ModelDesk

Project and Experiment Management

For ModelDesk 5.5

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

Contents

This document introduces you to the project and experiment management in ModelDesk.

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

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

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

or

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

Documents folder A standard folder for user-specific documents.

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

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

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

Accessing dSPACE Help and PDF Files

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

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

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

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

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

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

Basics and Instructions

Where to go from here

Information in this section

Creating Projects and Experiments

Introduction

Before you start parameterizing, you can get basic information on ModelDesk's project workflow and an introduction to the first steps.

Where to go from here

Information in this section

Basics of Projects and Experiments

Basics

In ModelDesk, the information and files required for the parameterization are collected in projects and experiments.

Project A project manages several experiments that belong together, such as the tasks for parameterizing specific model variants. It holds the experiments related to these tasks, and the Pool filled with the files for the entire project.

Experiment A container for collecting and managing information and files for parameterizing a simulation model. Only one experiment can be active at a time. An experiment can contain:

- Simulation model added to the experiment for parameterization
- Parameter sets, each containing the parameters of a simulation model variant
- Road created with the **Road Generator**
- Scenarios created with the **Scenario Editor** for simulating and defining the movements of the ASM vehicle and fellow vehicles (fellows) with absolute values or relative to the ASM vehicle.

Related topics

HowTos

References

How to Define a Project

Objective

Defining a project creates the basis for whatever you do in ModelDesk. In a project, you can manage different parameterization tasks that belong together. For example, you can group several ModelDesk experiments in one project. Each experiment contains one parameterization task comprising several parameter sets.

Preconditions

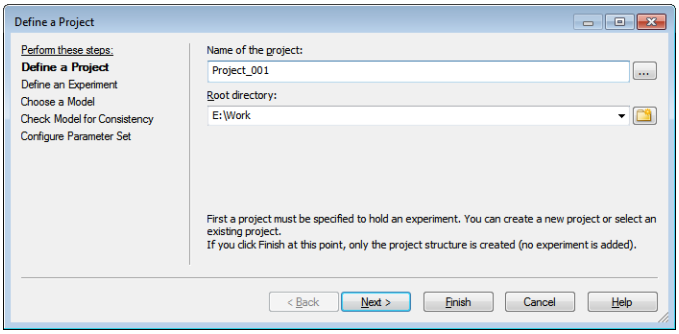
No other project must be open. If a project is currently open, ModelDesk lets you define a new experiment only.

Method

To define a project

1 On the File ribbon, click New – Project + Experiment.

The Define a Project dialog opens.



2 Enter a project name and the path of the root directory.

3 Click Next to continue with the Define an Experiment dialog.

Click Finish to create a project without an experiment. You can define an experiment afterwards.

Result

You created a project.

Next steps

You can define an experiment for the new project. Refer to How to Define an Experiment on page 12.

Related topics

References

How to Define an Experiment

Objective

Defining an experiment is the basis for carrying out parameterization and allows you to manage all the files and parameter sets for it. You can add one model to the experiment. Defining different experiments allows you to manage different models in one project.

Preconditions

The preconditions for defining an experiment are:

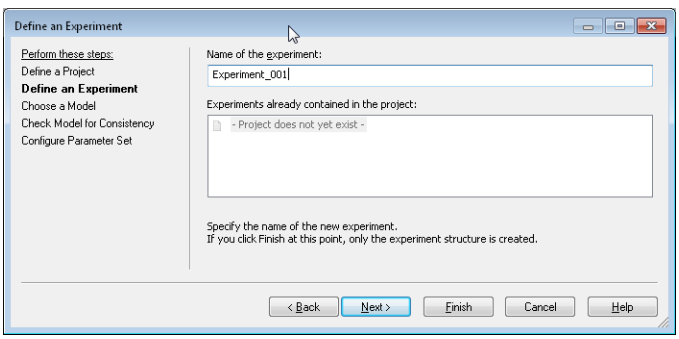
- A project must be open. Refer to How to Open a Project and Experiment on page 19.
- A project must be defined. Refer to How to Define a Project on page 11.

Method

To define a new experiment

1 On the File ribbon, click New – Experiment.

The Define an Experiment dialog opens.



2 In the Name of the experiment edit field, enter the experiment name.

3 Click Next to define an experiment and to proceed with choosing a model.

Click Finish to define an empty experiment without choosing a model. You can add a model afterwards.

Result

You created an experiment.

Next steps

You can add a model to the new experiment. Refer to How to Choose a Model and Initialize the Consistency Check (ModelDesk Parameterizing 🕮).

Related topics

References

How to Create a Project Based on an ASM Demo

Objective

To start with a ModelDesk project, you can use ASM demos installed with the ASM libraries.

ASM demos

The ASM installation contains several ASM demos. The ASM demos include all the necessary files for the simulation, for example:

- Simulation model based on the ASM blocks
- Simulation applications for the simulation platforms
- ModelDesk project for parameterizing the model
- ControlDesk project for experimenting with the model
- MotionDesk project for animation (if useful)

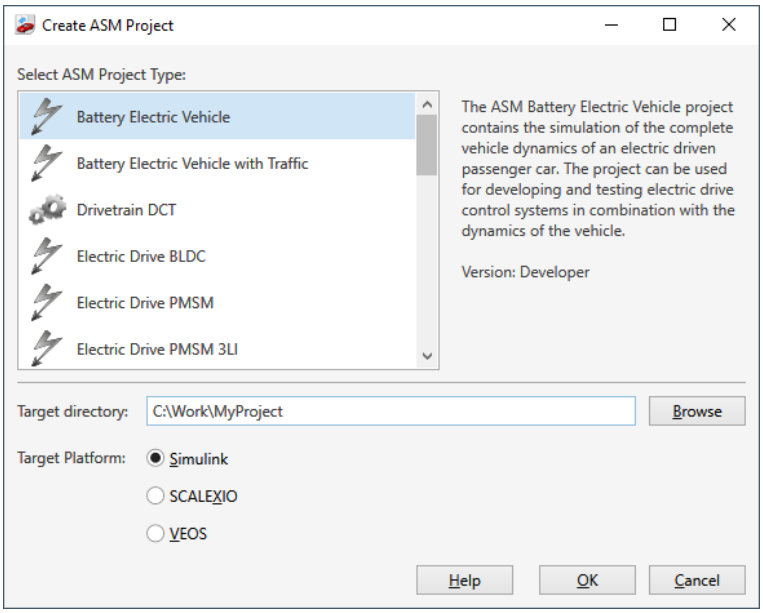
Preconditions

- No other project must be open.
- The license of the ASM library that is used by the ASM demos must be available.
- The ASM library must be decrypted.

Method

To define a project based on an ASM demo

1 On the File ribbon, click New – ASM Project.
The Define a Project dialog opens.



2 Select an ASM project type.

3 Specify the target directory. Select an empty directory or specify a new directory. You must have write permission to the directory.

4 Select the target platform to be activated in the experiment.

5 Click OK.

Result

ModelDesk copies all files of the selected ASM demo to the specified target folder and opens the project.

Next steps

Specify the the simulation platform. Refer to Handling the Automotive Simulation Model (ModelDesk Parameterizing 🕮).

Related topics

Basics

ASM Project Folder (ASM User Guide 🕮)

References

New ASM Project 37

Managing ModelDesk Projects

Where to go from here

Information in this section

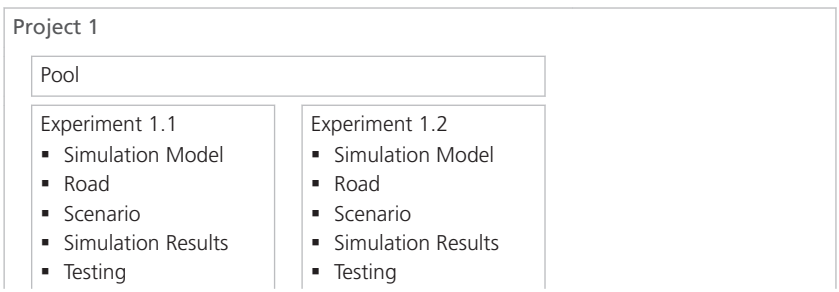
Managing Projects

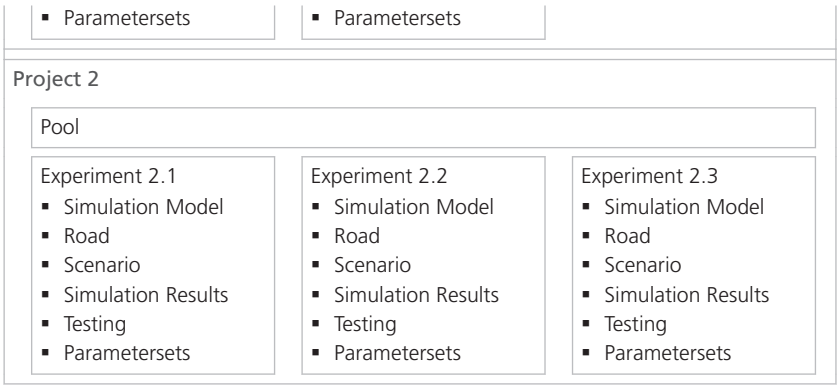
Introduction

Managing projects allows you to carry out and structure parameterization tasks according to your needs.

Project as the container for experiments

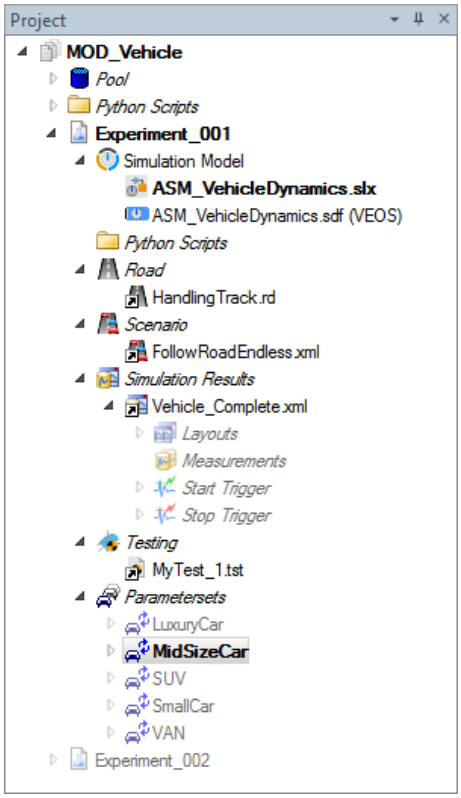
A project manages several experiments that belong together, such as the tasks for parameterizing specific model variants. It holds the experiments related to these tasks, and the Pool filled with the files for the entire project.





Handling projects with the Project Navigator

To handle projects and experiments, ModelDesk provides the Project Navigator. The Project Navigator gives you easy and intuitive access to all the experiments and files of a project. The illustration below shows the Project Navigator with a project and the files managed by the project.



To handle projects, the Project Navigator provides a context menu with commands for copying, adding, removing, or sorting experiments and files. The commands for handling individual files depend on the file type.

Hierarchical project structure

The structure displayed in the **Project Navigator** reflects the project hierarchy. The hierarchy helps you to organize parameterization tasks.

The items in a project, i.e., experiments and the Pool, are structured like this:

Pool Project-specific files such as parameter files, files used for processing, roads, scenarios, or tests are automatically stored in the Pool. From there, they are available for the entire project and linked to the experiments belonging to it. Pool files can also be exported and imported.

Experiments A project contains one or more experiments, only one of which can be active at a time. An experiment contains:

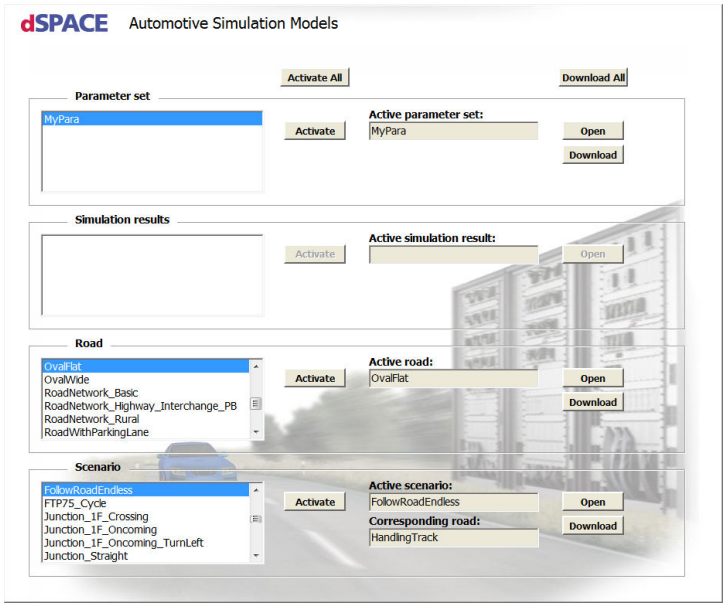
- Simulation model added to the experiment for parameterization and the assigned platform
- Parameter sets containing each the parameters of a simulation model variant
- Measurement data that is used for processing (calculating parameter values on basis of measured data)
- Road created with the **Road Generator**
- Scenarios created with the **Scenario Editor** for simulating scenarios which define the movements of the ASM vehicle and fellow vehicles (fellows) with absolute values or relative to the ASM vehicle
- Tests created with the ModelDesk testing component
- Simulation results which are created by the **Plot Manager**

Tip

For further information on the **Project Navigator** and illustrations of the project items, refer to Project Navigator on page 42.

Configuration page

ModelDesk provides the **Configuration** page, which lets you access the configured parameter sets, and the available simulation results, roads, and scenarios. The **Configuration** page also provides commands to these elements, such as **Activate**, **Open**, and **Download**. Refer to Configuration Page on page 36.



Project root folder for rooting projects

Each ModelDesk project is related to a project root folder. This is the folder on your file system to which ModelDesk will save all the experiments and files of a project. Several projects can use the same project root folder.

Default project root folder ModelDesk will use the Documents folder as the default project root folder unless you specify a different one. ModelDesk saves all project files to this folder.

Specifying further project root folders Although there is a default project root folder, you can specify further project root folders on the Project Page in the General Properties dialog. This allows you to specify different destination folders for your projects, and to group projects.

Automatic file archiving in the Pool folder

ModelDesk archives project-specific files such as parameter files or roads in the Pool automatically. For example, when you create a road, ModelDesk's Road Generator automatically saves it to the project's Pool Road folder (see the illustration above), from where you can link it to the experiment. You have neither to save files generated by ModelDesk to the file system manually or to specify file paths.

Related topics

HowTos

How to Open a Project and Experiment 19
How to Work with Several Experiments 20

How to Work with Several Parameter Sets (ModelDesk Parameterizing 🕮)

References

Project Page 44

How to Open a Project and Experiment

Objective

Opening an experiment and the project it belongs to is necessary for carrying out a parameterization task, which has to be done in an experiment.

Opening projects created with an earlier ModelDesk version

You can migrate projects created with ModelDesk 4.1 and higher. Projects created with an earlier ModelDesk version cannot be migrated. Refer to Migrating a Project and Its Experiments on page 23.

Method

To open a project and experiment

1. On the File ribbon, click Open – Project + Experiment.
   ModelDesk opens the Select an experiment dialog.
2. In the Root directory list, select the project root directory containing the project and experiment you want to open.
3. In the Projects and experiments list, select the experiment you want to open.
4. Click OK.

Result

ModelDesk opens and activates the selected experiment. The current project is closed.

Related topics

References

Open Project + Experiment 40

How to Rename a Project or Experiment

Objective

You can rename a project or experiment to use a different name for it. You can also save a project or experiment under a different name if you want to work with a copy of the project or experiment.

Preconditions

The project must be open in ModelDesk.

Method

To rename a project or experiment

1. In the Project Navigator, open the context menu of the project or experiment item.
2. From the context menu, select Rename or Save As.

   A dialog opens for you to specify the new name.
3. In the dialog, enter a new name. The name must be unique and must not contain any of the following characters: * ? | < > : / \ "
4. Click OK.

Result

The project or experiment is renamed or saved under a different name.

Related topics

References

Rename....48
Save As / Save Project As....50

How to Work with Several Experiments

Objective

Working with several experiments lets you handle several parameterization tasks in one project.

Working with several experiments

Working with several experiments comprises:

- Defining additional experiments
- Activating experiments

Defining additional experiments

You can define additional experiments for each project to organize parameterization tasks. For instructions, refer to How to Define an Experiment on page 12.

Activating experiments

Although a ModelDesk project can contain several experiments, you can work with only one experiment – the active experiment – at a time. All the other experiments of the project are inactive. To work with another experiment of the project, you have to activate it first.

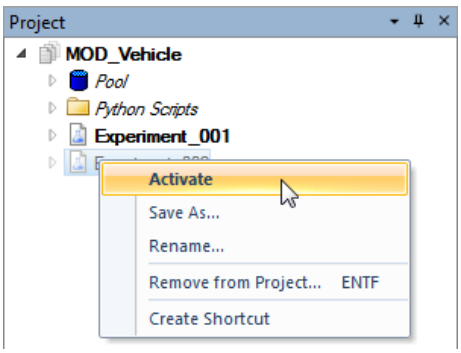
Restrictions

Activating an experiment is possible only within a currently open project.

Method

To work with several experiments

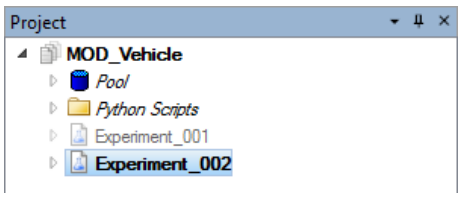
1 In the Project Navigator, right-click the inactive experiment you want to activate.



2 From the context menu, select Activate (Experiment).

Result

ModelDesk activates the selected experiment, and deactivates the other.



Related topics

HowTos

How to Open a Project and Experiment 19

References

Activate (Experiment) 30

How to Start MotionDesk or ControlDesk from ModelDesk

Objective

You can start MotionDesk or ControlDesk and load their projects that are related to the active ModelDesk project.

Basics

You must select the MotionDesk or ControlDesk project in ModelDesk before you can use the shortcuts. When you click the shortcuts, the related tools are started and the selected projects are loaded. The first experiment of the projects are activated.

Preconditions

The MotionDesk or ControlDesk project must exist.

Method

To start MotionDesk or ControlDesk from ModelDesk

1. In ModelDesk, click Home – Toolchain – dialog launcher.

   Or

   In the Project Navigator, open the context menu of the project and select Configure External Project Settings.

   The Toolchain Settings dialog opens.
2. In the Toolchain Settings dialog, select the projects that are related to the active ModelDesk project.
3. Click OK.

   When the project is specified, you can use the shortcuts to start the tools.
4. To start MotionDesk or ControlDesk, click Home – Toolchain – Open MotionDesk or Open ControlDesk.

Result

MotionDesk or ControlDesk starts, loads the specified project and activates the first experiment.

Related topics

Basics

Managing an Experiment (ModelDesk Basics 🕮)

References

Configure External Project Settings/Tool Chain Settings Dialog.....33
Open ControlDesk.....40
Open MotionDesk.....41

Migrating a Project and Its Experiments

Introduction

Projects and their experiments created with earlier ModelDesk versions must be migrated to use them with the current ModelDesk version.

Where to go from here

Information in this section

Basics on Migrating a ModelDesk Project

Introduction

Projects created with earlier ModelDesk versions cannot be used with the current ModelDesk version. You must therefore migrate these projects.

Overview

Migrating a ModelDesk project means migrating the project data and updating the model information.

ModelDesk versions You can migrate from the eight previous ModelDesk versions and use the project with the current ModelDesk version. The earliest version that can be migrated is ModelDesk version 4.5 of dSPACE Release 2017-A. Projects created with an earlier ModelDesk version cannot be migrated.

Migrating project data When you open the old project in ModelDesk for the first time, ModelDesk asks if you want to migrate the project data. If you click Yes, all the parameter, road, and scenario files are migrated. The files are migrated step-by-step from the original ModelDesk version up to the current ModelDesk version. Refer to Migrating project data on page 24.

Updating the model information The model information is updated when you open the experiment in a migrated project for the first time. You can do so directly after project data migration or later. If you reject the update, the model information remains obsolete and you cannot work with it.

Migration workflow

This workflow shows the steps necessary for migrating a project and its experiments:

1. Open an old project in ModelDesk and click **Yes** in the migrate query.

   If you refuse to migrate the project, it closes and you cannot work with it.
2. ModelDesk backs up the project data. Refer to Backing up project data on page 24.
3. Open an experiment in the migrated project and click **Yes** in the update query.

   If you refuse to update the experiment, it closes and you cannot work with it.
4. Repeat step 3 for every experiment in the migrated project that needs updating.

Documenting migration

The information on the performed migration is written to the *log files*. There is a separate log file for each migration step. The files are in the root folder of the project and named according to the following convention: `<ProjectName>_Migration<SourceModelDeskVersion>To<TargetModelDeskVersion>.log`. A log file contains details on the migrated data and problems that occurred during migration.

Backing up project data

Before migrating the project data, ModelDesk saves the project folder and all subfolders to a ZIP file. After successful migration, the ZIP file is saved to the folder of the migrated project and named according to the following convention: `<ProjectName>_Migration_Backup<SourceModelDeskVersion>.zip`.

If migration fails, the original project data is restored and the ZIP file deleted. If the original project data cannot be restored, ModelDesk saves the ZIP file to the project folder.

Migrating project data

Migrating the data of a project updates the parameter, scenario, and road files in the Pool, and the parameter sets to make them available in the current ModelDesk. The files are migrated step-by-step. One step migrates the files from their current version to the next newer one. After a migration step, all files in the Pool have the same version.

The information on each migration step is written to a log file. Refer to Documenting migration on page 24. If a migration step fails, ModelDesk generates a message and restores the original project data.

Migrating parameter files ModelDesk migrates the parameter files contained in the Pool `Parameter` subfolder.

It compares the parameter files with a template and handles them as follows:

- Adds new files that are not yet contained in the project with default values.
- Deletes obsolete files.
- Renames files whose names changed.
- Moves files whose locations in the project folder changed.

- Adds new parameters to the files.
- Deletes obsolete parameters.
- Moves parameters from one file to the other if the parameter is moved in the ASM from one block to the other.
- Renames parameters if the parameter address changes.
- Renames Pool folders whose names changed.

Migrating Road Generator files The Road Generator migrates its files contained in the Pool `Environment\Road` subfolder. It migrates only files known to the project. Unknown files are not migrated, even if they are correct road XML files. These files stay unchanged in the Pool folder. MAT files generated by the Road Generator for downloading are deleted to avoid inconsistencies between the XML and MAT files.

Migrating the Road Generator files is successful only if all the files known to the project are migrated to the target ModelDesk version and the corresponding MAT files are deleted.

Migrating traffic scenario files The Scenario Editor migrates the files contained in the Pool `Environment\Traffic` (ModelDesk 4.6 and earlier) or `Environment\Scenario` (ModelDesk 4.7 and later) subfolder and stores them in the `Environment\Scenario` subfolder. It migrates only files known to the project. Unknown files are not migrated, even if they are correct traffic XML files and remain unchanged in the Pool folder. MAT files generated by the Traffic Editor for downloading are deleted to avoid inconsistencies between the XML and MAT files.

Migrating the traffic scenario files is successful only if all the files known to the project are migrated to the target ModelDesk version and the corresponding MAT files are deleted.

Related topics

Basics

Basics on Model Migration (ASM User Guide 🕮)

How to Open an Experiment After Migration

Objective

Each experiment that references a model contains information about the parameters of the Automotive Simulation Models (ASM) which were found when the referenced model was last analyzed. This information must match the referenced model as well as the current ASM version.

After a project is migrated, the model information in its experiments no longer matches the updated ASM, and you cannot work with the experiments. You must first update their model information.

Updating the model information

How you update the model information depends on the type of the referenced model:

- For a Simulink model, you must provide the new model file by opening and saving the referenced ASM in MATLAB.
- For a real-time model, you must provide the new SDF file by building the migrated ASM in MATLAB.

Preconditions

- The project to which the experiment belongs was migrated.
- The ASM referenced by the experiment was migrated by opening and saving it in MATLAB.
- The migrated ASM was built in MATLAB to get a new SDF file for the real-time simulation.

Method

To open an experiment after migration

1. On the File ribbon, click Open – Experiment.

   ModelDesk opens the Select an experiment dialog.
2. In the Root directory list, select the project root directory containing the project and experiment you want to open.
3. In the Projects and experiments list, select the experiment you want to open.
4. Click OK.

   ModelDesk closes the current project and opens the project and experiment. If the model information in the experiment is obsolete, the Model Information Outdated dialog opens.
5. Click Update.

   If the model and SDF files are obsolete, ModelDesk generates warning messages.

   The Choose a model dialog opens.
6. Keep the settings for the Simulink and real-time model or select new files via the Browse buttons.
7. Follow the instructions in the dialog.
8. Click Finish. Closing the wizard in any other way returns the dialog to Model Information Outdated.

Result

The experiment is migrated.

Related topics

Basics

Basics on Migrating a ModelDesk Project..... 23

References

Model Information Outdated Dialog (ModelDesk Parameterizing 🕮)

Reference Information

Where to go from here

Information in this section

Activate (Experiment)

Access

This command is available only for inactive experiments. You can access it via:

Ribbon	None
Context menu of	**Project Navigator** – inactive experiment

Shortcut key	None
Icon	None

Purpose

To activate an experiment.

Description

Only one experiment in a ModelDesk project can be active at a time. All the other experiments are inactive. To work with an experiment, it must be active.

Related topics

Basics

Basics of Projects and Experiments..... 10

HowTos

How to Define an Experiment..... 12

Ascending

Access

You can access this command via:

Ribbon	None
Context menu of	Project Navigator – Sort
Shortcut key	None
Icon	None

Purpose

To sort the files of a folder in ascending alphabetical order.

Result

The files are sorted in ascending alphabetical order.

Related topics

Basics

Managing Projects..... 15

Close Project / Close

Access

This command is available only if at least one project is open. You can access it via:

Ribbon	File
Context menu of	Project Navigator – project
Shortcut key	None
Icon	None

Purpose

To close the currently loaded project and the experiment belonging to it.

Configure Experiment

Access

You can access this command via:

Ribbon	None
Context menu of	Project Navigator – project
Shortcut key	None
Icon	None

Purpose

To configure, activate, or delete an experiment.

Result

ModelDesk opens the Configure Experiment dialog for you to configure the experiment.

Description

If another experiment is already open, this is the active one. To work with the newly configured experiment, you must activate it. See the description above.

Configure Experiment dialog

To configure a new experiment, specify or change the settings and activate the selected one.

New Lets you add a new experiment to the project. ModelDesk opens the Define an Experiment dialog for you to specify the settings.

Activate Lets you activate an experiment. Only one experiment in a ModelDesk project can be active at a time. All the other experiments are inactive. To work with an experiment, it must be active.

Delete To delete the current experiment from the project.

OK Lets you finish experiment configuration. This button is disabled until configuration has finished.

Related topics

Basics

Managing Projects..... 15

HowTos

How to Define an Experiment..... 12

Configure External Project Settings/Tool Chain Settings Dialog

Access

You can access this command via:

Ribbon	**Home** – **Tool Chain** – dialog launcher
Context menu of	Project node in the **Project Navigator**
Shortcut key	None
Icon	None

Purpose

To specify the MotionDesk and ControlDesk projects that are related to the active ModelDesk project.

Description

Before you can use the **Open ControlDesk** or **Open MotionDesk** command, you must specify the project to be loaded.

Dialog settings

MotionDesk project Lets you select the related MotionDesk project.

ControlDesk project Lets you select the related ControlDesk project.

Related topics

HowTos

How to Start MotionDesk or ControlDesk from ModelDesk 22

References

Open ControlDesk 40
Open MotionDesk 41

Create Shortcut

Access

You can access this command via:

Ribbon	None
Context menu of	Project Navigator – experiment
Shortcut key	None
Icon	None

Purpose

To create a desktop shortcut of an experiment.

Result

A shortcut to the experiment is created on your desktop.

Description

You can create a desktop shortcut for each ModelDesk experiment. This allows you to open ModelDesk and quickly load a specific experiment.

To open an experiment via desktop shortcut, ModelDesk must be closed.

Related topics

Basics

Basics of Projects and Experiments 10

Descending

Access

You can access this command via:

Ribbon	None
Context menu of	Project Navigator – Sort
Shortcut key	None
Icon	None

Purpose

To sort the files of a folder in descending alphabetical order.

Result

The files are sorted in descending alphabetical order.

Related topics

Basics

Managing Projects..... 15

Download All

Access

You can access this command via:

Ribbon	Home – Experiment
Context menu of	None
Shortcut key	None
Icon	
Button	Button on the Configuration page

Purpose

To download road, scenario, and parameter sets to the simulation system.

Result

All the active parts are downloaded to the real-time hardware, Simulink, or VEOS.

Related topics

References

Configuration Page

Access

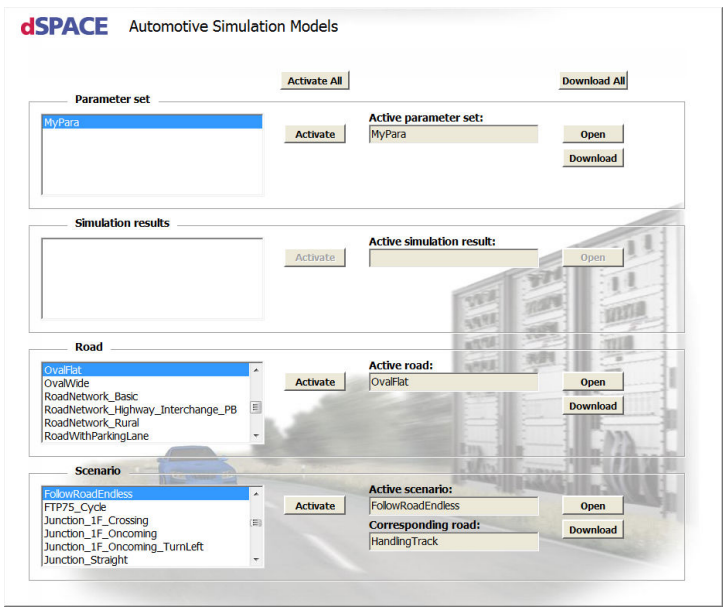
The Configuration page opens automatically when you select the node of the current active experiment in the Project Navigator.

Purpose

To manage the simulation results, scenarios, roads, and parameter sets added or linked to the current active experiment.

Description

The following illustration shows the Configuration page.



Dialog settings

Activate all Lets you activate all the selected road, scenario, or parameter sets at once.

Download all Lets you download the active road, scenario, or parameter sets at once.

Parameter Set Lists all the available parameter sets configured for the experiment for you to choose one.

Road Lists all the available roads for you to choose one.

Scenario Lists all the available scenarios for you to choose one.

Activate Lets you activate the selected element (road, scenario, or parameter set). Only one of these elements can be active in a ModelDesk project at a time. All the others are inactive. To work with an element, it must be active. The button is unavailable if no corresponding elements are available.

Active parameter set Displays the active parameter set.

Active road Displays the active road.

Active scenario Displays the active scenario.

Corresponding road Displays the name of the road if the current active scenario is created with road.

Open Opens the Road Generator, Scenario Editor, or navigation page to let you edit the selected file or parameter set.

Download Lets you download the active road, scenario, or parameter set.

Related topics

Basics

Changing the Background Image of the Configuration Page (ModelDesk Basics 🕮)
Managing an Experiment (ModelDesk Basics 🕮)

HowTos

How to Work with Several Parameter Sets (ModelDesk Parameterizing 🕮)

New ASM Project

Access

You can access this command via:

Ribbon	File – New
Context menu of	None
Shortcut key	None
Icon	
Others	Start page

Purpose

To use a ModelDesk project of an ASM demo.

Result

ModelDesk copies all the files of the ModelDesk project that belongs to an ASM demo model into the specified directory and opens the ModelDesk project.

Dialog settings

In the Create ASM Project dialog, you can select the ASM project type, target directory, and simulation platform.

Select ASM Project Type Lets you select the type of the ASM project. The list contains all the ASM demos that are installed and decrypted. The license of the corresponding ASM library must be available.

Target directory Lets you specify a folder where the files are saved. The target folder must be empty and you must have write permission for it. You can enter the full path in the edit field or click Browse to browse for a folder.

Target Platform Lets you select the simulation platform to be activated in the experiment.

Related topics

HowTos

How to Create a Project Based on an ASM Demo 13

References

Specify Platform Dialog 51
Start Page (ModelDesk Basics 🕮)

New Project + Experiment/New Experiment

Access

You can access this command via:

Ribbon	File – New
Context menu of	▪ Project Navigator – project ▪ Project Navigator (if no project is currently open)
Shortcut key	None
Icon	

Purpose

To define a new project or a new experiment.

Description

If no project is currently open, ModelDesk opens the **Define a Project** dialog. You have to define a new project or open an existing one before you can define a new experiment, see below.

If a project is currently open, ModelDesk opens the **Define an Experiment** dialog. This lets you define a new experiment within the open project, see below.

Define a Project dialog

To define a new ModelDesk project.

Name of the project Lets you enter a project name in the edit field, or select an existing project via the Browse button. The name you enter must not contain any of the following characters: `* ? | < > : / \ "`

Root directory Lets you select a project root folder.

Finish Lets you finish project creation without having to define an experiment. This button is disabled if the name of the project edit field is empty.

Define an Experiment dialog

To define a new ModelDesk experiment.

Name of the experiment Lets you enter an experiment name. It must not contain any of the following characters: `* ? | < > : / \ "`

Experiments already contained in the project If your project already contains experiments, they are displayed here (no changes possible).

Finish Lets you finish experiment creation without having to add a model to the experiment. This button is disabled if the name of the experiment edit field is empty.

Related topics

HowTos

References

Open ControlDesk

Access

You can access this command via:

Ribbon	Home – Tool Chain
Context menu of	None
Shortcut key	None
Icon	

Purpose

To open the related ControlDesk project.

Description

Before you can use this command, the related project must be specified in the Toolchain Settings dialog.

Result

ControlDesk starts and loads the specified project and the first experiment.

Related topics

HowTos

How to Start MotionDesk or ControlDesk from ModelDesk..... 22

References

Configure External Project Settings/Tool Chain Settings Dialog..... 33

Open Project + Experiment

Access

You can access this command via:

Ribbon	File – Open
Context menu of	Project Navigator (if no project is currently open)
Shortcut key	`Ctrl+Shift+O`
Icon	

Purpose

To open an experiment and the project it belongs to.

Result

Opens the Select a Project dialog for you to select an existing project and experiment to open.

Description

If another experiment is already open, it and the project it belongs to are closed. Then the experiment you selected is opened and activated.

If the project and experiment were created with an earlier ModelDesk version, you must migrate the project and update the model information to use the project and experiment. Refer to Model Information Outdated Dialog (ModelDesk Parameterizing 🕮).

Select a Project dialog

To select an experiment to open.

Root directory Lets you select the project root directory.

Lets you select a new root folder.

Projects and experiments Lets you browse in the list of projects and experiments available in the selected project root directory.

Related topics

HowTos

How to Open a Project and Experiment.....19
How to Open an Experiment After Migration.....25

Open MotionDesk

Access

You can access this command via:

Ribbon	Home – Tool Chain
Context menu of	None
Shortcut key	None
Icon	

Purpose

To open the related MotionDesk project.

Description

Before you can use this command, the related project must be specified in the Toolchain Settings dialog.

Result

MotionDesk starts and loads the specified project and the first experiment.

Related topics

HowTos

How to Start MotionDesk or ControlDesk from ModelDesk..... 22

References

Configure External Project Settings/Tool Chain Settings Dialog..... 33

Project Navigator

Access

You can display the Project Navigator via:

Ribbon	View – Controlbar – Switch Controlbars – Project
Context menu of	None
Shortcut key	None
Icon	None

Purpose

To display the Project Navigator, which lets you manage a project and its experiments.

Description

The Project Navigator provides access to a ModelDesk project and all the items – folders and files – belonging to it. It displays all the items of the project hierarchically.

Management of a ModelDesk project In the Project Navigator, you can manage the items belonging to the currently open project. The Project Navigator provides a context menu for each item, allowing you to carry out tasks such as removing items.

Item type and status The Project Navigator displays each item with a symbol indicating its type and status.

Project

Project (only one project can be loaded at a time)

Folder with files belonging to the project and saved in the Pool

Experiment

- Active (bold text) and inactive (grayed text) experiments
- Model folder with model files belonging to the experiment
- Road folder with road model files belonging to the experiment
- Scenario folder with scenario files belonging to the experiment
- Simulation results folder
- Testing folder with test cases belonging to the experiment
- Parameter set folder with parameter sets belonging to the experiment

Model

- Simulink model added to the experiment
- Real-time model added to the experiment
- VEOS model added to the experiment

Road

- Road file linked to the experiment

Scenario

- Scenario file linked to the experiment

Simulation Results

- Configuration file linked to the experiment
- Layout folder
- Layout file linked to the current configuration
- Measurement folder
- MAT file linked to the current configuration
- Start or Stop Trigger folder
- Trigger file linked to the current configuration

Testing

- Active test case

Parameter Sets

- Active parameter set
- Inactive parameter set
- Processing
- Navigation page
- Parameter page
- Parameter file linked to the parameter set

Pool

- Configuration
- Conversion file
- Function file
- Layout
- Measurement
- Measurement data file

- Measurement type file
- Parameter file
- Road file
- Scenario file
- Setting file
- Start or Stop Trigger
- Test file
- Traffic driver file
- Traffic object file

Related topics

Basics

User Interface of ModelDesk (ModelDesk Basics 🕮)

Project Page

Access

This page is part of the **ModelDesk Options** dialog.

Purpose

To specify project root directories.

Description

To define and work with projects and experiments in ModelDesk, at least one project root directory must be specified.

Dialog settings

Root directories Lets you specify one or more project root directories.

`Ins`	To specify a new project root directory. A new line is added to the list of project root directories. You can enter a directory name in the edit field or select a directory via the Browse button.
`Del`	To remove the selected directory from the list of project root directories.
`Alt+↑`	To move the selected directory up in the list of project root directories.
`Alt+↓`	To move the selected directory down in the list of project root directories.
	To browse the contents of the selected directory.
`F2`	To edit the selected directory in the list of project root directories.

Only show projects that contain experiments generated by this product Activates a filter when you browse for projects/experiments via Open Project + Experiment on page 40.

Related topics

References

Options (ModelDesk Basics 🕮)

Project Wizard

Access

ModelDesk's Project Wizard consists of a sequence of 5 dialogs. You access the wizard by one of the following commands:

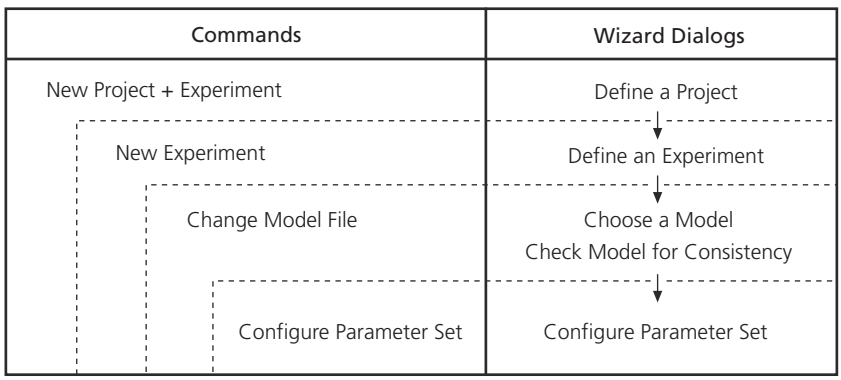
- New Project + Experiment / New Experiment
- Configure Parameter Set
- Change Model File

Purpose

To define a new project or experiment, add or replace a model file, check the model file for consistency, and configure a parameter set.

Description

You get only the wizard dialogs relevant to the command that you selected. The following illustration shows the commands, and the relevant dialogs in the order in which they appear.



For details on the dialogs, refer to:

- Define a Project dialog on page 39
- Define an Experiment dialog on page 39
- Choose a Model dialog (ModelDesk Parameterizing 🕮)

- **Check Model for Consistency dialog** (ModelDesk Parameterizing 🕮)
- **Configure Parameter Set / Configure** (ModelDesk Parameterizing 🕮)

Related topics

HowTos

How to Choose a Model and Initialize the Consistency Check (ModelDesk Parameterizing 🕮)
How to Configure a Parameter Set (ModelDesk Parameterizing 🕮)
How to Define a Project 11
How to Define an Experiment 12

References

Change Model File (ModelDesk Parameterizing 🕮)
Configure Parameter Set / Configure (ModelDesk Parameterizing 🕮)
New Project + Experiment/New Experiment 38

Recent Projects and Experiments

Access

You can access the command via:

Ribbon	File – Recently Used
Context menu of	None
Shortcut key	None
Icon	None
Others	Start page

Purpose

To open one of the most recent experiments that were opened in ModelDesk.

Description

If another experiment is already open, it and the project it belongs to are closed. Then the experiment you selected is opened and activated.

Recent Projects and Experiments

List of experiments Lets you select one of the most recent experiments that were open in ControlDesk.

Open (Available from the context menu of list items) To open the selected experiment. You can also simply open an experiment by left-clicking it in the list.

Clear Recent Projects + Experiments List (Available from the context menu of list items) To clear the list of recently opened projects + experiments.

Remove from List (Available from the context menu of list items) To remove the selected experiment from the list of recently opened projects + experiments.

Size of recent experiments list Lets you specify the maximum number of list entries. You can specify a value in the range 4 ... 100.

Reset Sort Direction (Available from the context menu of the column header of the list of experiments) You can click the column headers to sort the experiments in ascending or descending order according to a column. To remove this sorting you can reset the sort direction to its default, which is according to the time the experiments were last opened, starting with the most recently opened experiment.

Visible Columns - Path/Opened/Modified/Version (Available from the context menu of the column header of the list of experiments) Lets you specify whether to display:

- The path to the experiment folder
- The point in time when the experiment was opened last
- The point in time when the experiment was modified last

Related topics

References

Start Page (ModelDesk Basics 🕮)

Refresh

Access

You can access this command via:

Ribbon	None
Context menu of	Project Navigator – project
Shortcut key	None
Icon	None

Purpose

To renew the folder structure shown in the Project Navigator.

Result

The folder structure and included files are re-read and the view of the structure shown in the Project Navigator is refreshed.

Related topics

References

Project Navigator 42

Remove (from Project)

Access

You can access this command via:

Ribbon	None
Context menu of	▪ **Project Navigator** – **Road** ▪ **Project Navigator** – **Scenario** ▪ **Project Navigator** – Pool file ▪ **Project Navigator** – parameter file ▪ **Project Navigator** – inactive experiment ▪ Configuration page (road) ▪ Configuration page (scenario) ▪ Configuration page (testing)
Shortcut key	**`Del`**
Icon	None

Purpose

To remove an item from the currently loaded project.

Result

If you execute the command on the Scenario, Road, or Testing node, only the item of the linked file is removed from the currently loaded project, but the files are still in the pool. If you execute the command on a pool file, ModelDesk checks whether the element is linked to the experiment. If it is not linked, the element is removed from the project irretrievably.

Rename

Access

You can access this command via:

Ribbon	None
Context menu of	**Project Navigator** – project **Project Navigator** – experiment

Shortcut key	None
Icon	None

Purpose

To rename a project or experiment.

Result

The name of the project or experiment is displayed in an edit field that lets you enter the new name.

The **Rename Project/Experiment** dialog opens that lets you rename the selected project or experiment.

Related topics

HowTos

How to Rename a Project or Experiment..... 20

References

Rename Project/Experiment or Save Project/Experiment As Dialog..... 49

Rename Project/Experiment or Save Project/Experiment As Dialog

Access

This dialog opens when you click one of the following commands:

- Save As / Save Project As
- Rename

Purpose

To specify the new name for a project or experiment.

Result

The project or experiment is saved under the specified name.

Dialog settings

Enter a new name for Lets you enter another project or experiment name. The name must be unique within the project and must not cause conflicts with existing folders. The name must not exist and must not contain any of the following characters: `* ? | < > : / \ "`

Related topics

HowTos

How to Rename a Project or Experiment 20

References

Rename 48
Save As / Save Project As 50

Save / Save Project + Experiment

Access

You can access this command via:

Ribbon	File
Context menu of	▪ Project Navigator – project ▪ Project Navigator – experiment
Shortcut key	`Ctrl+Shift+S`
Icon	

Purpose

To save the loaded project and the experiment.

Save As / Save Project As

Access

You can access this command via:

Ribbon	File – Save As
Context menu of	Project Navigator – project Project Navigator – experiment
Shortcut key	None
Icon	None

Purpose

To save a project or experiment under a different name.

Result

The name of the project or experiment is displayed in an edit field that lets you enter the new name.

The **Save Project/Experiment As** dialog opens that lets you rename the selected project or experiment.

Related topics

HowTos

How to Rename a Project or Experiment..... 20

References

Rename Project/Experiment or Save Project/Experiment As Dialog..... 49

Specify Platform Dialog

Access

You can access this command via:

Ribbon	None
Context menu of	None
Shortcut key	None
Icon	None
Button	**Specify Platform** on the **Check Model for Consistency** dialog

Purpose

To select a dSPACE real-time system or VEOS platform for parameterization.

Result

ModelDesk opens the **Specify platform** dialog for you to specify the platform.

Dialog settings

Platform type Displays the platform type which is required to run the selected real-time application or offline simulation application.

Platform name Lets you select the platform. You must have registered the platform before. Refer to How to Register a Platform (ModelDesk Platform Management 🕮).

Related topics

HowTos

How to Choose a Model and Initialize the Consistency Check (ModelDesk Parameterizing 🕮)
How to Replace a Model and Initialize the Consistency Check (ModelDesk Parameterizing 🕮)

References

Change Model File (ModelDesk Parameterizing 🕮)
Update Model (ModelDesk Parameterizing 🕮)

Automation

Where to go from here

Information in this section

Programming ModelDesk Automation

Where to go from here

Information in this section

Handling Projects and Experiments in Python

Introduction

Before you can change parameter values or do other automation tasks, you must access ModelDesk and the experiment.

Automating projects created with earlier ModelDesk versions

You cannot open projects created with earlier ModelDesk versions via automation. You must migrate them first. Refer to Migrating a Project and Its Experiments on page 23.

Using the Interpreter

If you use the Interpreter in ModelDesk, an `Application` object is already available that you can use for your automation task. Refer to Accessing the Running ModelDesk Application from the Interpreter (ModelDesk Automation 🕮).

Accessing a ModelDesk experiment

The following example shows how you create a project and an experiment. Replace the project name by your own project in the listing below.

```
from win32com.client import Dispatch
# Start ModelDesk
Application = Dispatch("ModelDesk.Application")
# Set ModelDesk visible
Application.Visible = True
# Open the project (replace the name by your project name)
MyProject = Application.OpenProject(r"C:\ExamplePath\Example_001\Example_001.CDP")
# or create a new project
# MyProject = Application.NewProject(r"C:\ExamplePath", r"Example_001", True)
# Get object for Experiments collection
MyExperiments = MyProject.Experiments
```

```
# Get object of the first experiment
MyExperiment = MyExperiments.Item(0)
# or create a new experiment
# MyExperiment = MyExperiments.Add(r"MyExperimentName")
# Activate the experiment
MyActiveExperiment = MyExperiment.Activate(False)
```

The `Dispatch("ModelDesk.Application")` function starts ModelDesk and returns an object which is used to handle ModelDesk. ModelDesk is invisible when started by default. To make it visible, the `Visible` attribute must be set.

The `OpenProject` method loads the ModelDesk project.

The next objects are used to activate an experiment: The `MyExperiments` object gets a collection of all experiments. The `MyExperiment` object gets the object of the first experiment contained in the `MyExperiments` collection (`MyExperiments.Item(0)`). Finally, `MyExperiment` is activated and the `MyActiveExperiment` object is returned.

The `MyActiveExperiment` object can be used to access the active parameter sets, road, or scenario, refer to

- Modifying the Values of Model Parameters in Python (ModelDesk Parameterizing 🕮)
- Automatic Processing (ModelDesk Processing 🕮)
- Automated Plotting of Simulation Signals in MATLAB (ModelDesk Plotting 🕮)
- Modifying a Road in Python (ModelDesk Road Creation 🕮)
- Automating Scenarios in Python (ModelDesk Scenario Creation 🕮)
- Basics on the Automation of Testing (ModelDesk Testing 🕮)

Closing the automation

```
# Save the project
MyProject.Save()
# Exit ModelDesk
Application.Quit(False)
# Delete Application object
del Application
```

The `Save` method of the `Project` class saves the project.

The `Quit` method of the `Application` class exits ModelDesk.

Related topics

References

Overview of the Object Model for Accessing ModelDesk Experiments

Introduction

The object model overview of the ModelDesk automation interface gives a quick overview of object dependencies, and available object attributes and methods.

Symbols

The following symbols are used in the object model overview:

Symbol	Description
	Method, function
	Attribute (property, class)
	Collection
0, 1, 2, …	Level of dependency (0, 1, 2, …)
⊗	Read only

Application

The following table gives an overview of the Application's object model:

Application	0
Project ActiveProject ⊗	
Boolean Visible	
Project Project ⊗	1
String Fullname ⊗	
Boolean IsModified ⊗	
String Name ⊗	

Application	
ActiveExperiment ActiveExperiment ⊗	
Road RoadNetwork ⊗[1)]	
ManeuverControl ManeuverControl ⊗	
Start(Boolean DownloadExperimentIfApplicationNotRunning)	
Stop()	
Reset()	
ActiveTrafficScenario TrafficScenario[2)]	
ActiveParameterSet ActiveParameterSet ⊗[3)]	
String Name ⊗	
String FullName ⊗	
None Save()	
None Download()	
None ActivateRoad(variant Road)	
None ActivateTrafficScenario(String ScenarioName)	
ParameterSets ParameterSets ⊗[3)]	
Models Models ⊗	
ModelInfo ActiveModelInfo ⊗	
ModelInfo ActivateModel(PlatformKind TargetPlatformKind)	
ModelInfo GetModelInfo(PlatformKind TargetPlatformKind)	
ModelInfo SetActiveModel(String ModelFilePath, string MLWorkingDirectory, string InitializationCommand, string PlatformName)	
ModelInfo UpdateActiveModel()	
TrafficDriverManager TrafficDrivers[4)] ⊗	
String WorkingRoot ⊗	
None Close(Boolean Save)	
None Save()	
Experiments Experiments ⊗	
long Count ⊗	
Strings Names ⊗	
None Remove(variant Index)	
Experiment Item(variant Index)	
String FullName ⊗	
String Name ⊗	
ActiveExperiment Activate(Boolean DoSaveActive)	
Experiment Add(string ExperimentName)	
ModelInfo AnalyzeModel(String ModelFilePath)	

Application	
ModelValidity Validity ⊗	[3]
string Path ⊗	
PlatformKind PlatformKind ⊗	
string ParseResult ⊗	
string ModelDiagnostics ⊗	
string MATLABWorkingDirectory ⊗	
string ModelInitializationCommand ⊗	
string PlatformName ⊗	
Pool Pool ⊗	[2]
Object Import(String FilePath, Boolean OverWriteExisting)	[3]
Boolean CreateFile(ContentTypes Type, string Name, Boolean OverWriteExisting)	
Project OpenProject(String ProjectName, String InitialExperiment=" ", Boolean SaveActive=true)	
Project NewProject(string ProjectPath, string ProjectName, Boolean SaveActive = true)	
None Quit(Boolean SaveAll)	
ProjectRoots ProjectRoots ⊗	[1]
long Count ⊗	
ProjectRoot ActiveRoot ⊗	
ProjectRoot Add(String Path)	
None Remove(int Index)	
ProjectRoot Item(int Index) ⊗	[2]
String Path ⊗	
Strings ProjectNames ⊗	
None Activate	

1) Refer to Overview of the Classes for Working with Roads (ModelDesk Road Creation 🕮).
2) Refer to Overview of the Classes for Creating Scenarios (ModelDesk Scenario Creation 🕮).
3) Refer to Overview of the Object Model for Parameterizing (ModelDesk Parameterizing 🕮).
4) Refer to Classes for Configuring Traffic Drivers (ModelDesk Scenario Creation 🕮).

Related topics

Basics

Handling Projects and Experiments in Python..... 54

References

Classes for Accessing ModelDesk Experiments..... 59

Classes for Accessing ModelDesk Experiments

Where to go from here

Information in this section

Information in other sections

ActiveExperiment

Purpose

To handle a ModelDesk experiment.

Where to go from here

Information in this section

Information in other sections

Class Description (ActiveExperiment)

Syntax

No direct creation

Purpose

To handle a ModelDesk experiment.

Attributes

The class contains the following attributes:

Attributes	Type	Purpose
ActiveParameterSet	ActiveParameterSet[1)]	To get the active parameter set of the experiment.
Aliases	Aliases[2)]	To get the aliases of the experiment.
FullName	String	To get the absolute path of the experiment file.
ManeuverControl	ManeuverControl[3)]	To get the ManeuverControl object for controlling the maneuver.
Models	Models[4)]	To access the model of the experiment for analyzing, updating, or activating.
Name	String	To get the name of the active experiment.
ParameterSets	ParameterSets[5)]	To get the ParameterSets collection of the experiment.
Plotting	ActivePlotting[6)]	To get the active plotting object.
Road	RoadNetwork[7)]	To get the active road of the experiment.
Test	Test[8)]	To get the active test.
TrafficScenario	ActiveTrafficScenario[9)]	To get the active scenario object.

1) Refer to ActiveParameterSet (ModelDesk Parameterizing 🕮).
2) Refer to Aliases (ModelDesk Testing 🕮).
3) Refer to ManeuverControl (ModelDesk Scenario Creation 🕮).
4) Refer to Models on page 76.
5) Refer to ParameterSets (Collection) (ModelDesk Parameterizing 🕮).
6) Refer to ActivePlotting (ModelDesk Plotting 🕮).
7) Refer to RoadNetwork (ModelDesk Road Creation 🕮).
8) Refer to Test (ModelDesk Testing 🕮).
9) Refer to ActiveTrafficScenario (ModelDesk Scenario Creation 🕮).

Methods

The class contains the following methods:

Method	Purpose
ActivatePlotting	To activate a plotting for the experiment. Refer to ActivatePlotting on page 62.
ActivateRoad	To activate the specified road for the experiment. Refer to ActivateRoad on page 62.
ActivateTest	To activate a test for the experiment. Refer to ActivateTest on page 63.
ActivateTrafficScenario	To activate a scenario for the experiment. Refer to ActivateTrafficScenario on page 64.
Download	To download the active parameter set, the active scenario and the active road to the real-time hardware, Simulink, or VEOS. Refer to Download on page 64.
Save	To save the active experiment. Refer to Save on page 65.

Related topics

Basics

Handling Projects and Experiments in Python..... 54

ActivatePlotting

Class

ActiveExperiment

Syntax

```
ActiveExperiment.ActivatePlotting(String PlottingName)
```

Purpose

To activate the specified plotting for the experiment.

Parameters

The method uses the following parameters:

Parameter	Type	Description
PlottingName	String	The name of the plotting.

Return value

–

Related topics

References

Class Description (ActiveExperiment)..... 60

ActivateRoad

Class

ActiveExperiment

Syntax

```
ActiveExperiment.ActivateRoad(variant Road)
```

Purpose

To activate the specified road for the experiment.

Parameters

The method uses the following parameters:

Parameter	Type	Description
Road	variant	The specified road for the experiment.

Return value

–

Related topics

References

Class Description (ActiveExperiment)..... 60

ActivateTest

Class

ActiveExperiment

Syntax

```
ActiveExperiment.ActivateTest(String RelativePath)
```

Purpose

To activate the specified test for the experiment.

Parameters

The method uses the following parameters:

Parameter	Type	Description
RelativePath	String	The relative path to the test.

Return value

–

Related topics

References

Class Description (ActiveExperiment)..... 60

ActivateTrafficScenario

Class

ActiveExperiment

Syntax

```
ActiveExperiment.ActivateTrafficScenario(String ScenarioName)
```

Purpose

To activate the specified scenario for the experiment.

Parameters

The method uses the following parameters:

Parameter	Type	Description
ScenarioName	String	The name of the scenario.

Return value

–

Related topics

References

Class Description (ActiveExperiment)..... 60

Download

Class

ActiveExperiment

Syntax

```
ActiveExperiment.Download()
```

Purpose

To download the active parameter set, the active scenario and the active road to the real-time hardware, Simulink, or VEOS.

Parameters

–

Return value	–
Related topics	References Class Description (ActiveExperiment)..... 60

Save

Class	ActiveExperiment
Syntax	`ActiveExperiment.Save()`
Purpose	To save the active experiment.
Parameters	–
Return value	–
Related topics	References Class Description (ActiveExperiment)..... 60

Application

Purpose	To access ModelDesk for automation.
Where to go from here	Information in this section Class Description (Application)..... 66 To describe the class and its attributes.

Information in other sections

Class Description (Application)

Syntax

```
Application = Dispatch("ModelDesk.Application")
```

Purpose

To access ModelDesk for automation.

Tip

An `Application` object of ModelDesk is already created in ModelDesk's Interpreter. When you use ModelDesk's Interpreter, you can use this object in your automation.

Attributes

The class contains the following attributes:

Attributes	Type	Purpose
ActiveProject	Project[1)]	The active and currently loaded project.
ProjectRoots	ProjectRoots[2)]	The collection of project root folders.
Visible	Boolean	To get/set the visible state of the ModelDesk. True if ModelDesk is visible.

1) Refer to Project on page 83.
2) Refer to ProjectRoots (Collection) on page 88.

Methods

The class contains the following methods:

Method	Purpose
NewProject	To create a new project. Refer to NewProject on page 67.
OpenProject	To load a project. Refer to OpenProject on page 68.
Quit	To close the project and the graphical user interface. Refer to Quit on page 69.

Related topics

Basics

Handling Projects and Experiments in Python..... 54

NewProject

Class

Application

Syntax

```
Project = Application.NewProject(string ProjectPath, string
ProjectName, boolean SaveActive = true)
```

Purpose

To create a new project.

Parameters

The method uses the following parameters:

Parameter	Type	Description
ProjectPath	String	The folder without the file name of the new project.
ProjectName	String	The name of the new project.
SaveActive	Boolean	Defines if the opened project is to be saved. True if the project is saved (default).

Return value

The method returns an object of the following type:

Type	Description
Project[1)]	Instance of a project class.

[1)] Refer to Project on page 83.

Related topics

References

Class Description (Application).....66

OpenProject

Class

Application

Syntax

```
Project = Application.OpenProject(string ProjectName,
 string InitialExperiment="", boolean SaveActive = true)
```

Purpose

To load a project.

Parameters

The method uses the following parameters:

Parameter	Type	Description
ProjectName	String	The path of the CDP file of the loaded project.
InitialExperiment	String	The name of the experiment to be initially activated.
SaveActive	Boolean	If a project is currently open, you can specify to save it before it is closed. True if the project is saved (default).

Return value

The method returns an object of the following type:

Type	Description
Project[1)]	Instance of a project class.

1) Refer to Project on page 83.

Related topics

References

Class Description (Application).....66

Quit

Class

Application

Syntax

```
Application.Quit(boolean SaveAll)
```

Purpose

To close the project and the graphical user interface.

Parameters

The method uses the following parameters:

Parameter	Type	Description
SaveAll	Boolean	Defines if the project is saved before closing.

Return value

–

Related topics

References

Class Description (Application)..... 66

Experiment

Purpose

To handle an experiment for a project.

Where to go from here

Information in this section

Class Description (Experiment)..... 70
To describe the class and its attributes.

Activate..... 70
To activate the experiment.

Information in other sections

Handling Projects and Experiments in Python..... 54
Before you can change parameter values or do other automation tasks, you must access ModelDesk and the experiment.

Class Description (Experiment)

Syntax

No direct creation

Purpose

To handle an experiment for a project.

Attributes

The class contains the following attributes:

Attributes	Type	Purpose
FullName	String	To get the absolute path of the experiment file.
Name	String	To get the experiment's name.

Methods

The class contains the following methods:

Method	Purpose
Activate	To activate the experiment.

Related topics

Basics

Handling Projects and Experiments in Python..... 54

Activate

Class

Experiment

Syntax

```
ActiveExperiment = Experiment.Activate(boolean DoSaveActive)
```

Purpose

To activate the experiment.

Parameters

The method uses the following parameters:

Parameter	Type	Description
DoSaveActive	Boolean	Defines if the currently active experiment is saved before the selected experiment is activated.

Return value

The method returns an object of the following type:

Type	Description
ActiveExperiment	The activated experiment.

Related topics

References

Class Description (Experiment)..... 70

Experiments (Collection)

Purpose

To handle the collection of the experiments of a project.

Where to go from here

Information in this section

Information in other sections

Handling Projects and Experiments in Python..... 54
Before you can change parameter values or do other automation tasks, you must access ModelDesk and the experiment.

Class Description (Experiments (Collection))

Syntax

```
Experiments = Project.Experiments
```

Purpose

To handle the collection of the experiments of a project.

Attributes

The class contains the following attributes:

Attributes	Type	Purpose
Count	Long	To get the number of experiments in the collection.
Names	Strings	To get a list of names of experiments in the collection.

Methods

The class contains the following methods:

Method	Purpose
Add	To add an experiment to the project. Refer to Add on page 73.
Item	To get the experiment specified by the given index. Refer to Item on page 73.
Remove	To remove the specified experiment from the collection. Refer to Remove on page 74.

Related topics

Basics

Handling Projects and Experiments in Python..... 54

Add

Class

Experiments

Syntax

```
Experiment = Experiments.Add(string ExperimentName)
```

Purpose

To add an experiment to the project.

Parameters

The method uses the following parameters:

Parameter	Type	Description
ExperimentName	string	Name of the experiment.

Return value

The method returns an object of the following type:

Type	Description
Experiment[1)]	The added experiment.

[1)] Refer to Experiment on page 69.

Related topics

References

Class Description (Experiments (Collection)) 72

Item

Class

Experiments

Syntax

```
Experiment = Experiments.Item(variant Index)
```

Purpose

To get the experiment specified by the given Index.

Parameters

The method uses the following parameters:

Parameter	Type	Description
Index	Variant	Index of the experiment to be returned. The index can be numeric (0 ≤ index < number of elements in the collection) or the experiment's name (for example, Experiment_001).

Return value

The method returns an object of the following type:

Type	Description
Experiment	The specified experiment.

Related topics

References

Class Description (Experiments (Collection)) 72

Remove

Class

Experiments

Syntax

```
Experiments.Remove()
```

Purpose

To remove the specified experiment from the collection.

Parameters

The method uses the following parameters:

Parameter	Type	Description
Index	Variant	Index of the experiment to be returned. The index can be numeric (0 ≤ index < number of elements in the collection) or the experiment's name (for example, Experiment_001).

Return value

–

Related topics

References

Class Description (Experiments (Collection)).....72

ModelInfo

Purpose

To get information on the model.

Class Description (ModelInfo)

Syntax

```
ModelInfo = Project.AnalyzeModel(string ModelFilePath)
```

Purpose

To get information on the model.

Attributes

The class contains the following attributes:

Attributes	Type	Purpose
Validity	ModelValidity[1)]	To get validity state.
Path	String	To get the path of the model.
PlatformKind	PlatformKind[2)]	To get the platform kind.
ParseResult	String	To get the result of the model parse.
ModelDiagnostics	String	To get diagnostic information of the model.
MATLABWorkingDirectory	String	To get the MATLAB working directory.
ModelInitializationCommand	String	To get the model initialization command (the m file for initialization the model).
PlatformName	String	To get the platform name or board name if the platform is a real-time system.

1) Refer to ModelValidity on page 92.
2) Refer to PlatformKind on page 93.

Related topics

References

AnalyzeModel.....85

Models

Purpose

To analyze, update, or activate models.

Where to go from here

Information in this section

Class Description (Models)

Syntax

```
Models = ActiveExperiment.Models
```

Purpose

To analyze, update, or activate models.

Attributes

The class contains the following attributes:

Attributes	Type	Purpose
ActiveModelInfo	ModelInfo[1)]	To get information on the active model.

[1)] Refer to ModelInfo on page 75.

Methods

The class contains the following methods:

Method	Purpose
ActivateModel	To activate a model of a specific platform. Refer to ActivateModel on page 77.
GetModelInfo	To get information on the model of a specific platform. Refer to GetModelInfo on page 78.

Method	Purpose
SetActiveModel	To activate a model. Refer to SetActiveModel on page 78.
UpdateActiveModel	To update the active model. Refer to UpdateActiveModel on page 79.

Related topics

References

ActiveExperiment 60

ActivateModel

Class

Models

Syntax

```
ModelInfo = Models.ActivateModel(PlatformKind TargetPlatformKind)
```

Purpose

To activate a model of a specific platform.

Parameters

The method uses the following parameters:

Parameter	Type	Description
TargetPlatformKind	PlatformKind[1)]	Specifies the kind of the platform.

1) Refer to PlatformKind on page 93.

Return value

The method returns an object of the following type:

Type	Description
ModelInfo[1)]	The object containing information on the model.

1) Refer to ModelInfo on page 75.

Related topics

References

Class Description (Models) 76

GetModelInfo

Class

Models

Syntax

```
ModelInfo = Models.GetModelInfo(PlatformKind TargetPlatformKind)
```

Purpose

To get information on the model of a specific platform.

Parameters

The method uses the following parameters:

Parameter	Type	Description
TargetPlatformKind	PlatformKind[1)]	Specifies the kind of the platform.

1) Refer to PlatformKind on page 93.

Return value

The method returns an object of the following type:

Type	Description
ModelInfo[1)]	The object containing information on the model.

1) Refer to ModelInfo on page 75.

Related topics

References

Class Description (Models).....76

SetActiveModel

Class

Models

Syntax

```
ModelInfo = Models.SetActiveModel(String ModelFilePath, string MLWorkingDirectory,
string InitializationCommand, string PlatformName)
```

Purpose

To activate a model.

Note

When you use the `SetActiveModel` method, it can happen that the automation script continues although the method is not really completed. In such cases, let the script pause, for example, by adding a `time.sleep(<seconds>)` command in Python or a `pause(<seconds>)` command in MATLAB.

Parameters

The method uses the following parameters:

Parameter	Type	Description
ModelFilePath	String	Specifies the path and name of the model to be activated.
MLWorkingDirectory	String	Specifies the MATLAB working directory.
InitializationCommand	String	Specifies the initialization command (the m file used for initializing the model).
PlatformName	String	Specifies the platform name.

Return value

The method returns an object of the following type:

Type	Description
ModelInfo[1)]	The object containing information on the model.

[1)] Refer to ModelInfo on page 75.

Related topics

References

Class Description (Models) 76

UpdateActiveModel

Class

Models

Syntax

```
ModelInfo = Models.UpdateActiveModel()
```

Purpose

To update the active model.

Parameters

–

Return value

The method returns an object of the following type:

Type	Description
ModelInfo[1)]	The object containing information on the model.

1) Refer to ModelInfo on page 75.

Related topics

References

Class Description (Models)..... 76

Pool

Purpose

To import files to the Pool of the project.

Where to go from here

Information in this section

Class Description (Pool)..... 80
To describe the class and its attributes.

CreateFile..... 81
To create a new file in the pool.

Import..... 82
To import files to the Pool of the open project.

Class Description (Pool)

Syntax

```
Pool = Project.Pool
```

Purpose

To manage files of the Pool.

Attributes

–

Methods

The class contains the following methods:

Method	Purpose
Import	To import files to the Pool. Refer to Import on page 82.
CreateFile	To create a new scenario, road, or test file in the Pool. Refer to CreateFile on page 81.

Related topics

References

Project.....83

CreateFile

Class

Pool

Syntax

```
RetVal = Pool.CreateFile(ContentTypes Type, string Name, Boolean
OverWriteExisting)
```

Purpose

To create a new file in the pool.

Description

You can create files that have the following contents:

- Layout configuration
- Road
- Scenario
- Test
- Traffic driver

Parameters

The method uses the following parameters:

Parameter	Type	Description
Name	String	Specifies the name of the file to be created.
OverWriteExisting	Boolean	The new created file overwrites an existing file if true.

Parameter	Type	Description
Type	ContentTypes[1)]	Specifies the type of the file. Valid values are Road = 0 and Scenario = 1, for example.

[1)] Refer to ContentTypes on page 92.

Return value

The method returns the following parameter:

Type	Description
Boolean	True if successful.

Example

The following example shows how to create files of different contents. You can use the example in the Interpreter of ModelDesk when a project is loaded.

```
import dspace.com
Enums = dspace.com.Enums(Application)

MyProject = Application.ActiveProject
# To create a new road file
MyProject.Pool.CreateFile(Enums.ContentTypes.Road,'MyRoad',TRUE)
# To create a new scenario file
MyProject.Pool.CreateFile(Enums.ContentTypes.Scenario,'MyScenario',TRUE)
# To create a new test file
MyProject.Pool.CreateFile(Enums.ContentTypes.Test,'MyTest',TRUE)
# To create a new layout configuration file
MyProject.Pool.CreateFile(Enums.ContentTypes.LayoutConfiguration,'MyLayout',TRUE)
# To create a new traffic driver file
MyProject.Pool.CreateFile(Enums.ContentTypes.TrafficDriver,'MyTrafficDriver',TRUE)
```

Related topics

References

Class Description (Pool) 80

Import

Class

Pool

Syntax

```
RetVal = Pool.Import(string FilePath, Bool OverWriteExisting)
```

Purpose

To import files to the Pool of the open project.

Parameters

The method uses the following parameters:

Parameter	Type	Description
FilePath	String	Specifies the file to be imported.
OverWriteExisting	Boolean	Importing files overwrites existing files if true.

Return value

The method returns an object of the following type:

Type	Description
Object	An object containing information on the import.

Related topics

References

Class Description (Pool)..... 80
Import (ModelDesk Basics 🕮)

Project

Purpose

To handle the ModelDesk project.

Where to go from here

Information in this section

Class Description (Project)..... 84
To describe the class and its attributes.

AnalyzeModel..... 85
To analyze a model.

Close..... 85
To close the project.

Save..... 86
To save the project.

Information in other sections

Handling Projects and Experiments in Python..... 54
Before you can change parameter values or do other automation tasks, you must access ModelDesk and the experiment.

Class Description (Project)

Syntax

```
Project = Application.OpenProject()
```

Purpose

To handle the ModelDesk project.

Attributes

The class contains the following attributes:

Attributes	Type	Purpose
Fullname	String	To get the absolute path of the project.
IsModified	Boolean	To get information on whether the project was modified.
ModelConfiguration	ModelConfiguration[1)]	To modify the contents of the model configuration.
Name	String	To get the name of the project.
ActiveExperiment	ActiveExperiment[2)]	To get the active experiment of the project.
Experiments	Experiments[3)]	To get the collection of experiments.
Pool	Pool[4)]	To get the pool object.
TrafficDrivers	TrafficDrivers[5)]	To get the traffic drivers.
TrafficObjectManager	TrafficObjectManager[6)]	To get the traffic object manager.
WorkingRoot	String	To get the folder where the project is saved.

1) Refer to ModelConfiguration (ModelDesk Scenario Creation 🕮).
2) Refer to ActiveExperiment on page 60.
3) Refer to Experiments (Collection) on page 71.
4) Refer to Pool on page 80.
5) Refer to TrafficDrivers (ModelDesk Scenario Creation 🕮).
6) Refer to TrafficObjectManager (ModelDesk Traffic Object Management 🕮).

Methods

The class contains the following methods:

Method	Purpose
Close	To close the project. Refer to Close on page 85.
Save	To save the project. Refer to Save on page 86.
AnalyzeModel	To analyze a model. Refer to AnalyzeModel on page 85.

Related topics

References

AnalyzeModel

Class

Project

Syntax

```
ModelInfo = Project.Analyze(String ModelFilePath)
```

Purpose

To analyze a model.

Description

You can analyze models which are not part of the experiment. The method returns the `ModelInfo` object that contains information of the analyzed model.

Parameters

The method uses the following parameters:

Parameter	Type	Description
ModelFilePath	String	Specifies the path and name of the model to be analyzed.

Return value

The method returns an object of the following type:

Type	Description
ModelInfo[1)]	The object containing information on the model.

1) Refer to ModelInfo on page 75.

Related topics

References

Class Description (Project) 84

Close

Class

Project

Syntax

```
Project.Close(boolean Save)
```

Purpose

To close the project.

Parameters

The method uses the following parameters:

Parameter	Type	Description
Save	Boolean	Defines if the project is saved before closing.

Return value

–

Related topics

References

Class Description (Project) 84

Save

Class

Project

Syntax

```
Project.Save()
```

Purpose

To save the project.

Parameter

–

Return value

–

Related topics

References

Class Description (Project) 84

ProjectRoot

Purpose

To handle the project root folder.

Where to go from here

Information in this section

Information in other sections

Class Description (ProjectRoot)

Syntax

No direct creation

Purpose

To handle the project root folder.

Attributes

The class contains the following attributes:

Attributes	Type	Purpose
Path	String	To get the absolute path of the project root folder.
ProjectNames	Strings	To get a list of the project names in the project root folder.

Methods

The class contains the following methods:

Method	Purpose
Activate	To activate the project root folder. Refer to Activate on page 88.

Related topics

Basics

Handling Projects and Experiments in Python..... 54

Activate

Class

ProjectRoot

Syntax

```
ProjectRoot.Activate()
```

Purpose

To activate the project root folder.

Parameter

–

Return value

–

Related topics

References

Class Description (ProjectRoot)..... 87

ProjectRoots (Collection)

Purpose

To handle the collection of project root folders.

Where to go from here

Information in this section

Class Description (ProjectRoots)..... 89
To describe the class and its attributes.

Information in other sections

Class Description (ProjectRoots)

Syntax

No direct creation

Purpose

To handle the collection of project root folders.

Attributes

The class contains the following attributes:

Attributes	Type	Purpose
Count	Long	To get the number of project root folders.
ActiveRoot	Project[1)]	To get the active project root folder.

[1)] Refer to Project on page 83.

Methods

The class contains the following methods:

Method	Purpose
Add	To add a project root folder specified by the given path. Refer to Add on page 90.
Item	To get the project root folder specified by the given index. Refer to Item on page 91.
Remove	To remove the project root folder specified by the given index. Refer to Remove on page 91.

Related topics

Basics

Basics of Projects and Experiments..... 10

Add

Class

ProjectRoots

Syntax

```
ProjectRoot = ProjectRoots.Add(string Path)
```

Purpose

To add a project root folder to the collection.

Parameters

The method uses the following parameters:

Parameter	Type	Description
Path	String	The path of the project root folder added to the collection.

Return value

The method returns an object of the following type:

Type	Description
ProjectRoot[1)]	The specified project root.

1) Refer to ProjectRoot on page 87.

Related topics

References

Class Description (ProjectRoots)..... 89

Item

Class

ProjectRoots

Syntax

```
ProjectRoot = ProjectRoots.Item(integer Index)
```

Purpose

To get the project root folder specified by the given index.

Parameters

The method uses the following parameters:

Parameter	Type	Description
Index	Integer	Index of the project root to be returned.

Return value

The method returns an object of the following type:

Type	Description
ProjectRoot[1)]	The specified project root.

[1)] Refer to ProjectRoot on page 87.

Related topics

References

Class Description (ProjectRoots) 89

Remove

Class

ProjectRoots

Syntax

```
ProjectRoots.Remove(integer Index)
```

Purpose

To remove a project root folder from the collection.

Parameters

The method uses the following parameters:

Parameter	Type	Description
Index	Integer	The project root folder to be removed from the collection.

Return value

–

Related topics

References

Class Description (ProjectRoots) 89

Enumerations

Enumerations for Project and Experiment Management

Introduction

You can use predefined constants in the tool automation.

Enumerations

ContentTypes The following constants are used to specify the content type:

Value	Description
Road = 0	Road file
Scenario = 1	Scenario file
Test = 2	Test file
LayoutConfiguration = 3	Layout configuration file
TrafficDriver = 4	Traffic driver file

ModelValidity The following constants are used to specify the validity of the model:

Value	Description
mvUndefined = 0	Undefined
mvValid = 1	Model is valid.
mvInvalidPath = 2	Model has invalid path.
mvNoActualModel = 3	A model description is missing in SDF files.
mvUnexpectedActualModelVersion = 4	Model has an unexpected version.

Value	Description
mvNoMATLABConnection = 5	No connection to MATLAB.
mvDefectiveModel = 6	Model is defect.
mvModelWithTheSameNameOpen = 7	A model with the same name is already open.

PlatformKind The following constants are used to specify the platform type:

Value	Description
plkUndefined = 0	Undefined
plkSimulink = 1	Simulink
plkPHS = 2	PHS-bus-based system (DS1006, DS1007) and MicroLabBox
plkVEOS = 3	VEOS
plkSCALEXIO = 4	SCALEXIO

Related topics

References

Overview of the Object Model for Accessing ModelDesk Experiments..... 56