings via the Automation Server, the new settings are also valid for AutomationDesk sessions, and vice versa.

Note

The format of the **ReportType** property is determined by the **StyleSheetPath** property. If the format of the **ReportType** property does not match the format of the **StyleSheetPath** property, an error message is thrown.

Property type

This property uses a value of the ReportType enumeration to set or get the output format of a report. If you want to use the predefined constants, you must make some preparations beforehand. For further information, refer to Using API Constants on page 67.

Constant	Value	Meaning
adHTML	0	Specifies the output format of a report as HTML.
adPDF	1	Specifies the output format of a report as PDF.

Related objects

This property can be accessed by the following objects:
- ReportConfiguration on page 138
- Report on page 135 (read-only)

Related topics

References

StyleSheetPath 398

RepresentationType

Syntax

```
GetValue = Obj.RepresentationType
or
Obj.RepresentationType = SetValue
```

Purpose

To set or get the representation type of a calibration characteristic.

Description

You can specify the conversion mode for each value you read or write by selecting a representation type.

Possible values are:
- eRT_ECU
  Source value in its original format on the hardware.
- eRT_PHYSICAL
  Value in a converted form.

Property type

This property uses a string to set or get a value.

Related objects

This property can be accessed by the following objects:
- MC3Characteristics on page 204
- MC3Collector on page 206

ResultLevel

Syntax

```
GetValue = Obj.ResultLevel
or
Obj.ResultLevel = SetValue
```

Purpose

To set or get the result level of the specified object.

Description

The ResultLevel property gives you access to the result level of an object.You can specify the contents of the result and the report by combining the result level and the record depth (see RecordDepth on page 377). The default result level of a block element is *High*, and of a data object element it is *Medium*. The result level of a parent element controls the logging of its child elements. For example, if you specify the result level of a folder as *None*, its child elements are not logged in the result independently of their own result levels.

Property type

This property uses a value of the ResultLevel enumeration to set or get the result level of the specified object. If you want to use the predefined constants, you must make some preparations beforehand. For further information, refer to Using API Constants on page 67.

Constant	Value	Meaning
adResultHigh	0	Specifies that the result of this object is logged if you set the record depth to *High* (adRecordHigh) or *High and Medium* (adRecordHighAndMedium).
adResultMedium	1	Specifies that the result of this object is logged if you set the record depth to *High and Medium* (adRecordHighAndMedium).
adResultNone	2	Specifies that the result of this object is not logged independently of the record depth.

Related objects

This is a common property that can be accessed by any object *except for*:

- Any collection object
- Application on page 87
- ExecutionConfiguration on page 102
- Options (Object) on page 120
- Report on page 135
- ReportConfiguration on page 138
- Result on page 140
- ResultState (Object) on page 143

- StaticAttribute on page 149
- TAMVersion (Object) on page 151

Related topics

Basics

References

Results (Property)

Syntax

```
GetValue = Obj.Results
```

Purpose

To get the results of the specified object.

Description

The Results property gives you access to the collection object, which manages the results of a project, folder, or sequence. With the collection object, you can access a result or remove it from the project structure.

Property type

This property returns a Results (Object) object.

Related objects

This property can be accessed by the following objects:

- Project on page 121
- Folder on page 105
- Sequence on page 145

Related topics

References

ResultState (Property)

Syntax	`GetValue = Obj.ResultState`
Purpose	To get the result state of the specified object.
Description	The ResultState property gives you access to the ResultState object. The ResultState object contains some information about the execution, for example, the start and stop time. If the sequences contain Decision blocks, the execution results can be qualified as passed, failed, and undefined. You can get the occurrence of these decision results using the corresponding count properties. You can access the number of terminated sequences via the ErrorCount property.
Property type	This property returns a ResultState (Object) object.
Related objects	This property can be accessed by the following objects: ▪ Project on page 121 ▪ Folder on page 105 ▪ Sequence on page 145 ▪ Result on page 140
Related topics	References ResultState (Object)........ 143 StateIconPath........ 394

Revision

Syntax	`GetValue = Obj.Revision`
Purpose	To get the AutomationDesk revision number, i.e., the patch version.

Description

The TAMVersion object contains a major release number, a minor release number, and a revision number, for example, "6.3.1", in which the Revision property returns revision "1".

Property type

This property returns an int value.

Related objects

This property can be accessed by the following object:

- TAMVersion (Object) on page 151

Related topics

References

RootElement

Syntax

```
GetValue = Obj.RootElement
```

Purpose

To get the contents of a data object. The contents is based on a Dictionary, List or Tuple object.

Description

The Dictionary, List or Tuple object is only the top level instance of a data object in your AutomationDesk project. Like in AutomationDesk, it provides a root element that let you access the data object's contents. You must use the properties and methods of the data object's RootElement object for modifying its contents.

Properties provided by a RootElement object The RootElement's properties are available for Dictionary, Tuple, List and dictionary-based objects.

Property	Purpose
RootElement.Count on page 387	To get the number of items in the RootElement object.
RootElement.Keys on page 387	*Only Dictionary*: To get the keys available in the RootElement object.
RootElement.ParentObject on page 388	To get the parent of an item in the RootElement object.
RootElement.RootObject on page 388	To get the parent of the RootElement object.
RootElement.Type on page 389	To get the type of the contents of the RootElement object.
RootElement.Value on page 390	To get the contents of the RootElement object as key-value pairs.

Methods provided by a RootElement object The RootElement's methods differ for its root object type. The following methods are available for Dictionary objects.

Method (Dictionary)	Purpose
RootElement.Add on page 461	To add an item to the RootElement object.
RootElement.Clear on page 462	To clear the contents of the RootElement object.
RootElement.Contains on page 463	To check whether the specified key is available in the RootElement object.
RootElement.GetItem on page 464	To get an item of the RootElement object.
RootElement.Remove on page 466	To remove an item from the RootElement object.
RootElement.SetItem on page 468	To edit the value of an item of the RootElement object.

The following methods are available for List objects.

Method (List)	Purpose
RootElement.Add on page 461	To add an item to the RootElement object.
RootElement.Clear on page 462	To clear the contents of the RootElement object.
RootElement.GetItem on page 464	To get an item of the RootElement object.
RootElement.IndexOf on page 465	To get the first index of the specified value in a RootElement object.
RootElement.Insert on page 466	To add an item in the RootElement object at a specific position.
RootElement.Remove on page 466	To remove an item from the RootElement object.
RootElement.RemoveAt on page 467	To remove an item from the given position in the RootElement object.
RootElement.SetItem on page 468	To edit the value of an item of the RootElement object.

The following methods are available for Tuple objects.

Method (Tuple)	Purpose
RootElement.GetItem on page 464	To get an item of the RootElement object.
RootElement.IndexOf on page 465	To get the first index of the specified value in a RootElement object.

Property type

This property returns a specific value object depending on the type of the current object. For example, it returns a DictionaryValue object, if the current object is a Dictionary or a dictionary-based object.

Related objects

This property can be accessed by the following objects:

- Dictionary on page 163
- List on page 180
- Tuple on page 187
- MAPortConfiguration on page 251
- Mapping (Property) on page 354

RootElement.Count

Syntax	`GetValue = Obj.Count`
Purpose	To get the number of items in a RootElement object.
Description	This property is available via the RootElement object of the related object. You get the RootElement object by using the RootElement property.
Property type	This property returns a long value.
Related objects	This property can be accessed by the following objects: ▪ Dictionary on page 163 ▪ List on page 180 ▪ Tuple on page 187
Related topics	References RootElement.. 385

RootElement.Keys

Syntax	`GetValue = Obj.Keys`
Purpose	To get the keys available in a Dictionary or dictionary-based RootElement object.
Description	This property is available via the RootElement object of the related object. You get the RootElement object by using the RootElement property.
Property type	This property returns a variant value.
Related objects	This property can be accessed by the following object: ▪ Dictionary on page 163

Related topics

References

RootElement.. 385

RootElement.ParentObject

Syntax

```
GetValue = Obj.ParentObject
```

Purpose

To get the parent of an item in a RootElement object.

Description

This property is available via the RootElement object of the related object. You get the RootElement object by using the RootElement property.

Property type

This property returns NULL (<COMObject <unknown>>), because the root element is on top level of the related object.

Related objects

This property can be accessed by the following objects:

- Dictionary on page 163
- List on page 180
- Tuple on page 187

Related topics

References

RootElement.. 385

RootElement.RootObject

Syntax

```
GetValue = Obj.RootObject
```

Purpose

To get the parent of a RootElement object.

Description

This property is available via the RootElement object of the related object. You get the RootElement object by using the RootElement property.

Property type

This property returns an object from which the RootElement has been created.

Related objects

This property can be accessed by the following objects:
- Dictionary on page 163
- List on page 180
- Tuple on page 187

Related topics

References

RootElement.. 385

RootElement.Type

Syntax

```
GetValue = Obj.Type
```

Purpose

To get the type of the contents of the RootElement object.

Description

This property is available via the RootElement object of the related object. You get the RootElement object by using the RootElement property.

Property type

This property returns one of the predefined long values:
- 0 - adMainLibraryTupleValue
- 1 - adMainLibraryListValue
- 2 - adMainLibraryDictionaryValue

Related objects

This property can be accessed by the following objects:
- Dictionary on page 163
- List on page 180
- Tuple on page 187

Related topics

References

RootElement.Value

Syntax

```
GetValue = Obj.Value
```

Purpose

To get the contents of the RootElement object.

Description

This property is available via the RootElement object of the related object. You get the RootElement object by using the RootElement property.

Property type

The property type depends on the object, for which you use the property.

Object	Property Type
Dictionary	Key-value pairs to set or get the contents of the dictionary.
List	Variant data object to set or get the contents of the list.
Tuple	Variant data object to set or get the contents of the tuple.

Related objects

This property can be accessed by the following objects:
- Dictionary on page 163
- List on page 180
- Tuple on page 187

Related topics

References

- S -

Where to go from here

Information in this section

ServiceName

Syntax

```
GetValue = Obj.ServiceName
or
Obj.ServiceName = SetValue
```

Purpose

To set or get the name of a service.

Property type

This property uses a string to set or get a value.

Related objects

This property can be accessed by the following objects:

- D3Service on page 217

Services

Syntax

```
GetValue = Obj.Services
```

Purpose

To get the Services data container (collection object) of a LogicalLink object.

Description

If you create a LogicalLink object, it automatically provides the *ControlPrimitives* data container for all ControlPrimitive data objects, the *Services* data container for all Service data objects, and the *SingleJobs* data container for all SingleJob data objects, that you want to configure for your diagnostic task. The data containers represent the related collections for the COM API objects.

Property type

This property uses a LogicalLinkChildBase object to get the value.

Related objects

This property can be accessed by the following objects:

- D3LogicalLink on page 214

SingleJobName

Syntax

```
GetValue = Obj.SingleJobName
or
Obj.SingleJobName = SetValue
```

Purpose

To set or get the name of a single job.

Property type	This property uses a string to set or get a value.
Related objects	This property can be accessed by the following object: ▪ D3SingleJob on page 218

SingleJobs

Syntax	`GetValue = Obj.SingleJobs`
Purpose	To get the SingleJobs collection object of the D3LogicalLink object.
Description	The SingleJobs property is a collection based on the **DataObjects** collection. It provides the same methods, for example, Create, Remove and Copy, as the DataObjects collection.
Property type	This property uses a LogicalLinkChildBase object to get the value.
Related objects	This property can be accessed by the following object: ▪ D3LogicalLink on page 214

StartTime

Syntax	`GetValue = Obj.StartTime`
Purpose	To get the start time of an execution.
Description	The StartTime property shows you the time when the whole execution started. The start time of the execution has the YYYY/MM/DD, HH-MM-SS format by default, for example, 2011/10/06, 16-29-10.

You can change the format by using time format methods of your programming language. For example, when using Python, you can use the following methods:

```
StartTime = ResultState.StartTime
print (time.ctime(int(StartTime)))
# leads to 'Thu Oct 06 16:29:10 2011'
# Another format
print (time.strftime("%#c", time.localtime(int(StartTime))))
# leads to 'Thursday, October 06, 2011 16:29:10'
```

The StartTime property is one of the attributes that you can add to the report, refer to AvailableAttributes on page 291.

Property type

This property returns a date value.

Related objects

This property can be accessed by the following object:

- ResultState (Object) on page 143
- ResultState1 on page 144

Related topics

References

StateIconPath

Syntax

```
GetValue = Obj.StateIconPath
```

Purpose

To get the path to the symbol representing the state of the object.

Description

If you use Decision blocks in your AutomationDesk sequence, the execution states *passed*, *failed*, and *undefined* are indicated by symbols that are part of a block's graphical representation. The execution states of an automation block are transferred to all parent elements up to the project element.

Property type

This property returns a string value.

Related objects

This is a common property that can be accessed by any object *except for*:

- Any collection object
- Application on page 87
- ExecutionConfiguration on page 102
- Options (Object) on page 120
- Report on page 135
- ReportConfiguration on page 138
- Result on page 140
- ResultState (Object) on page 143
- StaticAttribute on page 149
- TAMVersion (Object) on page 151

StaticAttribute

Syntax

```
GetValue = Obj.StaticAttribute
or
Obj.StaticAttribute = SetValue
```

Purpose

To set or get the static attributes of a report.

Description

The StaticAttribute property gives you access to the additional attributes of the report.

The additional attributes are:

- Descriptions of all reported elements
- Result states of all reported elements
- Outputs of the executed Report blocks
- Report description of a project and folders, if they were involved in the execution

Property type

This property uses a StaticAttribute object to set or get the static attributes of a report.

Related objects

This property can be accessed by the following object:

- ReportConfiguration on page 138

Related topics

References

StopBits

Syntax

```
GetValue = Obj.StopBits
or
Obj.StopBits = SetValue
```

Purpose

To set or get the number of stop bits used for the RS232 interface.

Description

With this property, you can set and get the number of stop bits to be used for the RS232 interface.

Valid values are: 1, 1.5, 2

If you do not specify the number of stop bits, the default value 1 is used.

Note

Do not specify the following combinations of data bit and stop bit numbers:
- Number of data bits = 5 and number of stop bits = 2
- Number of data bits = 6, 7, or 8 and number of stop bits = 1.5

These combinations lead to invalid transmissions.

Property type

This property uses a double value to set or get a value.

Related objects

This property can be accessed by the following object:
- RS232Configuration on page 222

StopTime

Syntax

```
GetValue = Obj.StopTime
```

Purpose

To get the stop time of an execution.

Description

The StopTime property shows you the time when the whole execution finished. The stop time of the execution has the YYYY/MM/DD, HH-MM-SS format by default, for example, 2011/10/06, 16-29-10.

You can change the format by using time format methods of your programming language. For example, when using Python, you can use the following methods:

```
StopTime = ResultState.StopTime
print (time.ctime(int(StopTime)))
# leads to 'Thu Oct 06 16:29:10 2011'
# Another format
print (time.strftime("%#c", time.localtime(int(StopTime))))
# leads to 'Thursday, October 06, 2011 16:29:10'
```

The StopTime property is one of the attributes that you can add to the report, refer to AvailableAttributes on page 291.

Property type

This property returns a date value.

Related objects

This property can be accessed by the following object:

- ResultState (Object) on page 143
- ResultState1 on page 144

Related topics

References

AvailableAttributes..291

StorageType

Syntax

```
GetValue = Obj.StorageType
or
Obj.StorageType = SetValue
```

Purpose

To set or get the storage type of the MC3Collector object.

Description	With this property you can set or get the storage type for the values to be collected. It can be the internal data storage of the calibration tool (eST_AUSY) or a file (e_ST_FILE).
Property type	This property uses a string to set or get a value.
Related objects	This property can be accessed by the following object: ▪ MC3Collector on page 206

StyleSheetPath

Syntax

```
GetValue = Obj.StyleSheetPath
or
Obj.StyleSheetPath = SetValue
```

Purpose

To set or get the path of the report's style sheet.

Description

With the StyleSheetPath property, you can specify the path of a style sheet that should be used for the report. With the AutomationDesk API a few style sheets are supplied.

Note

The maximum path length must not exceed 255 characters.

They are located in `<DocumentsFolder>\Custom Report Stylesheets` If you want to use your own style sheet, you have to set this additionally in the IsCustomReport property.

AutomationDesk and the Automation Server access the same settings. If you modify the settings via the Automation Server, the new settings are also valid for AutomationDesk sessions, and vice versa.

Note

The format of the ReportType property is determined by the StyleSheetPath property. If the format of the ReportType property does not match the format of the StyleSheetPath property, an error message is thrown.

Property type

This property uses a string value to set or get the path of the report's style sheet.

The following table shows the mapping between the XSL stylesheet files and the settings in the AutomationDesk user interface (UI):

XSL File (HTML/PDF)	AutomationDesk UI Setting
▪ `BlockReportHTML10.xsl` ▪ `BlockReportPDF10.xsl`	Classic report
▪ `ExtendedBlockReportHTML-nonTF.xsl` ▪ `ExtendedBlockReportPDF-nonTF.xsl`	Detailed report
▪ `ExtendedBlockReportHTML-TB.xsl` ▪ `ExtendedBlockReportPDF-TB.xsl`	Detailed report for Test Builder
▪ `OnePagerHTML-nonTF.xsl` ▪ `OnePagerPDF-nonTF.xsl`	Brief report

Related objects

This property can be accessed by the following object:

- ReportConfiguration on page 138

Related topics

References

ReportType.. 380

SubBlocks

Syntax

```
GetValue = Obj.SubBlocks
```

Purpose

To get the subblocks of the specified object.

Description

The SubBlocks property gives you access to the Blocks collection object, which manages the subblocks of the specified object.

- Project / Folder

  A subblock can be a folder or a sequence. You can use subblocks to structure your project.
- Sequence

  The returned Blocks object is empty because a sequence cannot have further sequences or folders.

- LibraryFolder

  The SubBlocks property gives you access to the ReadOnlyBlocks collection object, which manages the subblocks of the library. A subblock can be a folder or a sequence.
- CustomLibrary / CustomLibraryFolder

  The SubBlocks property gives you access to the Blocks collection object, which manages the subblocks of the custom library. A custom library folder can contain the same objects as a custom library. Here, a subblock can be a sequence, a sequence frame, a test sequence or a custom library folder.

Property type

This property returns a Blocks object (ReadOnlyBlocks object).

Related objects

This property can be accessed by the following objects:

- Project on page 121
- Folder on page 105
- Sequence on page 145
- LibFolder on page 115
- CustomLibraryFolder on page 94

Related topics

References

Synect (Property)

Syntax

```
GetValue = Obj.Synect
```

Purpose

To get the Synect object..

Description

The Synect property provides an object with properties to configure the synchronization with SYNECT.

Property type

This property returns a Synect object.

Related objects

This property can be accessed by the following objects:

- Bool1 on page 156
- Condition1 (Object) on page 159
- CustomLibraryFolder2 on page 96
- DataContainer1 on page 162
- Dictionary1 on page 165
- File2 on page 170
- Float1 on page 173
- Int1 on page 176
- LabeledValue1 on page 179
- List1 on page 182
- Sequence1 on page 147
- String1 on page 185
- Tuple1 on page 188
- Variant1 on page 192
- Verdict1 (Object) on page 195

Related topics

Basics

Basics on Using AutomationDesk with SYNECT (AutomationDesk Basic Practices 🕮)

References

Clear Ignore Flag (AutomationDesk Basic Practices 🕮)
Set Ignore Flag (AutomationDesk Basic Practices 🕮)

- T -

Where to go from here

Information in this section

TAMVersion (Property)

Syntax	`GetValue = Obj.TAMVersion`
Purpose	To get the AutomationDesk version.
Description	The TAMVersion object contains a major release number, a minor release number, and a revision number. For example, for major release `"6"`, minor release `"3"`, and revision `"1"`, the TAMVersion returns `"6.3.1"`.
Property type	This property returns a TAMVersion (refer to TAMVersion (Object) on page 151) object.
Related objects	This property can be accessed by the following object: ▪ Application on page 87 ▪ Application1 on page 88 ▪ Application2 on page 89

Type

Syntax	`GetValue = Obj.Type`
Purpose	To get the type of the specified object.
Description	With the Type property, you can determine the element type of an instantiated object.
Property type	This property returns a value of the ElementType enumeration. For information on the predefined constants, refer to Overview of API Constants on page 24. If you want to use the predefined constants, you must make some preparations beforehand. For further information, refer to Using API Constants on page 67.

Related objects

This is a common property that can be accessed by any object *except for*:
- Any collection object
- Application on page 87
- ExecutionConfiguration on page 102
- Options (Object) on page 120
- Report on page 135
- ReportConfiguration on page 138
- Result on page 140
- ResultState (Object) on page 143
- StaticAttribute on page 149
- TAMVersion (Object) on page 151

Related topics

Basics

Using API Constants.. 67

- U -

UndefinedCount

Syntax

```
GetValue = Obj.UndefinedCount
```

Purpose

To get the undefined state of an execution, started in a folder, sequence, or project.

Description

Note

The semantic of this property has changed with AutomationDesk 4.0. It is still provided for compatibility reasons. For a proper expression evaluating ResultState1, refer to Verdict (Property) on page 409.

The UndefinedCount property shows you whether the execution led to an unconsidered result
- 0: No undefined state is occured
- 1: An undefined state is occured

Property type

This property returns a long value.

Related objects

This property can be accessed by the following object:
- ResultState (Object) on page 143
- ResultState1 on page 144

- V -

Where to go from here

Information in this section

Value

Syntax

```
GetValue = Obj.Value
or
Obj.Value = SetValue
```

Purpose

To set or get a value.

Description

The Value property gives you access to the object's value. If you instantiate a data object, a default value is used. You can modify the value, unless the object is a library template.

If you use this property with a Color object, you have access to a variant that represents a color in an array specifying the red, green and blue portions of the color or in a string specifying the CSS color name. In Python, the array is represented by a tuple.

If you use this property with a LabeledValue object, you can only set it to a value that is defined in the value mapping dictionary of the LabeledValue object, otherwise an error occurs.

Property type

The property type depends on the object, for which you use the property.

Object	Property Type
Color	variant (tuple or string)
Float	double
Int	long
LabeledValue	variant
String	string
Bool	boolean
Variant	variant

Related objects

This property can be accessed by the following objects:

- Color on page 221 (Report library)
- Float on page 171
- Int on page 174
- LabeledValue on page 177
- String on page 184
- Bool on page 155
- Variant on page 190

Related topics

References

Mapping (Property).. 354

ValueList

Syntax	`GetValue = Obj.Value`
Purpose	To get the list of valid values.
Description	The ValueList property lets you get the list of valid values that are defined for a data object.
Property type	This property returns a list of values of the same type as the related data object.
Related objects	This property can be accessed by the following objects: ▪ String1 on page 185 ▪ Int1 on page 176 ▪ Float1 on page 173 ▪ List1 on page 182 ▪ Dictionary1 on page 165 ▪ Tuple1 on page 188 ▪ Variant1 on page 192

ValueString

Syntax	`GetValue = Obj.ValueString`
	`or`
	`Obj.ValueString = SetValue`

Purpose	To set or get the values of a Dictionary, List or Tuple as a string without using the RootElement object.
Description	If you want to get or set the value of a Dictionary, List, Tuple, or dictionary-based object without using its RootElement object, you can do it via a string. For example, you get the string *'{"k1":1}'* for a dictionary item with the key - value pair *k1:1*.
Property type	This property uses a string to set or get a value.

Related objects

This property can be accessed by the following objects:
- Dictionary on page 163
- List on page 180
- Tuple on page 187
- MAPortConfiguration on page 251

ValueType

Syntax

```
GetValue = Obj.ValueType
or
Obj.ValueType = SetValue
```

Purpose

To set or get the type of the characteristic value.

Description

This property gives you access to the value type as used from the object model. This data object is available for WriteCharacteristic blocks only.

Possible values are:
- eVT_CONST
  One value (a constant) is used
- eVT_OFFSET_NEG
  A negative offset value is used.
- eVT_OFFSET_POS
  A positive offset value is used.
- eVT_VAL (default value)
  n values are used.

Property type

This property uses a string to set or get a value.

Related objects

This property can be accessed by the following object:
- MC3Characteristics on page 204

VehicleInformationName

Syntax	`GetValue = Obj.VehicleInformationName`
	`or`
	`Obj.VehicleInformationName = SetValue`

Purpose	To set or get the name of the instantiated VehicleInformation object.
Property type	This property uses a string to set or get a value.
Related objects	This property can be accessed by the following objects: ▪ D3VehicleInformation on page 212

VehicleInformations

Syntax	`GetValue = Obj.VehicleInformations`
Purpose	To get the VehicleInformations collection object of the diagnostic project.
Description	The VehicleInformations property is a collection based on the DataObjects collection. It provides the same methods, for example, Create, Remove and Copy, as the DataObjects collection.
Property type	This property uses a DataObjects (Object) object to get the value.
Related objects	This property can be accessed by the following objects: ▪ D3Project on page 211
Related topics	References DataObjects (Object).. 101

Verdict (Property)

Syntax

```
GetValue = Obj.Verdict
```

Purpose

To get a verdict.

Description

The Verdict property provides the expression qualifying the ResultState.

Property type

This property returns a long value.

Constant	Description

adVerdictExecuted = 0
adVerdictPassed = 1
adVerdictUndefined = 2
adVerdictFailed = 3
adVerdictError = 4
	The VerdictConstant constants are used to provide a verdict for automation elements.

Related objects

This property can be accessed by the following object:

- ResultState1 on page 144

Visible

Syntax

```
GetValue = Obj.Visible
or
Obj.Visible = SetValue
```

Purpose

To set or get the display mode of AutomationDesk.

Description

With the Visible property, you can specify whether AutomationDesk is displayed in visible mode or in invisible mode. When you call the Dispatch method for AutomationDesk, the application is started in invisible mode. To display the user interface, you can set the application to visible mode.

If you call this property for the application of the AutomationDesk - Automation Server, an exception is raised.

Property type	This property uses a Boolean value to set or get the value: ▪ False (0): The AutomationDesk user interface is set to invisible mode (default). ▪ True (1): The AutomationDesk user interface is set to visible mode.
Related objects	This property can be accessed by the following object: ▪ Application1 on page 88 ▪ Application2 on page 89

VisibleAttributes

Syntax

```
GetValue = Obj.VisibleAttributes
or
Obj.VisibleAttributes = SetValue
```

Purpose

To set or get the specified subset of attributes to be added to the report.

Description

With the VisibleAttributes property, you can specify the attributes to be added to the report. To add only the specified attributes to the report, you have to set this in the IsAllAttributes property. To list all available attributes, refer to AvailableAttributes on page 291.

Property type

This property uses a variant value to set or get the specified subset of attributes to be added to the report.

Related objects

This property can be accessed by the following object:
- ReportConfiguration on page 138

Related topics

References

IsAllAttributes..340

- X -

Where to go from here

Information in this section

XStartIndex

Syntax

```
GetValue = Obj.XStartIndex
or
Obj.XStartIndex = SetValue
```

Purpose

To set or get the start index of the x-axis.

Description

The XStartIndex property is only used for characteristics of Curve and Map type.

Property type

This property uses a long value to set or get a value.

Related objects

This property can be accessed by the following object:

- MC3Characteristics on page 204

XStopIndex

Syntax

```
GetValue = Obj.XStopIndex
or
Obj.XStopIndex = SetValue
```

Purpose	To set or get the stop index of the x-axis.
Description	The XStopIndex property is only used for characteristics of Curve and Map type.
Property type	This property uses a long value to set or get a value.
Related objects	This property can be accessed by the following object: ▪ MC3Characteristics on page 204

XVector

Syntax	`GetValue = Obj.XVector`
Purpose	To get the Signal data object's vector of time values.
Description	With this property, you can get the Signal data object's vector of x-axis values.
Property type	This property returns a variant value.
Related objects	This property can be accessed by the following object: ▪ Signal on page 152

- Y -

Where to go from here

Information in this section

YStartIndex

Syntax	`GetValue = Obj.YStartIndex`
	`or`
	`Obj.YStartIndex = SetValue`

Purpose	To set or get the start index of the y-axis.
Description	The YStartIndex property is only used for characteristics of Map type.
Property type	This property uses a long value to set or get a value.
Related objects	This property can be accessed by the following object: ▪ MC3Characteristics on page 204

YStopIndex

Syntax	`GetValue = Obj.YStopIndex`
	`or`
	`Obj.YStopIndex = SetValue`

Purpose	To set or get the stop index of the y-axis.
Description	The YStopIndex property is only used for characteristics of Map type.
Property type	This property uses a long value to set or get a value.
Related objects	This property can be accessed by the following object: ▪ MC3Characteristics on page 204

Methods in Alphabetical Order

Introduction

The COM objects of the AutomationDesk COM API provide specific methods. The following list shows you all the available methods. In their descriptions you find the COM objects they are supported by.

Where to go from here

Information in this section

- A -

AddParameter

Syntax

```
Obj.AddParameter(ParameterName, ParameterValue, ParameterUnit)
```

Purpose

To add a parameter to a diagnostic object.

Description

With this method, you can enlarge the parameter list of the instantiated diagnostic object.

Parameters

The following parameters are used:

Parameter	Type	Default Value	Description
ParameterName	string	" "	Specifies the name of the parameter you want to add to the diagnostic object.
ParameterValue	variant	None	Specifies the value of the new parameter.
ParameterUnit	string	" "	Specifies the unit of the value.

Return value

None

Related objects

This method can be accessed by the following objects:

- D3ControlPrimitive on page 215
- D3Service on page 217
- D3SingleJob on page 218

- B -

BreakLink

Syntax	`Obj.BreakLink()`
Purpose	To break the link between the custom library and the instantiated sequence.
Description	The BreakLink method breaks the link of a referenced sequence to its sequence template in the custom library. After the link is broken the instantiated sequence can no longer be synchronized with the custom library. For further information, refer to Working with Custom Libraries (AutomationDesk Basic Practices 🕮). **Note** A broken link cannot be restored.
Parameters	None
Return value	None
Related objects	This method can be accessed by the following object: ▪ Sequence on page 145
Related topics	Basics Working with Custom Libraries (AutomationDesk Basic Practices 🕮) References Libraries (Object)..110

- C -

Where to go from here

Information in this section

CheckSyntax

Syntax

```
Obj.CheckSyntax()
```

Purpose

To check the syntax of the specified condition object.

Parameters

None

Return value

This method returns a Boolean value:
- 0: The syntax check has failed.
- 1: The syntax check has passed.

Related objects

This method can be accessed by the following object:
- Condition (Object) on page 158

ClearMessages

Syntax

```
Obj.ClearMessages()
```

Purpose

To clear the messages shown in the Message Viewer.

Parameters

None

Return value

None

Related objects

This method can be accessed by the following object:
- Log (Object) on page 118

ClearValue

Syntax

```
Obj.ClearValue()
```

Purpose

To clear the values of the data object.

Description

This method is used to empty memory-consuming data objects that are filled with values during run time and need not to be saved after execution.

By clearing the data object values, the data object is reset to its data-type-specific default value. Usually, this leads to an empty data object with the exception of data objects parameterized by a value list. These data objects are set to the first value of their value list.

If the data object contains a reference, the referenced data object values and all assigned values of data objects in the reference chain are also cleared.

Parameters

None

Return value

None

Related objects

This method can be accessed by the following objects:

- Dictionary on page 163
- Dictionary1 on page 165
- List on page 180
- List1 on page 182
- Tuple on page 187
- Tuple1 on page 188
- Variant on page 190
- Variant1 on page 192
- Signal on page 152
- CaptureResult (XIL API) on page 234
- MAPortConfiguration on page 251
- Mapping (Object) on page 254
- SignalGroupValue on page 267
- SignalValue on page 270
- MC3Collector on page 206
- D3Results on page 220

Related topics

References

Clear Value (AutomationDesk Basic Practices 🕮)
ClearValues..419

ClearValues

Syntax

```
Obj.ClearValues(ClearOutputValues, ClearLocalValues)
```

Purpose

To recursively clear the values of all contained output data objects and/or local data objects.

Description

This method is used to empty memory-consuming data objects in the current object that are filled with values at run time and need not to be saved after execution. Via the method parameters, you can configure which values are affected.

By clearing the data object values, the data objects are reset to their data-type-specific default values. Usually, this leads to empty data objects with the exception of data objects parameterized by a value list. These data objects are set to the first value of their value list.

Parameters

The following parameters are used:

Parameter	Type	Default Value	Description
ClearOutputValues	boolean	True	Specifies to recursively clear the values of all output data objects in the subsystem of the current object, i.e., all data objects where the InOutState is set to output.
ClearLocalValues	boolean	True	Specifies to recursively clear the local values of referencing data objects in the subsystem of the current object, i.e. of all data objects that reference to another data object. The values of referenced data objects or data objects without reference are not cleared. In this way, all accessible data is left unchanged and only local values which are not accessible are cleared.

Return value

None

Related objects

This method can be accessed by the following objects:

- Project on page 121
- Project1 on page 123
- Folder on page 105
- Folder1 on page 107
- Sequence on page 145

- Sequence1 on page 147
- CustomLibraryFolder2 on page 96
- DataContainer on page 160

Related topics

References

Clear Values - Clear Local Values of Referencing Data Objects (AutomationDesk Basic Practices 🕮)
Clear Values - Clear Output Values (AutomationDesk Basic Practices 🕮)
ClearValue...418

Close

Syntax

```
Obj.Close()
```

Purpose

To close a project, a MATLAB instance, or a custom library.

Description

Project The Close method closes the project, but it does not save it.

Note

If the project was modified, it has to be saved, otherwise all the modifications in it are lost.

CustomLibraryFolder1, CustomLibraryFolder2 The Close method closes the custom library, but it does not save it.

Note

If the custom library was modified, it has to be saved, otherwise all the modifications in it are lost.

MATLAB A message is displayed if MATLAB is not already running.

Parameters

None

Return value

None

Related objects

This method can be accessed by the following objects:

- Project on page 121
- CustomLibraryFolder1 on page 95
- CustomLibraryFolder2 on page 96
- MATLAB on page 196

CloseAll

Syntax

```
Obj.CloseAll()
```

Purpose

To close all projects and custom libraries.

Description

Projects The CloseAll method closes all the projects in the project collection, but it does not save them.

> **Note**
>
> If any projects have been modified, they have to be saved, otherwise all the modifications are lost.

Libraries The CloseAll method closes all custom libraries. If you have changed the library, AutomationDesk prompts you to save the changes.

If you use AutomationDesk - Automation Server no confirmation dialogs are displayed, the library is automatically saved.

Parameters

None

Return value

None

Related objects

This method can be accessed by the following object:

- Projects (Object) on page 125
- Projects1 on page 126
- Projects2 on page 127
- Libraries1 on page 111
- Libraries2 on page 112
- Libraries3 on page 113

Connect

Syntax

```
Obj.Connect()
```

Purpose

To connect AutomationDesk with the configured diagnostic or calibration system.

Description

After you have configured the connection to the diagnostic or calibration system, you must use this method to connect to it. The connection must be established before you can configure the diagnostic or calibration project.

Parameters

None

Return value

This method returns a Boolean value:

- 0: The method has failed.
- 1: The method has passed.

Related objects

This method can be accessed by the following objects:

- D3System on page 209
- MC3System on page 200

Related topics

References

Copy

Syntax

```
Obj.Copy(SourceObject, Position, ConfirmBreakLink, InsertBeforeObject, InsertAfterObject)
```

Purpose

To create a copy of the specified object at the specified position.

Description

The Copy method copies a specific Block or DataObject object with its subelements to a specified position. When you use the Copy method, you have four alternatives do this, in all cases the position of the object is specified:

1. To position the object after the last object in the same hierarchy tree, you can use this alternative:

   ```
   Obj.Copy(SourceObject)
   ```

2. To position the object at a specific position, you have to specify the position like this:

   ```
   Obj.Copy(SourceObject, Position)
   ```

3. To position the object before a specific object, you have to specify the specific object like this:

   ```
   Obj.Copy(SourceObject, -1, ConfirmBreakLink,
   InsertBeforeObject)
   ```

   The `Position` parameter must have the default value, otherwise the object will not be positioned before the specific object, but at the given position.

4. To position the object after a specific object, you have to specify the specific object like this:

   ```
   Obj.Copy(SourceObject, -1, ConfirmBreakLink, None,
   InsertAfterObject)
   ```

   The `Position` parameter must have the default value and the `InsertBeforeObject` parameter must be `None`, otherwise the object will not be positioned as required.

Parameters

The following parameters are used:

Parameter	Type	Default Value	Description
SourceObject	Block or DataObject	-	Specifies the Block or DataObject object to be copied.
Position	variant	-1	Specifies the position at which the object must be added.
ConfirmBreakLink	boolean	False	Specifies whether the link between the object and the library element should break. The link to the object's parent object will also be broken.
InsertBeforeObject	variant	None	Specifies the predecessor of the copied object.
InsertAfterObject	variant	None	Specifies the successor of the copied object.

Return value

Block	This method returns a Block object.
DataObject	This method returns a DataObject object.

Related objects

This method can be accessed by the following objects:

- Blocks on page 92
- DataObjects (Object) on page 101

Related topics

References

Create

Syntax

Blocks, DataObjects

```
Obj.Create(SourceObject, Position, ConfirmBreakLink, InsertBeforeObject, InsertAfterObject)
```

Projects

```
Obj.Create(ProjectName, TemplateName, OptionsValue)
```

Libraries

```
Obj.Create(LibName, OptionsValue)
```

Purpose

To create a new object based on the collection.

Description

You must note the following different variants of this method:

Blocks, DataObjects The Create method creates a new Block or DataObject object. There are four ways of using the Create method, in each of which you have to specify the position of the object:

1. You can position the object after the last object in the same hierarchy tree like this:

   `Obj.Create(SourceObject)`
2. To position the object at a specific position, specify the position like this:

   `Obj.Create(SourceObject, Position)`
3. To position the object before a specific object, specify the specific object like this:

   ```
   Obj.Create(SourceObject, -1, ConfirmBreaklink,
   InsertBeforeObject)
   ```

   The `Position` parameter must have the default value, otherwise the object will not be positioned before the specific object, but at the given position.
4. To position the object after a specific object, specify the specific object like this:

```
Obj.Create(SourceObject, -1, ConfirmBreakLink, None,
InsertAfterObject)
```

The `Position` parameter must have the default value and the `InsertBeforeObject` parameter must be `None`, otherwise the object will not be positioned as required.

Projects To handle an automation task with the AutomationDesk API, you first need to define an automation project. All the elements and information relevant to the automation task are collected in the project.

When you create a new project, you have to select a project template. This provides information about the project structure and the project elements. At the moment you can only select the Standard Project template.

Libraries With the Libraries collection, you have access to each library in AutomationDesk, the built-in libraries and the custom libraries. The Create method can only be used for custom libraries. With the OptionsValue parameter you can decide whether to overwrite an opened custom library with the same name.

Parameters

Blocks, DataObjects The following parameters are used for Blocks and DataObjects objects:

Parameter	Type	Default Value	Description
SourceObject	Block, DataObject	-	Specifies the Block or DataObject object to be created.
Position	variant	-1	Specifies the position to create the object at.
ConfirmBreakLink	boolean	False	Specifies whether the link between the object and the library element should break. The link to the object's parent object will also be broken.
InsertBeforeObject	variant	None	Specifies the successor of the created object.
InsertAfterObject	variant	None	Specifies the predecessor of the created object.

Projects The following parameters are used for Projects objects:

Parameter	Type	Default Value	Description
ProjectName	string	" "	Specifies the file the project is created in.
TemplateName	string	" "	Specifies the template the project is built on.
OptionsValue	enumeration	adCancel	Specifies whether an existing project should be overwritten. The possible values of the FileOptions enumeration are adOverWrite (=1) to overwrite a project and adCancel (=0) to abort the operation.

If you want to use the predefined constants, you must make some preparations beforehand. For further information, refer to Using API Constants on page 67.

Libraries The following parameters are used for Libraries objects:

Parameter	Type	Default Value	Description
LibName	string	" "	Specifies the name of the library.
OptionsValue	enumeration	adCancel	Specifies whether an existing library should be overwritten. The possible values of the FileOptions enumeration are adOverWrite (=1) to overwrite a project and adCancel (=0) to abort the operation.

Return value

Blocks This method returns a Block object.

DataObjects This method returns a DataObject object.

Libraries This method returns a Libraries (Object) object.

Projects This method returns a Project object.

Related objects

This method can be accessed by the following object:
- Blocks on page 92
- DataObjects (Object) on page 101
- Libraries1 on page 111
- Libraries2 on page 112
- Libraries3 on page 113
- Projects (Object) on page 125
- Projects1 on page 126
- Projects2 on page 127
- Projects3 on page 129

Related topics

Basics

References

CreateSubFolder

Syntax	`Obj.CreateSubFolder()`
Purpose	To create a subfolder.
Description	With this method you can create a subfolder of the current custom library folder. The new created CustomLibraryFolder object is returned.
Parameters	None
Return value	This method returns a created CustomLibraryFolder2 object.
Related objects	This method can be accessed by the following object: ▪ CustomLibraryFolder2 on page 96
Related topics	References CustomLibraryFolder2 96

CreateSubItem

Syntax	`Obj.CreateSubItem(SubItem)`
Purpose	To create a subitem that can be added to the root element.
Description	You can create a tuple, a list or a dictionary as subitem and add it to the root element of the object. You can specify the subitem type to be created as: ▪ String ("Tuple", "List", "Dictionary") ▪ Integer (0,1,2) ▪ Enumeration value (adMainLibraryTupleValue, adMainLibraryListValue, adMainLibraryDictionaryValue)

Parameters

The following parameters are used:

Parameter	Type	Default Value	Description
SubItem	variant	None	Specifies the subitem to be added to the root element.

Property type

This property returns a variant value.

Related objects

This method can be accessed by the following objects:

- Dictionary on page 163
- List on page 180
- Tuple on page 187
- MAPortConfiguration on page 251

- D -

Where to go from here

Information in this section

DeleteParameter

Syntax

```
Obj.DeleteParameter(ParameterName)
```

Purpose

To delete a parameter from the parameter list of a diagnostic object.

Parameters

The following parameters are used:

Parameter	Type	Default Value	Description
ParameterName	string	" "	Specifies the name of the parameter you want to delete.

Return value

None

Related objects

This method can be accessed by the following objects:

- D3ControlPrimitive on page 215
- D3Service on page 217
- D3SingleJob on page 218

DeSelect

Syntax

```
Obj.DeSelect()
```

Purpose

To deselect the currently selected diagnostic or calibration project.

Description

Diagnostic project After deselecting the currently selected project, all eventually started Services and ComPrimitives are stopped. All VehicleInformation objects are destroyed and the connection to the diagnostic system is interrupted.

Calibration project After deselecting the currently selected project, all eventually started collectors are stopped. All LogicalLink objects are destroyed and the connection to the MC system is interrupted.

Parameters

None

Return value

This method returns a Boolean value:

- 0: The method has failed.
- 1: The method has passed.

Related objects

This method can be accessed by the following objects:

- D3Project on page 211
- MC3Project on page 201

Related topics

References

Disconnect

Syntax

```
Obj.Disconnect()
```

Purpose

To disconnect AutomationDesk from the configured diagnostic or calibration system.

Description

If a diagnostic or calibration system is connected to AutomationDesk, you can disconnect it by using this method. You can check the status of the connection by using the IsConnected property of the related system object.

Parameters

None

Return value

This method returns a Boolean value:

- 0: The method has failed.
- 1: The method has passed.

Related objects

This method can be accessed by the following objects:

- D3System on page 209
- MC3System on page 200

Related topics

References

- E -

Where to go from here

Information in this section

EditParameter

Syntax

```
Obj.ReferenceName(ParameterName, NewParameterName, NewParameterValue, NewParameterUnit)
```

Purpose

To edit a parameter of a diagnostic object.

Description

With this method, you can not only modify the value or the unit of a parameter, but also the name of the parameter.

Parameters

The following parameters are used:

Parameter	Type	Default Value	Description
ParameterName	string	" "	Specifies the name of the parameter you want to modify.
NewParameterName	string	" "	Specifies the new name of the parameter.
NewParameterValue	variant	None	Specifies the new value of the parameter.
NewParameterUnit	string	" "	Specifies the new unit of the value.

Return value

None

Related objects

This method can be accessed by the following objects:
- D3ControlPrimitive on page 215
- D3Service on page 217
- D3SingleJob on page 218

Execute

Syntax

```
Obj.Execute(ExecutionName, Description)
```

Purpose

To execute one or more sequences.

Description

The Execute method starts the execution of all the sequences which belong to the instantiated object. They are executed in the order in which they appear in the project structure. Each block executes its tasks and outputs the results according to the execution configuration settings (see ExecutionConfiguration on page 102).

Parameters

The following parameters are used:

Parameter	Type	Default Value	Description
ExecutionName	string	" "	Specifies the name of the execution.
Description	string	" "	Specifies the description of the execution.

Return value

This method returns a Result object.

Related objects

This method can be accessed by the following objects:
- Project on page 121
- Folder on page 105
- Sequence on page 145

Related topics

References

Export

Syntax

Project

```
Obj.Export(ZipFileName, OptionsValue)
```

Report1

```
Obj.Export(FilePath)
```

LibraryFavorites

```
Obj.Export(XmlFileName, OptionsValue)
```

Purpose

Project To export the project as a ZIP file.

Report1 To save a report to a specified folder in the same format (HTML or PDF).

LibraryFavorites To export the library favorites as an XML file.

Description

Project To archive or send an AutomationDesk project, you can export it to an archive file. The Export method exports the project to a ZIP archive. The project is saved automatically before it is exported.

Report1 —

LibraryFavorites The Export method exports the library favorites to an XML file.

Parameters

Project The following parameters are used for Projects objects:

Parameter	Type	Default Value	Description
ZipFileName	string	" "	Specifies the name of the ZIP file.
OptionsValue	enumeration	-	Specifies whether to overwrite an existing ZIP file. The possible values of the FileOptions enumeration are adOverWrite (=1) to overwrite the project and adCancel (=0) to abort the operation.

If you want to use the predefined constants, you must make some preparations beforehand. For further information, refer to Using API Constants on page 67.

Report1 The following parameters are used for report objects:

Parameter	Type	Default Value	Description
FilePath	string	" "	Specifies the name and the path of the report object.

LibraryFavorites The following parameters are used for LibraryFavorites objects:

Parameter	Type	Default Value	Description
XmlFileName	string	" "	Specifies the name of the XML file.
OptionsValue	enumeration	-	Specifies whether an existing XML file should be overwritten. The possible values of the FileOptions enumeration are adOverWrite (=1) to overwrite the project and adCancel (=0) to abort the operation.

If you want to use the predefined constants, you must make some preparations beforehand. For further information, refer to Using API Constants on page 67.

Return value

None

Related objects

This method can be accessed by the following object:

- Project on page 121
- Project1 on page 123
- Report1 on page 136
- LibraryFavorites on page 114

ExportFile

Syntax

```
Obj.ExportFile(FileName, FileFormats, FileOptions)
```

Purpose

To export a project, a folder, a sequence, or a custom library.

Description

The ExportFile method exports a project or custom library in ZIP or XML format. Folders and sequences can only be exported to XML files.

With AutomationDesk 6.1, a new XML format is introduced for exporting and importing AutomationDesk elements. The XML format used for exporting and importing elements with AutomationDesk 6.0 and earlier is now called *legacy*

XML. It is available only for importing existing XML export files. The legacy XML format is not available for exporting elements and will be discontinued in future versions of AutomationDesk.

Both XML file formats are specified by the `adXML` enumeration. The XML format to be used is automatically identified by the specified file suffix. If you want to export to a legacy XML file, an exception occurs. If you import a file in the legacy XML format, a warning is written to the log file, which informs you about the planned discontinuation.

Note

The XML import/export feature can be accessed by the Project1, Folder1, Sequence1 interfaces.

Parameters

The following parameters are used:

Parameter	Type	Default Value	Description
FileName	string	" "	Specifies the file the element is exported to.
FileFormats	enumeration	-	Specifies the format of the file you want to export to. This parameter is optionally. Only for a Project1 object you can decide whether to export to a ZIP file (adZip=0) or an XML file (adXML=1). For Folder, Sequence objects, only XML export is supported.
FileOptions	enumeration	-	Specifies whether an existing project should be overwritten. This parameter is optionally. The possible values of the FileOptions enumeration are adOverWrite (=1) to overwrite an element and adCancel (=0) to abort the operation.

If you want to use the predefined constants, you must make some preparations beforehand. For further information, refer to Using API Constants on page 67.

Return value

None

Related objects

This method can be accessed by the following object:

- Project1 on page 123
- Folder1 on page 107
- Sequence1 on page 147
- CustomLibraryFolder1 on page 95

Related topics

Basics

Using API Constants.. 67

- F -

FindElement

Syntax

```
Obj.FindElement(HierarchyPath)
```

Purpose

To get the object of the element with the specified hierarchy path.

Description

With this method, you can get the object of an element in the projects or libraries by specifying the object's hierarchy path.

Parameters

The following parameters are used:

Parameter	Type	Default Value	Description
HierarchyPath	string	" "	Specifies the hierarchy path of the wanted element in the projects or libraries. For a library element, you can alternatively specify the wanted template name.

Return value

This method returns the object of the element with the specified hierarchy path.

If no object is found, `None` is returned.

Related objects

This method can be accessed by the following objects:

- Blocks on page 92
- DataObjects (Object) on page 101
- Libraries (Object) on page 110
- Libraries1 on page 111
- Libraries2 on page 112
- Libraries3 on page 113
- Projects (Object) on page 125
- Projects1 on page 126
- Projects2 on page 127
- Projects3 on page 129
- PythonModules on page 131
- ReadOnlyBlocks on page 134
- ReadOnlyDataObjects on page 135

- Reports (Object) on page 139
- Results (Object) on page 142

Related topics

References

- G -

Where to go from here

Information in this section

GenerateReport

Syntax

```
Obj.GenerateReport()
```

Purpose

To generate a report.

Description

The GenerateReport method generates a report. The report is based on the corresponding result. It does not contain any result items, but some attributes which describe the executed result. These attributes, the output format, and the layout can be specified in the ReportConfiguration object. The advantage of this method is that you can generate a report if it is required.

When you create a Report object, it gets a default name. If there is more than one report in the same project hierarchy, a consecutive number is added to the name. For example, if you add 3 reports to the same project hierarchy, they are named "Report", "Report1", and "Report2".

If you want to work with the generated report file, you find it in the storage structure of your AutomationDesk project. The report folders are named by dynamically generated GUIDs.

Note

With AutomationDesk 2.0, the folder structure of a project was changed. If you have used user tools with earlier AutomationDesk versions, for example, batch files for an automatic backup, you must adapt them.

Parameters

None

Return value

This method returns a Report object.

Related objects

This method can be accessed by the following object:

- Reports (Object) on page 139

Related topics

References

Report.. 135

GetParameterDefaultValues

Syntax

```
Obj.GetParameterDefaultValues(ParameterName)
```

Purpose

To get the default values of a Parameter object.

Parameters

The following parameter is used:

Parameter	Type	Default Value	Description
ParameterName	string	" "	Specifies the name of the parameter from which you want to get its default values. You can get the available parameter names, for example, by using `Obj.D3Service.Parameters.Keys`.

Return value

This method returns a variant value.

Related objects

This method can be accessed by the following objects:

- D3ControlPrimitive on page 215
- D3Service on page 217
- D3SingleJob on page 218

GetParameterValue

Syntax

```
Obj.GetParameterValue(ParameterName)
```

Purpose

To get the value and unit of the specified parameter.

Description

With this method, you can get the value and unit of a specific parameter. These both values are separately returned. For example, you can use the following code:

```
Value, Unit = D3ServiceObj.GetParameterValue("MyParameter")
```

Parameters

The following parameter is used:

Parameter	Type	Default Value	Description
ParameterName	string	" "	Specifies the name of the parameter from which you want to get its default values. You can get the available parameter names, for example, by using `Obj.D3ControlPrimitive.Parameters.Keys`.

Return value

This method returns the following values:

Return value	Type	Default Value	Description
ParameterValue	variant	None	Returns the value of the specified parameter.
ParameterUnit	string	" "	Returns the unit that belongs to the parameter value.

Related objects	This method can be accessed by the following objects: ▪ D3ControlPrimitive on page 215 ▪ D3Service on page 217 ▪ D3SingleJob on page 218

- H -

Highlight

Syntax	`Obj.Highlight()`
Purpose	To highlight an element in AutomationDesk's user interface.
Description	The Highlight method corresponds to selecting an element in AutomationDesk's user interface.
Parameters	None
Return value	None
Related objects	This method can be accessed by the following objects: ▪ Block on page 91 ▪ CustomLibraryFolder on page 94 ▪ CustomLibraryFolder1 on page 95 ▪ CustomLibraryFolder2 on page 96 ▪ DataObject on page 98 ▪ DataObject2 on page 99 ▪ Folder on page 105 ▪ Folder1 on page 107 ▪ LibFolder on page 115 ▪ LibFolder1 on page 116 ▪ Project on page 121

- Project1 on page 123
- PythonModule on page 130
- PythonPackage on page 132
- Sequence on page 145
- Sequence1 on page 147

Related topics

References

- I -

Where to go from here

Information in this section

Import

Syntax

Projects

```
Obj.Import(ZipFileName, OptionsValue)
```

Libraries

```
Obj.Import(FileName, FileFormats, FileOptions, boolManuelUpdateConfirmation)
```

LibraryFavorites

```
Obj.Import(XmlFileName, FileOptions)
```

XilApiMapping

```
Obj.Import(XmlFileName, FileOptions)
```

Purpose

Projects To import a project from a ZIP file.

Libraries To import a custom library from a file.

LibraryFavorites To import the library favorites from an XML file.

XilApiMapping To import the XIL API variable mapping from an XML file.

Description

Projects The Import method imports an AutomationDesk project that is packed in a ZIP archive. After the project is imported, the file is extracted to your local drive in the archive's folder. If the OptionsValue parameter is not set, the operation will be aborted if a project with the same name exists.

Libraries The description of the ImportEx method is also valid in here, refer to ImportEx on page 445.

LibraryFavorites The Import method imports the libary favorites from an XML file.

XilApiMapping The Import method imports the XIL API variable mapping from an XML file.

Parameters

Projects The following parameters are used for Project objects:

Parameter	Type	Default Value	Description
ZipFileName	string	" "	Specifies the ZIP file the project is imported from.
OptionsValue	enumeration	adCancel	Specifies whether an existing project should be overwritten. The possible values of the FileOptions enumeration are adOverWrite (=1) to overwrite a project and adCancel (=0) to abort the operation.

If you want to use the predefined constants, you must make some preparations beforehand. For further information, refer to Using API Constants on page 67.

Libraries The parameters of the ImportEx method are also valid in here, refer to ImportEx on page 445.

LibraryFavorites The following parameters are used for LibraryFavorites objects:

Parameter	Type	Default Value	Description
XmlFileName	string	" "	Specifies the XML file the library favorites are imported from.

XilApiMapping The following parameters are used for XilApiMapping objects:

Parameter	Type	Default Value	Description
XmlFileName	string	" "	Specifies the XML file the variable mapping is imported from.

Return value

Projects This method returns a Project object.

Libraries This method returns a CustomLibraryFolder object.

LibraryFavorites None

XilApiMapping None

Related objects

This method can be accessed by the following object:

- Projects (Object) on page 125
- Projects1 on page 126
- Projects2 on page 127
- Projects3 on page 129
- Libraries1 on page 111
- Libraries2 on page 112
- Libraries3 on page 113
- LibraryFavorites on page 114
- Mapping (Object) on page 254

Related topics

Basics

References

ImportEx

Syntax

```
Obj.ImportEx(FileName, FileFormats, FileOptions, boolManuelUpdateConfirmation)
```

Purpose

To import a project from an XML or ZIP file with the option to suppress the update confirmation dialog when using a newer AutomationDesk version.

Description

The ImportEx method imports an AutomationDesk project that is packed in a ZIP archive or specified in an XML file. If the project is imported from a ZIP archive, the file is extracted to your local drive in the archive's folder. If the FileOptions parameter is not set, the operation will be aborted if a project with the same name exists. If the boolManuelUpdateConfirmation parameter is set to True, a dialog is displayed for you to confirm the automatic update when you import the project file to a newer AutomationDesk version.

Note

The boolManuelUpdateConfirmation parameter can be used only with the Projects2 interface.

Parameters

The following parameters are used:

Parameter	Type	Default Value	Description
FileName	string	" "	Specifies the file the project is imported from.
FileFormats	enumeration	adZip	Specifies the format of the file you want to import the project from. The possible values fo the FileFormat enumeration are adZip (=0) to import from a ZIP archive and adXML (=1) to import from an XML file.
FileOptions	enumeration	adCancel	Specifies whether an existing project should be overwritten. The possible values of the FileOptions enumeration are adOverWrite (=1) to overwrite a project and adCancel (=0) to abort the operation.
boolManuelUpdateConfirmation	boolean	True	Specifies whether an update confirmation dialog is displayed when importing a project to a newer AutomationDesk version. If set to False (=0), the update confirmation dialog is ignored.

If you want to use the predefined constants, you must make some preparations beforehand. For further information, refer to Using API Constants on page 67.

Return value

This method returns a Project object.

Related objects

This method can be accessed by the following object:
- Projects2 on page 127
- Projects3 on page 129

Related topics

Basics

References

ImportFile

Syntax

```
Obj.ImportFile(FileName, FileFormats, CreateDefaultName)
```

Purpose

To import a folder or a sequence to an AutomationDesk project from an XML file.

With AutomationDesk 6.1, a new XML format is introduced for exporting and importing AutomationDesk elements. The XML format used for exporting and importing elements with AutomationDesk 6.0 and earlier is now called *legacy XML*. It is available only for importing existing XML export files. The legacy XML format is not available for exporting elements and will be discontinued in future versions of AutomationDesk.

Both XML file formats are specified by the `adXML` enumeration. The XML format to be used is automatically identified by the specified file suffix. If you want to export to a legacy XML file, an exception occurs. If you import a file in the legacy XML format, a warning is written to the log file, which informs you about the planned discontinuation.

Description

The ImportFile method imports a folder or a sequence that is specified in an XML file. To import a project, you must use the ImportProject method.

Note

The XML import/export feature can be accessed by the Project1, Folder1, Sequence1 interfaces.

Parameters

The following parameters are used:

Parameter	Type	Default Value	Description
FileName	string	" "	Specifies the file the element is imported from.
FileFormats	enumeration	adZip	Specifies the format of the file you want to import from. The possible values of the FileFormat enumeration are adZip (=0) to import an entire project from a ZIP archive and adXML (=1) to import folders or sequences from an XML file. Only the XML format is supported by this method.
CreateDefaultName	bool	-1	Specifies whether to name the elements with the created default name or with the name specified in the file to be imported. The possible values are: ▪ 0 (false) The specified name is used. ▪ <> 0 (true) The default name is used.

If you want to use the predefined constants, you must make some preparations beforehand. For further information, refer to Using API Constants on page 67.

Return value

This method returns a Block object.

Related objects

This method can be accessed by the following object:

- Project1 on page 123
- Folder1 on page 107

Related topics

Basics

References

ImportProject

Syntax

```
Obj.ImportProject(FileName, FileFormats, FileOptions)
```

Purpose

To import a project from an XML or ZIP file.

Description

The ImportProject method imports an AutomationDesk project that is packed in a ZIP archive or specified in an XML file. If the project is imported from a ZIP archive, the file is extracted to your local drive in the archive's folder. If the FileOptions parameter is not set, the operation will be aborted if a project with the same name exists.

With AutomationDesk 6.1, a new XML format is introduced for exporting and importing AutomationDesk elements. The XML format used for exporting and importing elements with AutomationDesk 6.0 and earlier is now called *legacy XML*. It is available only for importing existing XML export files. The legacy XML format is not available for exporting elements and will be discontinued in future versions of AutomationDesk.

Both XML file formats are specified by the `adXML` enumeration. The XML format to be used is automatically identified by the specified file suffix. If you want to export to a legacy XML file, an exception occurs. If you import a file in the legacy XML format, a warning is written to the log file, which informs you about the planned discontinuation.

Note

The XML import feature can be accessed by the Projects1 interface.

Parameters

The following parameters are used:

Parameter	Type	Default Value	Description
FileName	string	" "	Specifies the file the project is imported from.
FileFormats	enumeration	adZip	Specifies the format of the file you want to import the project from. The possible values fo the FileFormat enumeration are adZip (=0) to import from a ZIP archive and adXML (=1) to import from an XML file.
FileOptions	enumeration	adCancel	Specifies whether an existing project should be overwritten. The possible values of the FileOptions enumeration are adOverWrite (=1) to overwrite a project and adCancel (=0) to abort the operation.

If you want to use the predefined constants, you must make some preparations beforehand. For further information, refer to Using API Constants on page 67.

Return value

This method returns a Project1 object.

Related objects	This method can be accessed by the following object: ▪ Projects1 on page 126 ▪ Projects2 on page 127 ▪ Projects3 on page 129
Related topics	Basics Using API Constants........67 References Project1........123

Init

Syntax	`Obj.Init()`
Purpose	To initialize the XIL API framework.
Description	This method lets you initialize the configured XIL API framework.
Parameters	None
Return value	None
Related objects	This method can be accessed by the following object: ▪ Framework (Object) on page 108

Item

Syntax	`Obj.Item(Index)`

Purpose

To get a specific item of the object.

Description

The Item method gives you access to a specific object of this collection. If you want to know which objects are available in the collection, you can use the Names property. You can access an object by specifying its name or position as a parameter. If you use the position number, note that the index starts with "0".

```
Obj.Item("Name")
or
Obj.Item(Position)
```

Parameters

The following parameter is used:

Parameter	Type	Default Value	Description
Index	Variant	-	Specifies the name or position of the object in the collection.

Return value

Blocks, ReadOnlyBlocks This method returns a Block object.

DataObjects, ReadOnlyDataObjects This method returns a DataObject object.

Libraries For custom libraries, this method returns a CustomLibraryFolder2 object. Otherwise, it returns a LibFolder object.

Projects This method returns a Project object.

PythonModules This method returns a PythonModule object.

Reports This method returns a Report object.

Results This method returns a Result object.

Selection This method returns an Element object.

Related objects

This method can be accessed by the following objects:

- Blocks on page 92
- DataObjects (Object) on page 101
- Libraries (Object) on page 110
- Libraries1 on page 111
- Libraries2 on page 112
- Libraries3 on page 113
- Projects (Object) on page 125
- Projects1 on page 126
- Projects2 on page 127

- Projects3 on page 129
- PythonModules on page 131
- Reports (Object) on page 139
- Results (Object) on page 142
- Selection (Object) on page 144

Related topics

References

- L -

Where to go from here

Information in this section

Load

Syntax

Projects

```
Obj.Load(FilePath)
```

Libraries

```
Obj.Load(FilePath, boolManuelUpdateConfirmation)
```

Purpose

Projects To load an AutomationDesk project from a file.

Libraries To load an AutomationDesk library from a file.

Description

Projects With the Load method, you can open an AutomationDesk project file. If the specified file is created with an earlier AutomationDesk version it is updated to the current version automatically.

Libraries With the Load method, you can open an AutomationDesk library file. Via the boolManuelUpdateConfirmation parameter, you can specify whether to display a dialog that lets you confirm or cancel the update of the library if it was created with an earlier AutomationDesk version. If the dialog is omitted, the file is updated automatically.

Parameters

Projects The following parameter is used for the Projects object:

Parameter	Type	Default Value	Description
FilePath	string	" "	Specifies the AutomationDesk project file (ADPX) or the legacy project file (ADP) to be loaded.

Libraries The following parameters are used for the Libraries object:

Parameter	Type	Default Value	Description
FilePath	string	" "	Specifies the AutomationDesk library file (ADLX) or the legacy library file (ADL) to be loaded.
boolManuelUpdateConfirmation	boolean	False	Specifies whether an update confirmation dialog is displayed when loading a file that was created with an earlier AutomationDesk version. If set to False, the update confirmation dialog is omitted and the file is updated automatically.

Return value

Projects This method returns a Project object.

Libraries This method returns a CustomLibraryFolder object.

Related objects

This method can be accessed by the following object:

- Projects (Object) on page 125
- Projects1 on page 126
- Projects2 on page 127
- Projects3 on page 129
- Libraries1 on page 111

- Libraries2 on page 112
- Libraries3 on page 113

Related topics

References

LoadEx

Syntax

```
Obj.LoadEx(FilePath, boolManuelUpdateConfirmation)
```

Purpose

To load a project with the option to suppress the update confirmation dialog.

Description

With the LoadEx method, you can open an AutomationDesk project file. Via the boolManuelUpdateConfirmation parameter, you can specify whether to display a dialog that lets you confirm or cancel the update of the project or library if it was created with an earlier AutomationDesk version. If the dialog is omitted, the file is updated automatically.

Parameters

The following parameters are used:

Parameter	Type	Default Value	Description
FilePath	string	" "	Specifies the AutomationDesk project file (ADPX) or the legacy project file (ADP) to be loaded.
boolManuelUpdateConfirmation	boolean	False	Specifies whether an update confirmation dialog is displayed when loading a file that was created with an earlier AutomationDesk version. If set to False, the update confirmation dialog is omitted and the file is updated automatically.

Return value

This method returns a Project object.

Related objects

This method can be accessed by the following object:

- Projects2 on page 127
- Projects3 on page 129

Related topics

References

- M -

Move

Syntax

```
Obj.Move(SourceObject, Position, ConfirmBreakLink, InsertBeforeObject, InsertAfterObject)
```

Purpose

To move a specified object at the specified position.

Description

The Move method moves a specific Block or DataObject object to a specified position. This lets you change the order of the objects at a hierarchy level. There are four ways of using the Move method:

1. You can position the object after the last object in the same hierarchy tree like this:

   `Obj.Move(SourceObject)`

2. To position the object at a specific position, specify the position like this:

   `Obj.Move(SourceObject, Position)`

3. To position the object before a specific object, specify the specific object like this:

   `Obj.Move(SourceObject, -1, ConfirmBreakLink, InsertBeforeObject)`

   The `Position` parameter must have the default value, otherwise the object will not be positioned before the specific object, but at the given position.

4. To position the object after a specific object, specify the specific object like this:

   `Obj.Move(SourceObject, -1, ConfirmBreakLink, None, InsertAfterObject)`

   The `Position` parameter must have the default value and the `InsertBeforeObject` parameter must be `None`, otherwise the object will not be positioned as required.

Parameters

The following parameters are used:

Parameter	Type	Default Value	Description
SourceObject	Block or DataObject	-	Specifies the Block or DataObject object to be moved.
Position	variant	-1	Specifies the position to move the object to.
ConfirmBreakLink	boolean	False	Specifies whether the link between the object and the library element should break.
InsertBeforeObject	variant	None	Specifies the successor of the moved object.
InsertAfterObject	variant	None	Specifies the predecessor of the object.

Return value

None

Related objects

This method can be accessed by the following objects:

- Blocks on page 92
- DataObjects (Object) on page 101

- O -

Open

Syntax

Projects, Libraries

```
Obj.Open(FileName, OptionsVal, ManualUpdateConfirmation)
```

MATLAB

```
Obj.Open()
```

Purpose

To open an AutomationDesk project, a library or a MATLAB instance.

Description

Projects, Libraries With the **`Open`** method, you can open an AutomationDesk project file or a library from a file.

You can decide whether to overwrite an existing project or library with the same name via the **`OptionsVal`** parameter.

If you use a newer AutomationDesk version to open a project file or library file, a dialog is displayed to confirm the update of the project or library. You can decide whether to display the dialog via the `ManualUpdateConfirmation` parameter.

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

Parameters

Projects, Libraries The following parameters are used:

Parameter	Type	Default Value	Description
FileName	string	" "	Specifies the file to be opened. For projects, files of the following formats are supported: ▪ ADPX files that contain an AutomationDesk project that is saved in XML format using AutomationDesk 6.2 or later. ▪ ADP files that contain an AutomationDesk project that is saved in a binary legacy format using AutomationDesk 6.1 or earlier. ▪ ZIP files that contain an AutomationDesk project that is exported as a compressed archive. ▪ APX files that contain an AutomationDesk project that is exported in legacy XML format using AutomationDesk 6.0 or earlier. ▪ ADPX files that contain an AutomationDesk project that is exported in XML format using AutomationDesk 6.1 or later. For libraries, files of the following formats are supported: ▪ ADLX files that contain a custom library that is saved in XML format using AutomationDesk 6.2 or later. ▪ ADL files that contain a custom library that is saved in a binary legacy format using AutomationDesk 6.1 or earlier. ▪ ZIP files that contain a custom library that is exported as a compressed archive. ▪ ALX files that contain a custom library that is exported in legacy XML format using AutomationDesk 6.0 or earlier. ▪ ADLX files that contain a custom library that is exported in XML format using AutomationDesk 6.1 or later.
OptionsVal	enumeration	adCancel	Specifies whether to overwrite an existing project or library. The following values of the FileOptions enumeration are supported: ▪ adCancel (=0) to abort the operation ▪ adOverWrite (=1) to overwrite a project or library
ManualUpdateConfirmation	boolean	False	Specifies whether an update confirmation dialog is displayed when loading a project with a newer AutomationDesk

Parameter	Type	Default Value	Description
			version. If set to False (0), the update confirmation dialog is ignored.

MATLAB None

Return value

None

Related objects

This method can be accessed by the following object:

- Projects3 on page 129
- Libraries3 on page 113
- MATLAB on page 196

Related topics

References

- Q -

Quit

Syntax

```
Obj.Quit()
```

Purpose

To close AutomationDesk.

Description

If you use AutomationDesk via the API, you should close AutomationDesk by using this method. If you close AutomationDesk interactively, the Application object becomes invalid.

If you call this method for the application of the AutomationDesk - Automation Server, an exception is raised.

Parameters	None
Return value	None
Related objects	This method can be accessed by the following object: ▪ Application1 on page 88 ▪ Application2 on page 89

- R -

Where to go from here

Information in this section

Remove

Syntax

Blocks, DataObjects

```
Obj.Remove(Index, ConfirmBreakLink)
```

Results, Reports

```
Obj.Remove(Index)
```

Purpose

To delete an object from the project structure.

Description

The Remove method removes the specified Block, DataObject, Result or Report object with its child elements from the project.

Block and DataObject objects can also be removed from the custom library. You cannot remove Block or DataObject objects from the built-in libraries.

Parameters

Blocks, DataObjects The following parameters are used for Blocks and DataObjects objects:

Parameter	Type	Default Value	Description
Index	variant	-	Specifies the name or position of the specific Block or DataObject object in the collection to be deleted.
ConfirmBreakLink	boolean	False	Specifies whether the link between the object and the custom library object should break. The link to the object's parent object will also be broken.

Results, Reports The following parameters are used for Results and Reports objects:

Parameter	Type	Default Value	Description
Index	variant	-	Specifies the name or position of the specific Result or Report object in the collection to be deleted.

Return value

None

Related objects

This method can be accessed by the following objects:

- Blocks on page 92
- DataObjects (Object) on page 101
- Results (Object) on page 142
- Reports (Object) on page 139

RemoveAll

Syntax

Blocks, DataObjects

```
Obj.RemoveAll(ConfirmBreakLink)
```

Results, Reports

```
Obj.RemoveAll()
```

Purpose

To delete all the created child elements of a collection.

Description

The RemoveAll method removes all the instantiated blocks and data objects of a collection from the project or custom library. You cannot remove blocks and data objects from the built-in libraries.

You can delete all the results of the parent object. This also deletes all the reports generated for these results too.

You can delete all the reports that you generated for the result. It does not delete all reports from your project.

Parameters

Blocks, DataObjects The following parameters are used for Blocks and DataObjects objects:

Parameter	Type	Default Value	Description
Index	variant	-	Specifies the name or position of the specific Block or DataObject object in the collection to be deleted.
ConfirmBreakLink	boolean	False	Specifies whether the link between the object and the custom library object should break. The link to the object's parent object will also be broken.

Results, Reports The following parameters are used for Results and Reports objects:

Parameter	Type	Default Value	Description
Index	variant	-	Specifies the name or position of the specific Result or Report object in the collection to be deleted.

Return value

None

Related objects

This method can be accessed by the following objects:

- Blocks on page 92
- DataObjects (Object) on page 101
- Results (Object) on page 142
- Reports (Object) on page 139

RootElement.Add

Syntax

Dictionary

```
Obj.Add(Key, Value)
or
Obj.Add(Key, Subitem)
```

List

```
Obj.Add(Value)
or
Obj.Add(Subitem)
```

Purpose

To add an item to a RootElement object.

Description

Dictionary You can add single key-value pairs to the dictionary and subitems to it. A subitem can consists of another dictionary, a list or a tuple. To create a subitem, refer to CreateSubItem on page 428.

List You can add single items to the list and subitems to it. A subitem can consists of another list, a tuple or a dictionary. To create a subitem, refer to CreateSubItem on page 428. The new item is added at the end of the list.

This method is available via the RootElement object of the related object. You get the RootElement object by using the object's RootElement property.

Parameters

Dictionary The following parameters are used for a Dictionary object:

Parameter	Type	Default Value	Description
Key	variant	None	Specifies the key of the new item.
Value	variant	None	Specifies the value of the new item.
Subitem	Dictionary, List or Tuple	None	Specifies an item of the dictionary as subitem instead of a single key-value pair.

List The following parameters are used for a List object:

Parameter	Type	Default Value	Description
Value	variant	None	Specifies the value of the new item.
Subitem	Dictionary, List or Tuple	None	Specifies an item of the list as subitem instead of a single value.

Return value

None

Related objects

This method can be accessed by the following objects:

- Dictionary on page 163
- List on page 180

Related topics

References

RootElement.. 385

RootElement.Clear

Syntax

```
Obj.Clear()
```

Purpose

To clear the contents of the RootElement object.

Description

This method is available via the RootElement object of the related object. You get the RootElement object by using the object's RootElement property.

Parameters

None

Return value

None

Related objects

This method can be accessed by the following objects:

- Dictionary on page 163
- List on page 180

Related topics

References

RootElement.Contains

Syntax

```
Obj.Contains(Key)
```

Purpose

To check whether the specified key is available in the dictionary-based RootElement object.

Description

This method is available via the RootElement object of the related object. You get the RootElement object by using the object's RootElement property.

Parameters

The following parameter is used:

Parameter	Type	Default Value	Description
Key	variant	None	Specifies the key to be checked.

Return value

This method returns a Boolean value:

- 0 - The specified key is not available in the RootElement object.
- 1 - The specified key is available in the RootElement object.

Related objects

This method can be accessed by the following object:

- Dictionary on page 163

Related topics

References

RootElement.GetItem

Syntax

Dictionary

```
Obj.GetItem(Key)
```

List, Tuple

```
Obj.GetItem(Index)
```

Purpose

To get an item of the RootElement object.

Description

This method is available via the RootElement object of the related object. You get the RootElement object by using the object's RootElement property.

Dictionary An item of a Dictionary consists on a key-value pair. By requesting a key, you can get the related value.

List, Tuple An item of a list or tuple consists of a value and an internally assigned index. By requesting an index, you can get the related value.

Parameters

Dictionary The following parameter is used for Dictionary objects:

Parameter	Type	Default Value	Description
Key	variant	None	Specifies the key of the item you want to get.

List, Tuple The following parameter is used for List and Tuple objects:

Parameter	Type	Default Value	Description
Index	long	None	Specifies the index of the item you want to get.

Return value

Dictionary This method returns the value of the key-value pair as a variant value.

List, Tuple This method returns the value of the specified index as a variant value.

Related objects

This method can be accessed by the following objects:

- Dictionary on page 163
- List on page 180
- Tuple on page 187

Related topics

References

RootElement 385

RootElement.IndexOf

Syntax

```
Obj.IndexOf(Value)
```

Purpose

To get the first index of the specified value in the RootElement object.

Description

An item of a list or tuple consists of a value and an internally assigned index. You can use this method to get the first position (index) in the list where the specified value is found.

This method is available via the RootElement object of the related object. You get the RootElement object by using the object's RootElement property.

Parameters

The following parameters are used:

Parameter	Type	Default Value	Description
Value	variant	None	Specifies the value of the item you want to get.

Return value

This method returns the index as a long value.

Related objects

This method can be accessed by the following objects:

- List on page 180
- Tuple on page 187

Related topics

References

RootElement 385

RootElement.Insert

Syntax

```
Obj.Insert(Index, NewValue)
```

Purpose

To add an item in the List.RootElement object at a specific position.

Description

This method is available via the RootElement object of the related object. You get the RootElement object by using the object's RootElement property.

Parameters

The following parameters are used:

Parameter	Type	Default Value	Description
Index	long	None	Specifies the position in the list where the value is to be inserted. The index value must be in the range 0 ... `count-1`.
NewValue	variant	None	Specifies the new value to be set.

Return value

None

Related objects

This method can be accessed by the following object:

- List on page 180

Related topics

References

RootElement.Remove

Syntax

Dictionary

```
Obj.Remove(Key)
```

List

```
Obj.Remove(Value)
```

Purpose

To remove the specified item from the RootElement object.

Description

This method is available via the RootElement object of the related object. You get the RootElement object by using the object's RootElement property.

Dictionary The item to be deleted is specified by its key.

List The first item in the list that has the specified value is removed.

Parameters

Dictionary The following parameter is used for Dictionary objects:

Parameter	Type	Default Value	Description
Key	variant	None	Specifies the key of the item to be removed.

List The following parameter is used for a List object:

Parameter	Type	Default Value	Description
Value	variant	None	Specifies the value of the item to be removed.

Return value

None

Related objects

This method can be accessed by the following objects:

- Dictionary on page 163
- List on page 180

Related topics

References

RootElement 385

RootElement.RemoveAt

Syntax

```
Obj.RemoveAt(Index)
```

Purpose

To remove an item from the given position in the List.RootElement object.

Description

The item of the specified index is removed from the list.

This method is available via the RootElement object of the related List object. You get the RootElement object by using the List's RootElement property.

Parameters

The following parameters are used:

Parameter	Type	Default Value	Description
Index	long	None	Specifies the index of the item you want to remove.

Return value

None

Related objects

This method can be accessed by the following object:
- List on page 180

Related topics

References

RootElement.SetItem

Syntax

Dictionary

```
Obj.SetItem(Key, Value)
```

List

```
Obj.SetItem(Index, Item)
```

Purpose

To edit the value of an item of a RootElement object.

Description

This method is available via the RootElement object of the related object. You get the RootElement object by using the object's RootElement property.

Parameters

Dictionary The following parameters are used for Dictionary objects:

Parameter	Type	Default Value	Description
Key	variant	None	Specifies the key of the item to be modified.
Value	variant	None	Specifies the new value to be set.

List The following parameters are used for List objects:

Parameter	Type	Default Value	Description
Index	long	None	Specifies the position of the item to be modified.
Item	variant	None	Specifies the new value to be set.

Return value

None

Related objects

This method can be accessed by the following objects:
- Dictionary on page 163
- List on page 180

Related topics

References

RootElement.. 385

- S -

Where to go from here

Information in this section

Save

Syntax	`Obj.Save()`
Purpose	To save a project or a custom library folder.
Description	**Project** You can use the Save method to save a project with all its subelements in its original path. **CustomLibraryFolder** The Save method saves the custom library elements. Each newly created library element below the custom library node has to be stored to make it available for further AutomationDesk sessions. When the custom library is saved, all custom library folders and all included custom library elements are saved. If it is not saved, a new element is inserted temporarily and can be used only in the current session. Any changes to the custom library have to be saved explicitly.
Parameters	None
Return value	None
Related objects	This method can be accessed by the following objects: ▪ CustomLibraryFolder on page 94 ▪ Project on page 121

SaveAll

Syntax	`Obj.SaveAll()`
Purpose	To save all opened projects and custom libraries.
Description	With the SaveAll method, you can save all the AutomationDesk projects/custom libraries of a collection.
Parameters	None
Return value	None
Related objects	This method can be accessed by the following objects: ▪ Projects (Object) on page 125 ▪ Projects1 on page 126 ▪ Projects2 on page 127 ▪ Projects3 on page 129 ▪ Libraries (Object) on page 110 ▪ Libraries1 on page 111 ▪ Libraries2 on page 112 ▪ Libraries3 on page 113

SaveAs

Syntax	`Obj.SaveAs(NewFileName, OptionsVal)`
Purpose	To save the project with a new file name.
Description	The SaveAs method allows you to enter a file name. The project is saved under the file name and path you specified. The project name is changed accordingly.

Parameters

The following parameters are used:

Parameter	Type	Default Value	Description
NewFileName	string	" "	Specifies the new file the project is saved to.
OptionsVal	enumeration	-	Specifies whether an existing project should be overwritten. The possible values of the FileOptions enumeration are adOverWrite (=1) to overwrite the project and adCancel (=0) to abort the operation, if a project with same name exists.

If you want to use the predefined constants, you must make some preparations beforehand. For further information, refer to Using API Constants on page 67.

Return value

None

Related objects

This method can be accessed by the following object:

- CustomLibraryFolder1 on page 95
- Project on page 121

Related topics

Basics

Select

Syntax

```
Obj.Select()
```

Purpose

To select a diagnostic or calibration project from the connected system.

Description

After the system is connected and you have configured the diagnostic or calibration project, you must use this method to select it. The selection must be established before you can configure the VehicleInformation objects of your diagnostic project or the LogicalLink objects of your calibration project.

Parameters

None

Return value

This method returns a Boolean value:
- 0: The method has failed.
- 1: The method has passed.

Related objects

This method can be accessed by the following objects:
- D3Project on page 211
- MC3Project on page 201

Related topics

References

SetParameterValue

Syntax

```
Obj.SetParameterValue(ParameterName, ParameterValue, ParameterUnit)
```

Purpose

To set the parameter value of the specified Parameter object.

Description

With this method, you can modify the default values of a parameter.

Parameters

The following parameters are used:

Parameter	Type	Default Value	Description
ParameterName	string	" "	Specifies the name of the parameter you want to modify.
ParameterValue	variant	None	Specifies the value of the parameter.
ParameterUnit	string	" "	Specifies the unit of the value.

Return value

None

Related objects

This method can be accessed by the following objects:
- D3ControlPrimitive on page 215
- D3Service on page 217
- D3SingleJob on page 218

SetValue

Syntax

```
Obj.SetValue(XValues, FcnValues)
```

Purpose

To replace the signal's values.

Description

With this method, you can set values of the time vector and the function value vector of the Signal data object. Both vectors must be of the same length.

Parameters

The following parameters are used:

Parameter	Type	Default Value	Description
XValues	variant	None	Specifies the vector of time values of the signal.
FcnValues	variant	None	Specifies the vector of function values of the signal.

Return value

None

Related objects

This property can be accessed by the following object:
- Signal on page 152

Shutdown

Syntax

```
Obj.Shutdown()
```

Purpose	To shut down the XIL API framework.
Description	This method lets you shut down the configured XIL API framework.
Parameters	None
Return value	None
Related objects	This method can be accessed by the following object: ▪ Framework (Object) on page 108

StopExecution

Syntax

```
Obj.StopExecution()
```

Purpose

To automatically stop a running execution.

Description

StopExecution is a method of the ExecutionConfiguration1 object. The ExecutionConfiguration object can be accessed from `ApplicationObj.Options.Execution`. If a COM wrapper is used in the script, the object returned from the `ApplicationObj.Options.Execution` has to be typecasted as `ExecutionConfiguration1` for accessing the StopExecution method.

Note

Immediately after a StopExecution call, cleanup activties might lead to a delayed switch of the IsExecutionRunning property.

Parameters

None

Return value

None

Related objects

This method can be accessed by the following objects:

- ExecutionConfiguration1 on page 103
- ExecutionConfiguration2 on page 104

Synchronize

Syntax

```
Obj.Synchronize()
```

Purpose

To synchronize the sequence(s) with the custom library templates.

Description

The Synchronize method synchronizes instantiated custom sequences with the corresponding templates in the custom library while their library links exist. If the library element is modified, the instantiated block is not automatically updated. With the synchronizing operation, the block is updated to the library element. Data objects that have been added to the library element are added to the block. Data objects that have been removed from the library element are removed from the block. Data objects whose type has been changed are replaced. If subsystems have been added to or removed from the library element, they are also added to or removed from the block. For further information, refer to Working with Custom Libraries (AutomationDesk Basic Practices 📖).

Parameters

None

Return value

None

Related objects

This method can be accessed by the following objects:

- CustomLibraryFolder on page 94
- CustomLibraryFolder1 on page 95
- Folder on page 105
- Folder1 on page 107
- Project on page 121
- Project1 on page 123
- Sequence on page 145

- W -

Where to go from here

Information in this section

WriteError

Syntax

```
Obj.WriteError(Message)
```

Purpose

To write an error message to the logs.

Description

To simultaneously write an error message to the Message Viewer and to the dSPACE log file.

Parameters

The following parameters are used:

Parameter	Type	Default Value	Description
Message	string	" "	Specifies the message text to be logged.

Return value

None

Related objects

This method can be accessed by the following object:

- Log (Object) on page 118

WriteInformation

Syntax

`Obj.WriteInformation(Message)`

Purpose

To write an informational message to the logs.

Description

To simultaneously write an informational message to the Message Viewer and to the dSPACE log file.

Parameters

The following parameters are used:

Parameter	Type	Default Value	Description
Message	string	`" "`	Specifies the message text to be logged.

Return value

None

Related objects

This method can be accessed by the following object:

- Log (Object) on page 118

WriteMessage

Syntax

`Obj.WriteMessage(Severity, Message)`

Purpose

To write a message of a specified severity to the log.

Description

To simultaneously write a message of a specified severity to the Message Viewer and to the dSPACE log file.

Parameters

The following parameters are used:

Parameter	Type	Default Value	Description
Severity	enumeration	Error	Specifies the severity of the logged message. The possible values are defined by the LogMessageSeverity enumeration. Refer to Overview of API Constants on page 24.
Message	string	" "	Specifies the message text to be logged.

If you want to use the predefined constants, you must make some preparations beforehand. For more information, refer to Using API Constants on page 67.

Return value

None

Related objects

This method can be accessed by the following object:

- Log (Object) on page 118

Related topics

Basics

WriteWarning

Syntax

```
Obj.WriteWarning(Message)
```

Purpose

To write a warning message to the logs.

Description

To write a warning message to the Message Viewer and to the dSPACE log file.

Parameters

The following parameters are used:

Parameter	Type	Default Value	Description
Message	string	" "	Specifies the message text to be logged.

Return value	None
Related objects	This method can be accessed by the following object: ▪ Log (Object) on page 118

Events in Alphabetical Order

Introduction

Some of the COM objects of the AutomationDesk COM API provide specific events. The following list shows you all the available events. In their descriptions you find the COM objects they are supported by.

Where to go from here

Information in this section

OnAdd

Syntax

```
OnAdd(DispatchObject, Position)
```

Purpose

To react to a specific element being created.

Description

An event is a method that an object invokes to inform the client application of a state or value modification. You can define a subroutine that specifies the client application's reaction to each event.

The OnAdd event occurs when:

- A folder or sequence is created.
  The event can be accessed by a Blocks object.
- A data object is created.
  The event can be accessed by a DataObjects object.
- A result is created.
  The event can be accessed by a Results object.
- A report is created.
  The event can be accessed by a Reports object.

Parameters

The following parameters are used:

Parameter	Type	Default Value	Description
DispatchObject	Dispatch	-	Contains the received Dispatch object from the server (sequence, folder, data object, result, report) that the client can react to.

Parameter	Type	Default Value	Description
Position	long	-	Contains the position in the project structure where the element is added.

Related objects

This event can be accessed by the following objects:
- Blocks on page 92
- DataObjects (Object) on page 101
- Results (Object) on page 142
- Reports (Object) on page 139

OnError

Syntax

```
OnError(ErrorString)
```

Purpose

To react to an error in the application.

Description

An event is a method that an object invokes to inform the client application of a state or value modification. You can define a subroutine that specifies the client application's reaction to each event.

The OnError event occurs if an error occurs in your application.

Parameters

The following parameter is used:

Parameter	Type	Default Value	Description
ErrorString	string	" "	Contains the received error string from the server that the client can react to.

Related objects

This event can be accessed by the following object:
- Application on page 87
- Application1 on page 88
- Application2 on page 89

OnExecutionFinished

Syntax	`OnExecutionFinished()`
Purpose	To react to an execution finishing.
Description	An event is a method that an object invokes to inform the client application of a state or value modification. You can define a subroutine that specifies the client application's reaction to each event. The OnExecutionFinished event occurs after execution has finished.
Parameters	None
Related objects	This event can be accessed by the following objects: ▪ Project on page 121 ▪ Folder on page 105 ▪ Sequence on page 145

OnExecutionProgress

Syntax	`OnExecutionProgress(ProgressValue)`
Purpose	To react to the progress of an execution.
Description	An event is a method that an object invokes to inform the client application of a state or value modification. You can define a subroutine that specifies the client application's reaction to each event. The OnExecutionProgress event occurs during the progress of an execution.
Parameters	The following parameter is used:

Parameter	Type	Default Value	Description
ProgressValue	int	-	Contains the percentage of the execution's progress received from the server.

Related objects	This event can be accessed by the following objects: ▪ Project on page 121 ▪ Folder on page 105 ▪ Sequence on page 145

OnExecutionStarted

Syntax	`OnExecutionStarted()`
Purpose	To react to an execution starting.
Description	An event is a method that an object invokes to inform the client application of a state or value modification. You can define a subroutine that specifies the client application's reaction to each event. The OnExecutionStarted event occurs after an execution has started.
Parameters	None
Related objects	This event can be accessed by the following objects: ▪ Project on page 121 ▪ Folder on page 105 ▪ Sequence on page 145

OnModified

Syntax	`OnModified(AttributeName, NewValue)`
Purpose	To react to a property being modified.
Description	An event is a method that an object invokes to inform the client application of a state or value modification. You can define a subroutine that specifies the client application's reaction to each event.

The OnModified event occurs when one of the following properties of an object has been changed:

- Name
- ResultLevel

Parameters

The following parameters are used:

Parameter	Type	Default Value	Description
AttributeName	string	" "	Contains the name of the modified property.
NewValue	variant	-	Contains the value the object's property is modified with.

Related objects

This event can be accessed by the following objects:

- Project on page 121
- Folder on page 105
- PythonModule on page 130
- PythonPackage on page 132
- Sequence on page 145
- Result on page 140
- Report on page 135
- Any supported data object

OnPathChanged

Syntax

```
OnPathChanged(NewValue)
```

Purpose

To react to the path of a File data object being modified.

Description

An event is a method that an object invokes to inform the client application of a state or value modification. You can define a subroutine that specifies the client application's reaction to each event.

The OnPathChanged event occurs when the path of a File data object is modified.

Parameters

The following parameter is used:

Parameter	Type	Default Value	Description
NewValue	string	" "	Contains the modified path that the client can react to.

Related objects

This event can be accessed by the following object:

- File on page 167

OnProjectActivate

Syntax

```
OnProjectActivate(ActivatedProject)
```

Purpose

To react to project activation.

Description

An event is a method that an object invokes to inform the client application of a state or value modification. You can define a subroutine that specifies the client application's reaction to each event.

The OnProjectActivate event occurs when a project is activated.

Parameters

The following parameter is used:

Parameter	Type	Default Value	Description
ActivatedProject	Project	-	Contains the received Project object from the server that the client can react to.

Related objects

This event can be accessed by the following object:

- Application on page 87
- Application1 on page 88
- Application2 on page 89

OnProjectClose

Syntax

```
OnProjectClose(CloseProject, Cancel)
```

Purpose

To react to a project being closed.

Description

An event is a method that an object invokes to inform the client application of a state or value modification. You can define a subroutine that specifies the client application's reaction to each event.

The OnProjectClose event occurs when a project is closed.

Parameters

The following parameters are used:

Parameter	Type	Default Value	Description
CloseProject	Project	-	Contains the received Project object from the server that the client can react to.
Cancel	boolean	-	Defines the behavior of the close operation for a modified project. The client can cancel the close operation by setting the Cancel parameter to `True`. If the Cancel parameter is set to `False`, the project is closed.

Related objects

This event can be accessed by the following object:

- Application on page 87
- Application1 on page 88
- Application2 on page 89

Related topics

References

OnProjectClosed....................488

OnProjectClosed

Syntax

```
OnProjectClosed()
```

Purpose

To react to a closed project.

Description

An event is a method that an object invokes to inform the client application of a state or value modification. You can define a subroutine that specifies the client application's reaction to each event.

The OnProjectClosed event occurs after a project is closed.

Parameters

None

Related objects

This event can be accessed by the following object:

- Application on page 87
- Application1 on page 88
- Application2 on page 89

Related topics

References

OnProjectClose..........488

OnProjectCreate

Syntax

```
OnProjectCreate(ProjectName, Cancel)
```

Purpose

To react to project creation.

Description

An event is a method that an object invokes to inform the client application of a state or value modification. You can define a subroutine that specifies the client application's reaction to each event.

The OnProjectCreate event occurs when a project is created.

Parameters

The following parameters are used:

Parameter	Type	Default Value	Description
ProjectName	string	" "	Contains the received name of the project that the client can react to.
Cancel	boolean	-	Defines the behavior of the create operation when the specified project already exists. The client can cancel the create operation by setting the Cancel parameter to `True`. If the Cancel parameter is set to `False`, the project is created.

Related objects

This event can be accessed by the following object:

- Application on page 87
- Application1 on page 88
- Application2 on page 89

Related topics

References

OnProjectCreated

Syntax

```
OnProjectCreated(NewProject)
```

Purpose

To react to a created project.

Description

An event is a method that an object invokes to inform the client application of a state or value modification. You can define a subroutine that specifies the client application's reaction to each event.

The OnProjectCreated event occurs after a project is created.

Parameters

The following parameter is used:

Parameter	Type	Default Value	Description
NewProject	Project	-	Contains the received Project object that the client can react to.

Related objects

This event can be accessed by the following object:

- Application on page 87
- Application1 on page 88
- Application2 on page 89

Related topics

References

OnProjectOpen

Syntax

```
OnProjectOpen(ProjectName, Cancel)
```

Purpose

To react to a project being opened.

Description

An event is a method that an object invokes to inform the client application of a state or value modification. You can define a subroutine that specifies the client application's reaction to each event.

The OnProjectOpen event occurs when a project is opened.

Parameters

The following parameters are used:

Parameter	Type	Default Value	Description
ProjectName	string	" "	Contains the received Project object from the server that the client can react to.
Cancel	boolean	-	Defines the behavior of the open operation when the specified project is already opened. The client can cancel the open operation by setting the Cancel parameter to `True`. If the Cancel parameter is set to `False`, the project is opened.

Related objects

This event can be accessed by the following object:

- Application on page 87
- Application1 on page 88
- Application2 on page 89

Related topics

References

OnProjectOpened...491

OnProjectOpened

Syntax

```
OnProjectOpened(OpenedProject)
```

Purpose

To react to an opened project.

Description

An event is a method that an object invokes to inform the client application of a state or value modification. You can define a subroutine that specifies the client application's reaction to each event.

The OnProjectOpened event occurs after a project was opened.

Parameters

The following parameter is used:

Parameter	Type	Default Value	Description
OpenedProject	Project	-	Contains the received Project object from the server that the client can react to.

Related objects

This event can be accessed by the following object:

- Application on page 87
- Application1 on page 88
- Application2 on page 89

Related topics

References

OnProjectOpen..491

OnProjectSave

Syntax

```
OnProjectSave(SaveProject, ProjectFile)
```

Purpose

To react to a project being saved.

Description

An event is a method that an object invokes to inform the client application of a state or value modification. You can define a subroutine that specifies the client application's reaction to each event.

The OnProjectSave event occurs when a project is saved.

Parameters

The following parameters are used:

Parameter	Type	Default Value	Description
SaveProject	Project	-	Contains the received Project object from the server that the client can react to.
ProjectFile	string	" "	Contains the file the project is saved to.

Related objects

This event can be accessed by the following object:
- Application on page 87
- Application1 on page 88
- Application2 on page 89

Related topics

References

OnProjectSaved

Syntax

```
OnProjectSaved(SavedProject, ProjectFile)
```

Purpose

To react to a project being saved.

Description

An event is a method that an object invokes to inform the client application of a state or value modification. You can define a subroutine that specifies the client application's reaction to each event.

The OnProjectSaved event occurs after a project was saved.

Parameters

The following parameters are used:

Parameter	Type	Default Value	Description
SavedProject	Project	-	Contains the received Project object from the server that the client can react to.
ProjectFile	string	" "	Contains the file the project is saved to.

Related objects	This event can be accessed by the following object: ▪ Application on page 87 ▪ Application1 on page 88 ▪ Application2 on page 89

Related topics

References

OnRemove

Syntax

```
OnRemove(DispatchObject)
```

Purpose

To react to a specific element being deleted.

Description

An event is a method that an object invokes to inform the client application of a state or value modification. You can define a subroutine that specifies the client application's reaction to each event.

The OnRemove event occurs when:

- A folder or a sequence is deleted.

  The event can be accessed by a Blocks object.
- A data object is deleted.

  The event can be accessed by a DataObjects object.
- A result is deleted.

  The event can be accessed by a Results object.
- A report is deleted.

  The event can be accessed by a Reports object.

Note

Do not call a method or property for the removed object.

Parameters

The following parameter is used:

Parameter	Type	Default Value	Description
DispatchObject	Dispatch	-	Contains the received Dispatch object (folder, sequence, data object, result, report) from the server that the client can react to.

Related objects

This event can be accessed by the following objects:

- Blocks on page 92
- DataObjects (Object) on page 101
- Results (Object) on page 142
- Reports (Object) on page 139

OnShouldExecutionBeStopped

Syntax

```
OnShouldExecutionBeStopped()
```

Purpose

To react to an execution stopping.

Description

An event is a method that an object invokes to inform the client application of a state or value modification. You can define a subroutine that specifies the client application's reaction to each event.

This event is triggered before executing each block. If this method is implemented, the client application can determine whether to stop the execution by calling the Stop method of the ExecutionConfiguration.

Parameters

None

Related objects

This event can be accessed by the following objects:

- Project on page 121
- Project1 on page 123
- Folder on page 105
- Folder1 on page 107
- Sequence on page 145
- Sequence1 on page 147

OnValueChanged

Syntax

```
OnValueChanged(Value)
```

Purpose

To react to a value being changed.

Description

An event is a method that an object invokes to inform the client application of a state or value modification. You can define a subroutine that specifies the client application's reaction to each event.

The OnValueChanged event occurs when a value of the object is changed.

Parameters

The following parameter is used:

Parameter	Type	Default Value	Description
Value	variant	-	Contains the changed value that the client can react to.

Related objects

This event can be accessed by the following objects:

- Float on page 171
- Int on page 174
- String on page 184

OnWrite

Syntax

```
OnWrite(OutputString)
```

Purpose

To react to an output by the application.

Description

An event is a method that an object invokes to inform the client application of a state or value modification. You can define a subroutine that specifies the client application's reaction to each event.

The OnWrite event occurs when your application generates an output, for example, using the Python print command. In your event subroutine, you can specify the OutputString parameter, which can be received by your client application.

Parameters

The following parameter is used:

Parameter	Type	Default Value	Description
OutputString	string	" "	Contains the received output string from the server that the client can react to.

Related objects

This event can be accessed by the following object:

- Application on page 87
- Application1 on page 88
- Application2 on page 89

Limitations

Limitations When Using the AutomationDesk API

Timing problem with COM

If you execute the same operation, for example, creating an object or renaming an object, thousands of times in a loop, the Windows message queue for COM communication can overflow. To avoid this, you must add the `pythoncom.PumpWaitingMessages()` function to your script. This guarantees that the COM-based messages get enough time to be handled.

Dispatch call might fail when using Windows 10

If you use a Python script to remote-control the dSPACE software via its COM API and you use Windows 10 as the operating system, you have to start the dSPACE software and PythonWin or Python.exe in the same way. Otherwise, the dispatch call is not executed correctly.

For example:

- If you first start AutomationDesk via its Desktop shortcut or via double-click on `AutomationDesk.exe`, and then PythonWin or Python.exe via `Run as administrator` in the context menu, the `Dispatch` call in the API script exits with the error message: *Server execution failed*.
- If you first start AutomationDesk via `Run as administrator` in its context menu, and then PythonWin or Python.exe via Desktop shortcut or double-click, the `Dispatch` call in the API script is executed, but a second instance is opened, which might lead to unpredicted behavior.

Using blocks of the Dialogs library

AutomationDesk projects containing blocks of the Dialogs library cannot be executed via DCOM. The execution hangs if it is started via DCOM.

Features not provided

It has the following limitations compared with the AutomationDesk features:

- The features of the Sequence Builder are not provided by the API:
  - You cannot build sequences with automation blocks from the AutomationDesk library.

  - You cannot parameterize data objects of the automation blocks used in a sequence.
  - You cannot build custom blocks for the custom library.
- You cannot access all the information of a result directly via API objects, only the result states. Information on the control flow is not available at all. To make the execution results available, you must generate a report.

Restrictions on using Int objects

The integer value used with the COM API is restricted to 32 bits (long data type). In AutomationDesk, an Int data object is represented by a Python integer with unlimited precision.

- Use Float objects instead of Int objects in your API script if you are not sure about the required data range.

Restrictions using a Variant object with collection data types

- Assigning a tuple to a variant

  The recommended method to assign the values of a tuple object to a variant is:

  ```
  VariantObject.Value = TupleObject.RootElement
  ```

  , the data type of the COM object is used. Other methods, such as,

  `VariantObject.Value = TupleObject.RootElement.Value`

  or the direct assignment

  `VariantObject.Value = (Value1, Value2, Value3)`

  retrieve the same result, but the data type is specified by Python.
- Assigning a list to a variant

  The recommended method to assign the values of a list object to a variant is:

  ```
  VariantObject.Value = ListObject.RootElement
  ```

  This method uses the data type of the COM object. Other methods, such as,

  `VariantObject.Value = ListObject.RootElement.Value`

  or the direct assignment

  `VariantObject.Value = [Value1, Value2, Value3]`

  retrieve the values as a tuple.
- Assigning a dictionary to a variant

  The recommended method to assign the values of a dictionary object to a variant is:

  ```
  VariantObject.Value = DictionaryObject.RootElement
  ```

  This method uses the data type of the COM object. The assignment via Value property

  `VariantObject.Value = DictionaryObject.RootElement.Value`

  is also correct. The direct assignment is not possible.

Restrictions for assigning lists as subitems to a List object

You cannot assign lists as subitems directly to a List object. You must use the Add method of the List.RootElement object.

For example, you have specified two lists as

`SubList1.Value = [1,2,3]` and

`SubList2.Value = [3,4,5]`

To add these lists to a List object as subitems, you must use the Add method:

```
ListObject.RootElement.Add(SubList1)
ListObject.RootElement.Add(SubList2)
```

It is not possible to use the Value property: `ListObject.RootElement.Value = [SubList1.Value, SubList2.Value]`

Glossary

Introduction

The glossary briefly explains the most important expressions and naming conventions used in the AutomationDesk documentation.

Where to go from here

Information in this section

Symbols

@ADLX folder A file system folder with the name `<CustomLibraryName>@adlx` that stores information on the elements in the related custom library, such as template files, attached Python modules, or packages.

In the Library Bowser, you always use this folder in combination with the related library file (ADLX).

@ADPX folder A file system folder with the name `<ProjectName>@adpx` that stores information on the elements in the related project, such as sequence files, attached files, results, and reports.

In the Project Manager, you always use this folder in combination with the related project file (ADPX).

@BLKX folder A file system folder with the name `<ElementName>@blkx` that stores information on the subelements in an exported element, such as sequence files, attached files, Python modules, or packages.

You always use this folder in combination with the related parent element file (BLKX).

A

ADL file An AutomationDesk legacy library file that contains the specification of a custom library which was saved to the file system.

These files were created using AutomationDesk 6.1 or earlier. You can open and migrate them to library XML files (ADLX).

ADL.ZIP file An AutomationDesk legacy archive file that contains the specification of a custom library which was managed under version control.

These files were created using AutomationDesk 6.1 or earlier and must be migrated to let you continue to work under version control.

ADLX file An AutomationDesk library XML file that contains the specification of a saved or exported custom library. The ADLX file is located in the same folder as the @ADLX folder.

You can open, edit, save, import, and export ADLX files via the Library Bowser.

ADO file An AutomationDesk display options file that contains information on how a project or library is displayed when it is opened in the AutomationDesk user interface. This includes bookmarks, breakpoints, and the collapse state of folders and blocks.

These files are created when a project or library is saved or closed.

ADP file An AutomationDesk legacy project file that contains a project's specification, its results, and its reports.

These files were created using AutomationDesk 6.1 or earlier. You can open and migrate them to project XML files (ADPX).

ADP.ZIP file An AutomationDesk legacy archive file that contains the specification of a project which was managed under version control.

These files were created using AutomationDesk 6.1 or earlier and must be migrated to let you continue to work under version control.

ADPX file An AutomationDesk project XML file that contains the specification of a saved or exported AutomationDesk project. The ADPX file is located in the same folder as the @ADPX folder.

You can open, edit, save, import, and export ADPX files via the Project Manager.

ALX file An AutomationDesk library legacy XML file that contains the specification of an exported custom library.

These files were created using AutomationDesk 6.0 or earlier. You can import ALX files in the Library Bowser.

APX file An AutomationDesk project legacy XML file that contains the specification of an exported AutomationDesk project.

These files were created using AutomationDesk 6.0 or earlier. You can import APX files in the Project Manager.

ASAM AE XIL API An API standard for the communication between test automation tools, such as AutomationDesk, and test benches, such as dSPACE real-time hardware. The notation XIL indicates that the standard can be used for various *in-the-loop* systems, e.g., SIL, MIL, PIL, and HIL. The XIL API standard is defined by the Association for Standardisation of Automation and Measuring Systems (ASAM).

ASAM General Expression Syntax (ASAM GES) The syntax definition that is used in AutomationDesk to specify trigger conditions. It is part of the XIL API standard that is defined by the Association for Standardisation of Automation and Measuring Systems (ASAM).

Automation block A part of a sequence that implements an automation task, similar to a subroutine.

Templates for automation blocks are provided by AutomationDesk libraries. Via the Sequence Builder, you can arrange automation blocks to implement the control flow of your automation task.

AutomationDesk Options A dialog that lets you modify the appearance and behavior of some AutomationDesk panes and the layout of the generated reports.

Automotive Simulation Model (ASM) The dSPACE product that provides open MATLAB®/Simulink® models that are relevant for the simulation of automotive engines (gasoline and diesel) and vehicle dynamics.

B

BLKX file An AutomationDesk element XML file that contains the specification of a saved or exported AutomationDesk element.

You can import and export BLKX files in the Project Manager, the Sequence Hierarchy Browser, the Sequence Builder, and the Library Bowser.

Block-specific data object A data object that resides in the interface of an automation block. It can be used to parameterize the block or to return a resulting data object after block execution.

Most blocks provided by AutomationDesk provide a static interface. However, some blocks let you add data objects to their interfaces dynamically, for example, Exec blocks.

Bookmark A label that you can attach to an automation block to use it later for quick navigation within the user interface.

Breakpoint A flag that you can set for a sequence or an automation block that pauses the execution in debug mode when the element with a set breakpoint is reached. You can manually control whether to resume the execution or to terminate it.

Built-in library The type of library that is included in AutomationDesk as a software component.

In contrast to custom libraries, you cannot create your own built-in libraries and you cannot view the library's source code.

C

Capture A data object type of the ASAM AE XIL API that is used to parameterize the capturing of measurement data.

In addition to the model access port (MAPort) to be used and the variables to be captured, you can specify, a condition to start or to stop data capturing, for example.

CaptureResult A data object type of the ASAM AE XIL API that is used to handle the captured data. It contains the time stamps and the related measured values of the captured variables.

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>`

Component Object Model (COM) An interface in Microsoft Windows that allows software products of different providers to communicate and to control each other.

ControlDesk The dSPACE software product for managing, instrumenting and executing experiments for ECU development. ControlDesk also supports calibration, measurement and diagnostics access to ECUs via standardized protocols such as CCP, XCP, and ODX.

Control-flow-based testing A test strategy that is based on implementing an automation task by specifying its control flow in sequences.

Custom library The type of library that you can create and include in AutomationDesk. The elements that you can add to a custom library are templates for data objects, automation blocks, and sequences. You can use the library elements as templates by adding library links to projects or sequences.

Some predefined custom libraries are part of the AutomationDesk product. They are read-only by default.

D

Data object Objects that can store a value according to the data object's type. You can specify a data object *by value* via an editor that depends on the type or *by reference* via the Data Object Editor.

Data objects can be instantiated specific to a project, to a sequence, or to an automation block.

Templates for data objects of various types are provided via AutomationDesk libraries and can be created via the Project Manager or the Sequence Builder, for example.

Data Object Editor A pane that lets you access the values and references of the data objects of the selected object.

Data Object Selector A dialog that lets you specify a data object by selecting one from the tree of available data objects.

DataContainer An element that lets you bundle data objects to structure them. DataContainers can be nested.

Debug mode A mode that lets you execute a project or a sequence successively and control the execution manually, for example, by using breakpoints.

Documents folder A standard folder for user-specific documents.

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

dSPACE Help The component that contains all the relevant user documentation for dSPACE products. Via the **F1** key or the **Help** button in the dSPACE software, you get context-sensitive help on the active context.

dSPACE Log A pane that displays the errors, warnings, information, and advice issued by all installed dSPACE products.

E

Edit dialog The dialog that lets you specify the value of a data object. The default edit dialog depends on the data type of the data object, but you can also use a customized edit dialog.

Electrical error simulation (EES) The simulation of errors in the wiring, such as loose contacts, broken cables, or short-circuits. Electrical error simulation is performed by the EES hardware of an HIL simulator.

Electrical error simulation port (EESPort) A data object type of the ASAM AE XIL API that is used to provide access to the electrical error simulation (EES) hardware of an HIL simulator.

Element The representation of a resource of a project in the Project Manager or a library in the Library Bowser.

An element is displayed as an icon that reflects the element's type followed by the element's name.

Error configuration file A file in XML format that contains the specification of the simulated electrical errors as a series of states which are each specified via an error set.

Error set A list of electrical errors that occur to the signals at the same time and that specifies the simulated state of the wiring. An empty error set specifies a state with no errors.

Exec block An automation block that is specified by the Python script to be executed.

You can edit the script via AutomationDesk's Python Editor.

F

FDX file An AutomationDesk project folder legacy XML file that contains the specification of an exported AutomationDesk project folder.

These files were created using AutomationDesk 6.0 or earlier. You can import FDX files in the Project Manager.

H

Hyperlink A click-able reference. When you click the link, the target is opened in an appropriate component.

I

Input dialog A dialog window that demands a manual input.

Instance description The property of an instantiated element that contains a text which describes the element's purpose.

L

LabeledValue A type of data object for which you can define a dictionary of valid label-value pairs. LabeledValues can be set either by specifying a label or by specifying a value.

LFX file An AutomationDesk library folder legacy XML file that contains the specification of an exported library folder.

These files were created using AutomationDesk 6.0 or earlier. You can import LFX files in the Library Bowser.

Library A container for templates that you can use to instantiate data objects, sequences, or automation blocks in your projects.

Libraries are handled via the Library Bowser. Each library is organized as a tree and can be structured using library folders.

There are built-in libraries and custom libraries.

Library Bowser A pane in AutomationDesk that provides access to the elements of the open libraries.

Library folder An element that structures the contents of a library as a tree.

Library link A type of element that you can create in a project or sequence. This type of element is linked to a template in a library. Depending on the link mode, the library link represents an instance of the linked library element or a reference to this library element.

Library links let you reuse a library element at multiple positions in one or multiple projects.

Link mode The way in which an instantiated object in your project can be connected to its related template in the library.

The link mode determines the synchronization behavior after you modified an object's template.

The following link modes are available:

- *Dynamically linked* - A modification of the template takes immediate effect.
- *Statically linked* - A modification of the template takes effect after you manually synchronized it.

If you break the link between an instantiated object and its template, the object becomes independent from the template and cannot be linked again.

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>
```

M

Mapping (For XIL API Testbench only) An object type of the ASAM AE XIL API for a data object that contains a mapping of variable aliases to their model paths in the related simulation application.

Only one Mapping data object is supported per project and it always resides at the top level of the object hierarchy.

Mapping Editor (For XIL API Framework only) A component that lets you configure an XIL API Framework. This includes, for example, the mapping of aliases to model paths, which you can use in your test cases and which are required to access variables via a model access port.

The configuration is saved to a framework configuration file (XML) as well as related port configuration and mapping files.

Mapping Viewer A pane that displays the contents of the used variable mapping.

If you are working with an XIL API Framework, the mapping relates to the framework configuration, which you can edit via the Mapping Editor.

If you are working with the XIL API Testbench, the mapping relates to the project's Mapping data object, which you can edit in the Mapping Viewer.

MAT file A file that contains measurement data in a format that allows data exchange with MATLAB.

MDF file A file that contains measurement data in a format that complies with the ASAM Common MDF standard. For version 4.1 of this standard, the file name extension is MF4.

Message dialog A dialog that demands manual confirmation for a message, error, warning, or information.

Message Viewer A pane that displays the history of all error and warning messages that occur while you are working with AutomationDesk.

MF4 file Refer to MDF file.

Model access port (MAPort) A data object type of the ASAM AE XIL API that is used to provide access to the variables of a running simulation application.

ModelDesk The dSPACE software product for parameterizing ASM models via graphical representations of the modeled components and controlling the related real-time simulation, offline simulation, or MATLAB®/Simulink® simulation.

MotionDesk The dSPACE software product that lets you visualize the movement of 3-D objects controlled by a running simulation application.

O

Object hierarchy The hierarchy tree that is built by all objects that are instantiated in a specific project.

Offline simulation application (OSA) A simulation application that can be executed without real-time hardware on a host PC with VEOS. The OSA file that implements the simulation application can be built from a Simulink model by the VEOS Player.

Operation mode A feature that is provided by some libraries and lets you decide whether to work online with the related device or to work with previously recorded data.

Operation signal The signal type of signals that are specified as an arithmetic operation (addition or multiplication) of two other signals.

Output Viewer A pane that displays all output messages generated by AutomationDesk.

P

PADL.ZIP file An AutomationDesk legacy element archive file that contains the specification of a custom library, a library folder, or a template which was managed under version control.

These files were created using AutomationDesk 6.1 or earlier and must be migrated to let you continue to work under version control.

PADP.ZIP file An AutomationDesk legacy element archive file that contains a project, a project folder, a sequence, or a result which was managed under version control.

These files were created using AutomationDesk 6.1 or earlier and must be migrated to let you continue to work under version control.

Pane The section of a window that provides related controls. AutomationDesk panes are arranged in view sets.

Parameter Any variable type that can be calibrated.

PCONFIG file An ASAM AE XIL API EES port configuration file that provides the hardware-dependent information for an electrical error simulation (EES) in XML format.

Platform A software component representing a simulator where a simulation application is computed in real-time (on dSPACE real-time hardware) or in non-real-time (on VEOS).

Platform Manager A software component that is commonly used by various dSPACE products to register and access platforms and to control the execution of simulation applications on the platforms.

Project A container for all instantiated resources that implement a specific automation task.

Projects are handled via the Project Manager. Each project is organized as a tree and can be structured using project folders.

Project folder An element that structures the contents of a project as a tree.

Project Manager A pane in AutomationDesk that provides access to the elements of the open projects.

Project-specific data object A data object that is created within a Project or a Project folder in the Project Manager. It can be used to parameterize elements a lower level in the object hierarchy.

Properties A pane that lets you access the properties of selected elements.

Python Editor A component that lets you edit the Python scripts for Exec blocks, their templates, and Python modules and packages that are integrated in AutomationDesk libraries. Each of these elements can be opened in a separate Python Editor pane.

R

Real-time application An application that can be executed in real time on dSPACE real-time hardware. A real-time application can be built from a Simulink model containing RTI blocks, for example.

Real-Time Testing (RTT) The dSPACE software product that provides components for creating and executing Python scripts which run on the real-time hardware in parallel to the real-time application.

Record depth The attribute of an execution that specifies which project elements are to include in the execution's result depending on the element's result levels.

The following record depths are provided:

- No result
- High elements only
- High and medium elements

Report A document in PDF or in HTML format that is generated from an execution's result.

Result A set of data that results from the execution of a project, a project folder, or a sequence.

From a result, you can generate a report.

Result Browser A component that displays the result of the execution of a project, a project folder, or a sequence during the execution in form of a tree of the involved data objects and their values .

Each result that you open in the Result Browser is displayed in a separate pane.

Result level The attribute of an element that specifies whether to include the element in an execution's result, depending on the execution's record depth.

AutomationDesk provides the None, Medium, and High result levels.

Result parameter The attributes that specify whether an element is included in an execution's result. For this, AutomationDesk provides the result level and the record depth attributes.

Root element The top-level element of a tree data structure. A root element represents the entire element tree of a project in the Project Manager or a library in the Library Bowser, for example.

S

Segment signal The signal type of the signals that are specified as a sequence of signal segments.

Sequence The implementation of an automation task as a control flow specified with automation blocks.

Sequences are edited via the Sequence Builder.

Sequence Builder A component that lets you graphically edit the control flow of a sequence, sequence template or subsequence template. Each of these elements can be opened in a separate Sequence Builder pane.

Sequence Hierarchy Browser A pane in AutomationDesk that provides access to the elements of the sequence that is currently displayed in the Sequence Builder.

SequenceFrame A template that is provided by the Framework Builder built-in library and that lets you specify a predefined frame for implementing similar sequences.

SFX file A sequence frame legacy XML file that contains the specification of an exported SequenceFrame object.

These files were created using AutomationDesk 6.0 or earlier. You can use the Project Manager to import SFX files for handling instantiated sequence frames and the Library Bowser for handling sequence frame templates.

Signal The specification or measurement of the change of a value over time.

Signals can be specified by their shape in a signal description set as a segment signal or as an operation signal.

Signal description set A container for a set of signal specifications that implement a specific signal-based test.

Signal description sets are handled via the Signal Editor as a table of the contained signals.

Signal Editor A component that lets you graphically edit a signal description set as a table of its contained signals.

Multiple signal description sets can be opened in separate Signal Editor panes.

Signal file A file in CSV format that defines via failure classes which electrical errors can be simulated by the specific EES hardware.

Signal generator A software component, that can be configured and controlled via a data object in AutomationDesk. A signal generator can be downloaded to a platform and stimulate variables in a running simulation application in real-time.

Signal segment One member in the sequence of segments that builds a segment signal. A segment is specified by its type and by its other properties.

The segment type is specified at the segment's creation via the Signal Selector. Its other properties can be specified via the Signal Editor or the Properties panes.

Signal Selector The pane that provides elements to add segment signals, operation signals, and segments of various segment types to your signal description set by dragging them to the Signal Editor.

Signal-based testing A test strategy that is based on implementing an automation task by using templates of the **Signal-Based Testing** library and specifying all involved signals in a signal description set.

Simulation application The generic term for offline simulation application (OSA) and real-time application.

SQX file An AutomationDesk sequence legacy XML file that contains the specification of an exported AutomationDesk sequence.

These files were created using AutomationDesk 6.0 or earlier. You can use the Project Manager to import SQX files for handling instantiated sequences and the Library Bowser for handling sequence templates.

Stylesheet An XSL file that specifies the layout for the generation of a report from an execution's result.

STZ file A ZIP file that contains the description of a signal description set in STI format. The STI format is defined by the ASAM AE XIL API standard.

You can create and manage STZ files in AutomationDesk's Signal Editor.

Subsequence An automation block that can contain other automation blocks to implement a part of a sequence's control flow, for example, a loop or a subroutine.

T

Task A thread that is executed on dSPACE real-time hardware.

The execution of tasks is triggered by timer events, I/O events, or software events.

TBX file A block template legacy XML file that contains the specification of an exported library folder.

These files were created using AutomationDesk 6.0 or earlier. You can import TBX files in the Library Bowser.

Template The reusable pattern of a data object, an automation block, or a sequence.

To make a template executable, you must instantiate it as an object in your project.

Template description The property of a template that provides a text which describes the template's purpose.

TSX file A sequence frame legacy XML file that contains the specification of an exported TestSequence (Test Framework) object.

These files were created using AutomationDesk 6.0 or earlier. You can use the Project Manager to import TSX files for handling instantiated sequence frames and the Library Bowser for handling TestSequence templates.

U

User function The call of an external program that you can integrate in AutomationDesk's user interface.

V

Value Editor A component that opens a modal Input dialog to edit the selected data object's value.

The appearance of the dialog depends on the type of the selected data object.

Variable A parameter in the simulation application that can be read and written.

A parameter identified by its variable path.

Variable description file The SDF file, the RTA file, or the OSA file that contains the specifications for an executable simulation application.

Variable path The path to the variable in the hierarchy of the model from which the simulation application is built.

Variables pane A component that lets you edit the configuration of an model access port (MAPort). You can select the platform type to be accessed and specify the variable description file to be used. Then you can browse the tree of the provided model variables. Each MAPort configuration can be opened in a separate Variables pane.

Variant A type of data object that can reference other data objects of any type.

VEOS A dSPACE software product that can execute offline simulation applications on a HostPC independently of real time. No real-time hardware is required.

Verdict A type of data object that is used to qualify the current success status of a sequence, subsequence, or automation block.

View set A configuration of the screen arrangement. You can create various view sets and switch between them. By default, AutomationDesk provides the preconfigured view sets Sequences, Signals, and Execution.

VirtualCOM An interface object for handling AutomationDesk's COM objects. VirtualCOM ensures a proper cleanup of deleted objects in AutomationDesk's namespace.

W

Working area The central area of AutomationDesk's user interface.

X

XIL API Framework An access layer that is defined in the ASAM AE XIL API standard.

It lets you centrally configure the access to the entire test infrastructure in XML files. This decouples test cases from the real and virtual test systems you use.

XIL API Testbench An access layer that is defined in the ASAM AE XIL API standard.

It lets you configure the access from a test to its environment, such as a simulator, by using ports. For example, the access to variables of a simulation application is configured by using a model access port. This decouples test software from test hardware.